

Software Engineering and Foundation Models: Insights from Industry Blogs Using a Jury of Foundation Models

Hao Li
Queen's University
Kingston, Canada
hao.li@queensu.ca

Cor-Paul Bezemer
University of Alberta
Edmonton, Canada
bezemer@ualberta.ca

Ahmed E. Hassan
Queen's University
Kingston, Canada
ahmed@cs.queensu.ca

Abstract—Foundation models (FMs) such as large language models (LLMs) have significantly impacted many fields, including software engineering (SE). The interaction between SE and FMs has led to the integration of FMs into SE practices (FM4SE) and the application of SE methodologies to FMs (SE4FM). While several literature surveys exist on academic contributions to these trends, we are the first to provide a practitioner's view. We analyze 155 FM4SE and 997 SE4FM blog posts from leading technology companies, leveraging an FM-powered surveying approach to systematically label and summarize the discussed activities and tasks. We observed that while code generation is the most prominent FM4SE task, FMs are leveraged for many other SE activities such as code understanding, summarization, and API recommendation. The majority of blog posts on SE4FM are about model deployment & operation, and system architecture & orchestration. Although the emphasis is on cloud deployments, there is a growing interest in compressing FMs and deploying them on smaller devices such as edge or mobile devices. We outline eight future research directions inspired by our gained insights, aiming to bridge the gap between academic findings and real-world applications. Our study not only enriches the body of knowledge on practical applications of FM4SE and SE4FM but also demonstrates the utility of FMs as a powerful and efficient approach in conducting literature surveys within technical and grey literature domains. Our dataset, results, code and used prompts can be found in our online replication package at <https://zenodo.org/records/14563992>.

Index Terms—Foundation models, FM4SE, SE4FM, LLM-as-a-judge, industry trends, LLM

I. INTRODUCTION

In recent years, the rapid advancements in machine learning (ML) have fundamentally transformed various fields, including software engineering (SE). Among these developments, foundation models (FMs) such as large language models (LLMs) have emerged as a major force, reshaping how software is developed, tested, and maintained [17]. The interaction between SE and FMs has led to the emergence of two key trends: (1) FMs for SE (FM4SE), where FMs are leveraged to automate or enhance various SE tasks, such as code generation and testing, and (2) SE for FMs (SE4FM), where SE practices are adapted to the development and deployment of FMs.

Academic research has made significant strides in exploring these trends, but literature surveys have only focused on

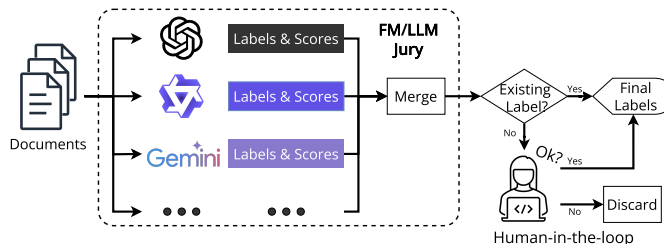


Fig. 1. An overview of our FM/LLM Jury approach for labelling blog posts. A blog post is labelled by every FM in the jury, and the final label is selected through a majority vote (using the normalized confidence score as a tie-breaker).

published, peer-reviewed literature [17], [28], mostly leaving out the perspectives and experiences of industry practitioners. The input of practitioners, who work at the intersection of SE and FMs in real-world settings, is a crucial yet under-explored source of insights. While the research community has recognized the value of user-generated contents such as Q&A websites [2] and issue reports [32], less attention has been paid to grey literature such as technical blog posts from industry leaders. Tech companies publish blog posts for several reasons, including positioning themselves as innovation leaders and establishing thought leadership [4]. As a result, these blog posts often provide in-depth discussions on cutting-edge challenges and solutions in SE and FM integration.

To bridge the gap between academic findings and industry practices, we analyze blog posts from leading technology companies, focusing on how practitioners discuss the challenges and approaches related to FM4SE and SE4FM. By systematically labelling and examining these blog posts, we seek to provide a clearer picture of how FMs are being integrated into the SE domain (i.e., FM4SE), and how SE principles are being applied to FMs (i.e., SE4FM) in industry. This study stands out by offering a synthesized industry voice, derived directly from real-world, practitioner-driven insights. We employ an ensemble of FMs as judges [50] into an FM/LLM Jury [42] (see Figure 1) to assist with the labelling and synthesis of knowledge within 155 FM4SE and 997 SE4FM blog posts.

Our study focuses on these research questions (RQs):

RQ1. Which FM4SE activities are discussed in industry blog posts? Software development tasks, particularly code generation, are the most frequently discussed across FM4SE blogs. FMs are increasingly integrated as code assistants, providing developers with multifunctional tools to boost productivity. Vulnerability detection is the dominant software quality assurance task, while software maintenance activities primarily focus on refactoring and transforming existing codebases.

RQ2. Which SE4FM activities are discussed in industry blog posts? The most discussed activities in SE4FM blog posts are model deployment & operation, with a focus on cloud hosting and model serving & scaling. Other trends include prompt chaining, workflow orchestration, and building AI agents. Data management activities focus on RAG and vector databases to support unstructured data and information retrieval. Model customization relies on fine-tuning methods such as full fine-tuning, LoRA, and RLHF.

The main contributions of this paper are:

- The first study of industry blog posts on FM4SE and SE4FM to provide the practitioner’s view on these emerging and crucial topics in today’s software industry.
- A dataset of 1,152 blog posts from top technology companies related to FM4SE or SE4FM.
- A list of eight research directions that are driven by the findings of our survey on blog posts.
- A demonstration of an efficient approach that leverages a jury of FMs to assist with grey literature surveys on SE-related topics.

Paper Organization. The rest of this paper is organized as follows. Section II presents background information and related work. Section III details the proposed FM/LLM Jury framework. Section IV presents our methodology. Sections V and VI present the findings of our research questions. Section VII discusses promising future research directions that follow from our survey. Section VIII discusses the threats to the validity our study. Section IX concludes this paper.

II. BACKGROUND AND RELATED WORK

A. LLM-as-a-judge

Leveraging FMs/LLMs as evaluators, or LLM-as-a-judge, has emerged as a scalable alternative to traditional human evaluations to assess the quality of outputs from LLMs [21]. This assessment is not straightforward, as LLMs generate natural language, which needs to be compared with a ground truth semantically. LLM-as-a-judge builds on the idea that state-of-the-art models, especially those trained with Reinforcement Learning from Human Feedback (RLHF) [36] (e.g., GPT-4) are well-aligned with human judgments, making them promising substitutes [50]. While the use of FMs like GPT-4 as evaluators has become more common, these models often exhibit biases, such as favouring their own outputs over those from other models [38]. To mitigate these biases, researchers

have proposed the use of a panel of FM evaluators instead of relying on a single model [42]. Instead of using a single FM/LLM to evaluate FM/LLM outputs, in this paper, we propose using a jury of FMs/LLMs to assist with the labelling and summarization of industry blog posts.

B. Related work

The work that is closest related to our work consists of other literature surveys on FM4SE and SE4FM. Recent comprehensive surveys on FM4SE have examined the rapidly growing field of FMs/LLMs applied to SE activities and tasks. Hou et al. [17] conducted a systematic survey of 395 studies covering the application of LLMs to 84 specific SE tasks across 6 SE activities. In addition, Wang et al. [44] surveyed 102 studies that have used LLMs for software testing. These applications have shown promise, but several challenges remain. For example, Fan et al. [12] emphasized technical challenges like hallucinations when applying LLMs for SE, and highlighted the importance of hybrid techniques that combine traditional SE with LLMs.

Other surveys (on SE4FM) focus on how established SE practices can be adapted to support building, testing, deploying and maintaining FMs. Chang et al. [5] reviewed evaluation methods and benchmarks for LLMs in different areas such as education and social sciences, highlighting the importance of robust benchmarks to assess the performance of LLMs. Most prior surveys on SE for models have focused on SE4ML rather than SE4FM (i.e., SE for machine learning models that are not foundation models). Martínez-Fernández et al. [28] reviewed 248 studies and classified them based on the Software Engineering Body of Knowledge (SWEBOK), identifying gaps in areas like maintenance and data handling. Villamizar et al. [43] discussed gaps in requirements engineering for ML, while Masuda et al. [29] surveyed quality assurance approaches, highlighting the need for specialized testing techniques in verifying an ML system’s output.

All prior surveys on FM4SE and SE4FM focused on academic efforts. Our work is the first to provide an overview of FM4SE and SE4FM activities in top technology companies.

III. USING AN FM/LLM JURY FOR LABELLING BLOG POSTS

Labelling blog posts using a single frontier FM (e.g., GPT-4o) in the LLM-as-a-judge approach [50] can be both expensive and potentially biased. To address these limitations, we propose FM/LLM Jury, a methodology that leverages multiple FMs to collaboratively label blog posts. In this framework, each model provides a label along with a confidence score, and these outputs are merged using a majority vote to determine a final label. This framework is inspired by Verga et al. [42].

A. Constructing the prompt

The prompt construction process is iterative and consists of the following key steps:

Step 1 – Create the golden dataset. To evaluate the performance of the prompts that we send to the FM and the quality of

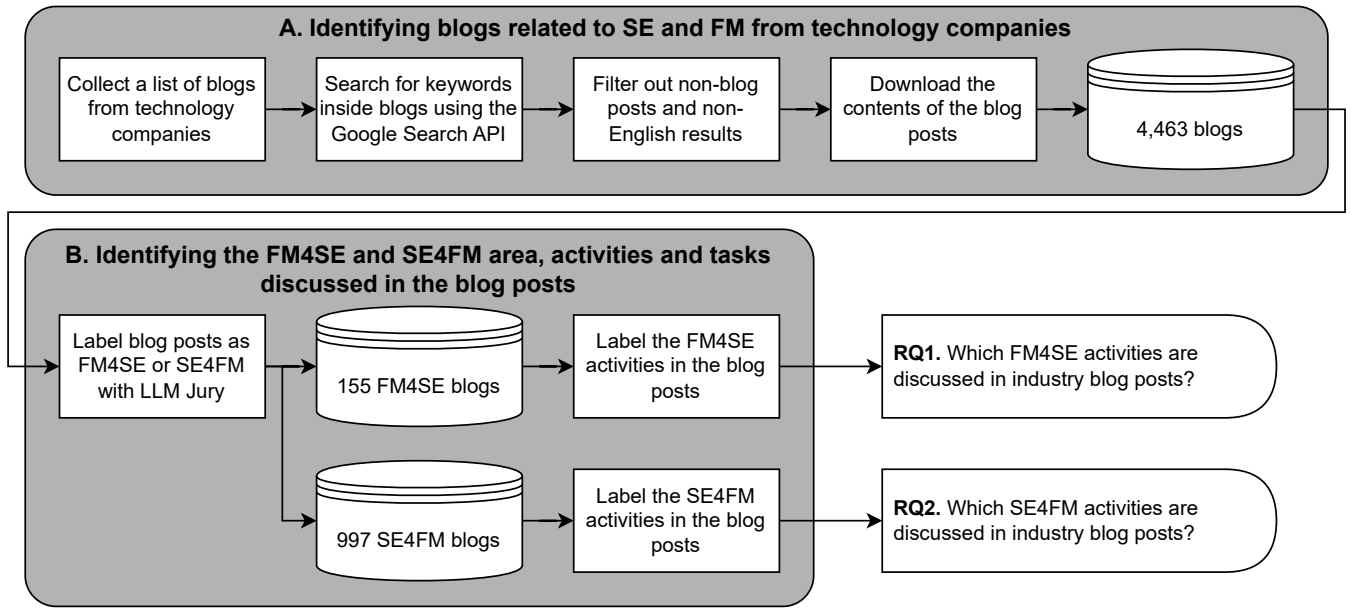


Fig. 2. An overview of our methodology.

our FM/LLM Jury, we begin by constructing a golden dataset. We randomly sample a subset of blog posts and manually label them to serve as ground truth.

Step 2 – Design the prompt. We design prompts following best practices in prompt engineering [35] and techniques outlined by Liu et al. [26]. The prompt should instruct the FMs on how to label the blog posts, specifying the labelling criteria and providing the necessary context. To improve labelling accuracy, we incorporate advanced techniques such as Chain-of-Thought prompting [46] and few-shot in-context learning [3]. We used a predefined set of labels to ensure a common vocabulary for the classification. Because FMs generate natural language text, a common vocabulary is necessary to (1) group many different blog posts that may use different terms to describe the same aspects and (2) facilitate a comparison with prior work which also uses that vocabulary. We also ask FMs to provide new labels if the predefined ones do not fit the content. Our replication package [23] includes all used prompts. We encourage researchers to refine and rerun the process with new blog posts to ensure relevance in this rapidly moving field.

Step 3 – Run the prompt on the golden dataset. Each FM in the jury is prompted to label the blog posts in the golden dataset. For each blog post, the FM outputs both a label and an associated confidence score. As FMs may exhibit overconfidence, directly using the raw confidence scores is likely to introduce bias [25]. While calibration methods exist to address this issue, they typically require access to the model’s internal information or fine-tuning [31], which is not feasible with closed-source FMs. Therefore, we apply a z-score standardization based on the confidence score distribution across the dataset to normalize the confidence values.

Step 4 – Compare FM labels with human labels. The

produced labels are compared with the human-provided labels from the golden dataset. We assess inter-rater reliability using Cohen’s κ coefficient [11], which measures the degree of agreement between the FM-generated labels and human labels. We set a threshold of $\kappa > 0.78$ (indicating excellent agreement) for at least one FM in the jury. Additionally, all FMs must achieve $\kappa \geq 0.63$ (indicating substantial agreement). If these thresholds are not met, we return to Step 2 and refine the prompt iteratively. The refinement process involves adding clarifications, improving instructions, or reordering prompt components to resolve ambiguities [14]. The iterative loop continues until the desired agreement is reached.

Step 5 – Freeze the prompt for full dataset labelling. Once the prompt achieves the required level of agreement in Step 4, it is finalized and used to label the entire dataset.

B. Merging the FM outputs

After the individual FMs in the jury provide their labels, we aggregate the results using a majority vote, where the final label is determined by the label that receives the most votes from the individual FMs. In case of a tie, we use the normalized confidence scores to break the tie.

C. Human-in-the-loop

A human-in-the-loop process was employed to decide whether to accept or reject new labels proposed by FMs. However, the process is only required when the FM/LLM Jury cannot resolve the final label. In our study, the FM/LLM Jury successfully handled all cases without the need for human involvement in the labelling process.

D. Selected FMs

The jury FMs are selected based on their performance and ability to follow complex instructions. We used the LLM

TABLE I
COLLECTED BLOGS WITH AT LEAST ONE RELEVANT POST.

Company	Blog
AMD	https://community.amd.com/t5/ai/
Adobe	https://blog.developer.adobe.com/ https://blog.adobe.com/
Alibaba	https://www.alibabacloud.com/blog/
Amazon	https://www.amazon.science/blog https://aws.amazon.com/blogs/
Cisco	https://blogs.cisco.com/ https://blog.talosintelligence.com/
Google	https://developers.googleblog.com/en/ https://blog.google/ https://cloud.google.com/blog/ https://research.google/blog/ https://deepmind.google/discover/blog/
IBM	https://research.ibm.com/blog https://developer.ibm.com/blogs/ https://www.ibm.com/blog/
Meta	https://tech.facebook.com/engineering/ https://research.facebook.com/blog/ https://engineering.fb.com/
Microsoft	https://www.microsoft.com/en-us/research/blog/ https://blogs.microsoft.com/ https://devblogs.microsoft.com/ https://techcommunity.microsoft.com/t5/
Nvidia	https://blogs.nvidia.com/ https://developer.nvidia.com/blog/
Oracle	https://blogs.oracle.com/
Qualcomm	https://www.qualcomm.com/developer/blog
SAP	https://community.sap.com/t5/technology-blogs-by-sap
Salesforce	https://www.salesforce.com/blog/ https://engineering.salesforce.com/blog/ https://blog.salesforceairesearch.com/

Arena Leaderboard [8] to identify strong candidates. We selected the open-source Qwen2-72B-Instruct [47] model, and two closed-source models, GPT-4o-mini [34] and Gemini-1.5-Flash [41]. These models were selected due to the balance between cost and performance. We did not select the top-performing models, as they are around ten times as expensive.

IV. METHODOLOGY

Figure 2 gives an overview of our methodology. In this section, we discuss every step.

A. Identifying blogs related to SE and FM from technology companies

We employed a systematic approach using search queries based on keywords related to FM4SE and SE4FM to gather blog posts. The data collection process consists of these steps:

Step 1 – Collect a list of blogs from technology companies. We began by collecting a list of companies from the “Technology” sector with a market capitalization greater than \$200 billion (“Mega”) based on data from NASDAQ.¹ In addition, we manually went through the top 100 companies from Forbes’ Global 2,000 list [33] and included companies that are categorized under the “IT Software & Services” industry. In total, we included 20 companies. For each of these companies, we searched for relevant blogs using this query:

¹<https://www.nasdaq.com>

TABLE II
COHEN’S KAPPA BETWEEN LLMs AND HUMAN LABELS ON THE GOLDEN SET FOR SE-FM AREA, FM4SE ACTIVITY, AND SE4FM ACTIVITY CLASSIFICATION TASKS.

Model	SE-FM Area	FM4SE Activities	SE4FM Activities
Gemini-1.5-Flash-002	0.64	0.81	0.69
GPT-4o-mini-2024-07-18	0.70	0.81	0.77
Qwen2-72B-Instruct	0.85	0.76	0.73
FM/LLM Jury	0.92	0.91	0.86

blog AND (“software” OR “research” OR “engineer” OR “engineering”)

We identified 35 blogs from 17 companies (see Table I). This list is easily updated to include new companies/blogs.

Step 2 – Search for keywords inside blogs using the Google Search API. We adapted the keywords used in prior work [48] to suit the FM4SE and SE4FM contexts:

“software” AND (“large language model” OR “large language models” OR “LLM” OR “LLMs” OR “foundation model” OR “foundation models” OR “FM” OR “FMs” OR “generative AI” OR “GenAI”) site:{url}

Using these keywords, we queried the Google Search API across all the blogs identified in Step 1, limiting the date range from August 10, 2023, to August 10, 2024. This process yielded 7,120 search results, including information about each result’s URL, title, and snippet. We assigned a unique identifier (ID) to each search result, ranging from 0 to 7,120.

Step 3 – Filter out non-blog posts and non-English results. The search results might contain unrelated pages such as index pages and author information pages. We filtered out URLs that contain strings such as “/index/” and “/author/” to eliminate non-blog posts. We used the *langdetect* library² to remove results where the title or snippet was not in English.

Step 4 – Download the contents of the blog posts. We downloaded all the content of all valid links. To further remove potential noise, we applied an outlier filter based on the interquartile range (IQR) to exclude content that was either too short or too long. After filtering, we retained 4,463 blog posts, each referenced by a unique ID from Step 2. Throughout this paper, individual blog posts are cited using their corresponding IDs (e.g., [122] for blog post 122), allowing for direct reference to specific entries in the dataset. The full list of blog posts can be found in our online replication package [23].

B. Identifying the FM4SE and SE4FM area, activities and discussed tasks in the blog posts

We first label blog posts as FM4SE or SE4FM-related. Second, we label the FM4SE or SE4FM activities discussed in the blog posts. Finally, we summarize the activity-specific tasks that are discussed, to facilitate our manual review of the posts. Tables III and IV show the identified activities

²<https://github.com/Mimino666/langdetect>

and tasks. The first author, a PhD graduate with 5 years of SE research experience, conducted all labelling and validated through discussions with co-authors.

Step 1 – Label blog posts as FM4SE or SE4FM with FM/LLM Jury. We used the FM/LLM Jury framework (Section III) to classify blog posts as SE4FM, FM4SE, or unrelated. A golden dataset of 100 blog posts was randomly selected to evaluate this classification. Table II shows that the framework achieved an excellent agreement ($\kappa = 0.92$) with human labels on the test set. We then applied the framework to the entire dataset of 4,463 blog posts, classifying 3,126 as unrelated, 156 as FM4SE and 1,122 as SE4FM.

Step 2 – Label the FM4SE activities in the blog posts. We used the FM4SE activity labels from prior work [17] and a golden set of 30 FM4SE posts was randomly sampled for evaluation. Following our iterative prompt construction process (Section III), we reached an excellent agreement ($\kappa = 0.91$) between the FM/LLM Jury and the human labels (see Table II). We applied the Jury to label all FM4SE activities in the 156 FM4SE blog posts. One blog post was labelled as ‘Other’ as the FM/LLM Jury could not identify any specific FM4SE activities, ending up with 155 FM4SE blog posts in total.

Step 3 – Label the SE4FM activities in the blog posts. Following a similar process, we labelled the SE4FM activities in the 1,122 SE4FM blogs. We created a list of SE4FM activity labels based on prior work [1], [15], [28]. We randomly sampled a golden set of 30 SE4FM blog posts and manually labelled the SE4FM activities. Following our prompt construction process, the FM/LLM Jury demonstrated strong agreement ($\kappa = 0.86$) with the human labels. We used the best-performing prompt to label the SE4FM activities in all 1,122 SE4FM blog posts. 125 blog posts were labelled as ‘Other’ as the FM/LLM Jury could not identify any SE4FM activities, ending up with 997 FM4SE blog posts in total.

Step 4 – Identify the activity-specific tasks in the blog posts. To facilitate our manual review of the blog posts, we prompted the FMs to take an additional step to identify the activity-specific tasks in the posts. For FM4SE blogs, we prompted the FMs to identify relevant tasks from a curated task set [12], [17], [24]. Similarly, for SE4FM blogs, the FMs identified SE4FM-related tasks from a curated task set [1], [15], [28]. If no predefined tasks were relevant, we asked the FMs to generate new tags based on the blog content. In cases where the FMs proposed new task tags, we employed a human-in-the-loop process to determine whether to reclassify or ignore these tags. However, no human intervention was required for FM4SE posts, and eight new SE4FM task tags identified by the FM Gemini-1.5-Flash were resolved by the FM/LLM Jury.

V. RQ1: WHICH FM4SE ACTIVITIES ARE DISCUSSED IN INDUSTRY BLOG POSTS?

Motivation. Industry practitioners are at the forefront of applying FMs to SE, sharing practical insights and real-world experiences through blogs. While academic research has explored many aspects of FM4SE, the industry’s perspective remains underexplored. This study analyzes industry blogs to uncover

TABLE III
FM4SE ACTIVITIES AND TASKS THAT ARE DISCUSSED IN INDUSTRY BLOG POSTS. THE COMP. COLUMN INDICATES THE NUMBER OF COMPANIES PUBLISHING BLOG POSTS ABOUT THE TASK.

Activity	Posts	Task	Comp.
Software development	101	Code generation	11
	54	Code completion	8
	12	Code assistant	5
	10	Code understanding	5
	6	Code summarization	4
	3	Code optimization	3
	1	API recommendation	1
# Total	121		11
Software quality assurance	10	Vulnerability detection	3
	5	Debugging	3
	4	Test generation/automation	2
# Total	17		5
Software maintenance	4	Code refactoring or revision	3
	7	Code transformation	2
	3	Code translation	2
	2	Code review	1
	1	Program repair	1
	1	Software operations	1
	1	Log analysis	1
# Total	15		4
Software management	1	Software tool configuration	1
Requirement engineering	1	Requirements analysis	1
Software design	0	–	0
# Total	155		11

key FM4SE activities and tasks discussed by practitioners, providing insights into real-world applications of FMs in SE.

Approach. We used the FM/LLM Jury to label the FM4SE activities and tasks in the 155 FM4SE blog posts. To avoid overrepresentation, we counted each activity and task uniquely per company, even if they were mentioned in multiple blog posts from the same company. To gain insights into how these activities and tasks are discussed, we manually reviewed the selected blog posts. We also compared our findings with those reported by Hou et al. [17].

A. Software development

Although code generation is the most prominent task, FMs are used for many other tasks in the software development process. As shown in Table III, code generation emerges as the most prominent task. Practitioners report leveraging FMs to generate code in modern languages [B140] such as Python and Java, but there is also growing attention for legacy systems, with FMs being used to generate COBOL code [B830]. Additionally, FMs are applied to specialized domains such as SQL query generation [B183], and domain-specific languages (DSLs) tailored to industry-specific needs such as semiconductor design [B8]. The flexibility of FMs to adapt across various programming languages and domains highlights their versatility in software development.

A topic that is closely related to, and often discussed together with code generation is *code completion*. A notable technique is *fill-in-the-middle* [13] for code completion, which allows models to complete code based on partial inputs [B157], further expanding the usability of FMs in practical coding environments.

Beyond code generation and completion, FMs are increasingly being integrated into software development as code assistants, offering a range of functionalities. These assistants not only generate and complete code but also help with *code understanding*, *code summarization*, *code optimization*, and *API recommendations*. For instance, code assistants can understand code and explain it to developers [B5668], summarize code changes for reviews [B322], optimize code for performance [B3357], or recommend APIs based on both public and private code repositories [B3177]. This multifaceted support streamlines the development process, boosting developer productivity and efficiency.

B. Software quality assurance

Vulnerability detection is the most frequently discussed software quality assurance (QA) task. Practitioners employ FMs to automate *common vulnerabilities and exposures (CVE) detection* and analysis [B101]. Other tasks under this category include *test generation and automation*, where FMs are used to generate test cases based on the functionality of a given code. For instance, the FM can suggest test cases for invalid inputs, edge cases, and error handling [B5518]. For *debugging* tasks, FMs can suggest where to insert logging and exception handling to track code execution and errors, and FMs can also be used to detect anomalies and fix common issues such as syntax and logical errors [B5465].

C. Software maintenance

The use of FMs in software maintenance focuses on the refactoring, translation, and transformation of existing codebases. Practitioners often discuss these tasks in the context of modernizing legacy systems. For instance, FMs are employed to *refactor* and *translate* legacy COBOL code into Java [B769]. In addition, upgrading Java applications to newer Long-Term Support (LTS) versions [B369] or migrating Java codebases to cloud-based infrastructures [B6378] are tasks frequently associated with *code transformation*. Using FMs for these tasks helps industries transition from older systems to modern architectures more efficiently.

D. Software management, requirement engineering, and software design

Software management receives the least attention in FM4SE blogs, and no discussions were found regarding requirements engineering or software design. Practitioners use FMs to generate *software tool configurations* for managing cloud infrastructure components [B5625]. We did not find discussions on *requirements engineering* or *software design* in FM4SE blogs. One blog, which was initially misclassified as covering requirements analysis [B1182], was actually about

using FMs to extract developer intent from code comments or function documentation for generating formal postconditions. Overall, FM-based requirements engineering and software design remain an underreported area in the industry.

E. Comparison with Academic Research

Software development is the most discussed activity in both industry blog posts and SE research papers [17]. Both practitioners and researchers frequently highlight code generation as a key task. However, there are some notable differences. Regarding software maintenance, for example, industry blogs focus more on code refactoring and revision, while academic research papers focus more on program repair. Additionally, while Hou et al. [17] reported that 4.3% of surveyed papers (17 out of 395) cover requirements engineering, we found no substantial discussion on this topic in the industry blogs we analyzed (except the one misclassified).

RQ1 Summary: Software development tasks, particularly code generation, are the most frequently discussed across FM4SE blogs. FMs are increasingly integrated as code assistants, providing developers with multifunctional tools to boost productivity. In the domain of software quality assurance, vulnerability detection is the dominant task, while software maintenance activities are primarily focused on refactoring and transforming existing codebases. Software management, requirements engineering, and software design receive less attention in these industry blogs.

VI. RQ2: WHICH SE4FM ACTIVITIES ARE DISCUSSED IN INDUSTRY BLOG POSTS?

Motivation. As FMs are integrated into production systems, applying SE principles to the development cycle of FM-based systems becomes increasingly important. While academic research studied SE for AI-based systems [28], FM-based systems present unique challenges [15], such as the resource-intensive nature of FMs, and the complexities involved in fine-tuning, deployment, and monitoring at scale. Industry blog posts offer valuable insights into how practitioners adapt SE principles for developing, deploying, managing, and scaling FMs in real-world settings. This study gives an overview of key SE4FM activities and tasks discussed in these blog posts.

Approach. We used the FM/LLM Jury to label the SE4FM activities and tasks in the 997 SE4FM blog posts. Similar to Section V, we counted each activity and task uniquely per company and manually reviewed selected blog posts for each task. We did not compare the discussion frequency with previous research like we did in Section V, because there exists no prior survey on SE4FM, and SE4FM activities and tasks are quite different from those for SE4ML [28].

A. Model deployment & operation

Model deployment & operation is the most frequently discussed activity in SE4FM industry blog posts. *Model*

deployment on cloud is the dominant task (see Table IV), reflecting the industry’s reliance on cloud environments for hosting foundation models. Foundation models are very resource intensive. For example, a large model such as Meta Llama 3 (with 405 billion parameters [9]) [B1749, B5439, B6691] requires ~810GB of GPU VRAM for inference, ~3.25TB for fine-tuning [40] and much more for training from scratch. The cloud facilitates using such large models on-demand without the need for buying very expensive hardware.

For *model serving & scaling* [B111, B627, B6028], techniques such as speculative decoding [7] accelerate model inference by using draft models for faster response times [B5308]. In addition, automatic model scaling ensures that resources automatically adjust to workload needs [B701, B4518]. *Model monitoring* [B6659] is another key task, involving tracking token usage [B406, B920] and monitoring system metrics such as memory and GPU load [B5192].

While cloud deployment dominates, there is increasing interest in model deployment on local devices such as edge or mobile devices, and PCs. For example, practitioners deploy FM-based medical chatbots for healthcare applications on edge devices, to facilitate data privacy [B139]. Another reason to deploy FMs on smaller devices is to overcome the GPU supply-and-demand problem, which makes it hard for many companies to integrate FMs into their products [B1355]. To enable running the resource-intensive FMs on relatively small devices, companies frequently use *model compression* techniques [51] to reduce the required resources [B1744, B1993, B5463, B5714]. For example, quantization techniques (e.g., 4-bit precision) enable running models on CPUs, avoiding the need for GPUs [B669]. Compression techniques are not limited to text models: model quantization is also applied to image generative models such as Stable Diffusion [39] [B2254]. Several libraries are used by practitioners that assist with running FMs on CPU, such as *LLaMA.cpp*³ and *ExLlama*⁴ [B1355]. Also, several practitioners describe how the use of Neural Processing Units (NPUs) can facilitate running FMs locally [B914, B1750].

B. System architecture & orchestration

System architecture & orchestration activities, including building AI agents and model & prompt chaining, have become a popular topic in SE4FM industry blog posts. One of the most frequently discussed tasks is *model & prompt chaining*, which is used to manage complex workflows by breaking them into smaller, more manageable steps, each handled by different models or prompts [B881]. For example, a task such as responding to customer reviews might be divided into steps like filtering harmful content, performing sentiment analysis, and generating an appropriate response [B4881]. Tools and frameworks like *LangChain* [6] and *PromptFlow* [30] are commonly mentioned as practical solutions for implementing these chained workflows [B1357].

³<https://github.com/ggerganov/llama.cpp>

⁴<https://github.com/turboderp/exllama>

TABLE IV
SE4FM ACTIVITIES AND TASKS THAT ARE DISCUSSED IN INDUSTRY BLOG POSTS. THE COMP. COLUMN INDICATES THE NUMBER OF COMPANIES PUBLISHING BLOG POSTS ABOUT THE TASK.

Activity	Posts	Task	Comp.
Model deployment & operation	237	Model deployment on cloud	13
	168	Model serving & scaling	12
	30	Model monitoring	8
	31	Model compression	7
	39	Model deployment on local	6
# Total	373		13
System architecture & orchestration	94	Model & prompt chaining	11
	108	Workflow orchestration	10
	83	Building AI agents	10
	99	Development platform & studio	9
	6	Implementing guardrails	3
# Total	287		12
Data management	90	RAG integration	11
	48	Specialized databases	8
	40	Dataset cleaning & preparation	8
	43	Dataset collection	6
	7	Feature engineering	5
3	Dataset labelling	2	
# Total	182		11
Model customization	85	General fine-tuning	10
	20	LoRA	4
	10	RLHF	4
# Total	104		11
Evaluation & quality assurance	17	Model evaluation	6
	15	Model safety & compliance	6
	6	Model risk & trust	4
	5	Testing strategies	4
	2	Model fairness & bias	2
1	Model explainability	1	
# Total	40		7
Prompt construction	10	Prompt engineering	5
	4	Automated prompt generation	3
# Total	11		6
Requirements engineering	0	-	0
# Total	997		14

A closely related discussion topic is *building AI agents*, which extend the functionality of FMs by integrating external tools and *orchestrating workflows*. In multi-agent systems, multiple AI agents collaborate using a complex workflow to handle different parts of a task [B1166]. These AI agents can autonomously decide which tools to use, retrieve necessary data, and execute predefined plans based on user input or real-time data [B414]. Another key feature of AI agents is their ability to leverage working memory, allowing them to retain information from previous interactions or external tool outputs, which can be critical for managing long-term tasks [B442].

Enterprise *development platforms & studios* provide support for building FM-based systems based on chaining or AI agents [B414, B1166, B4881]. These platforms, such as *FMArts* [15], simplify the orchestration of workflows [B358],

allowing developers to easily integrate FMs with external systems. These platforms also support *implementing guardrails*. Guardrails can take the form of filters that are applied to user inputs or LLM outputs [B4677], or can be embedded in the prompts to guide the model’s responses [B2456].

C. Data management

Data management has evolved in the FM era, with new techniques supporting the vast amounts of both structured and unstructured data. At the core of this evolution is the shift to specialized, more dynamic data management techniques. The most frequently discussed task is *Retrieval-Augmented Generation (RAG)* [22], which combines the use of private datasets (i.e., data on which the FM was not trained before such as proprietary data) with FMs. With RAG, the FM generates its response based on the prompt and information in the private dataset [B4270]. Several practitioners discuss how they build datasets for RAG [B6492, B6688] using techniques including document chunking, embedding, and vector storage. Another example of usage of RAG by practitioners is GraphRAG [10] which enhances RAG by generating knowledge graphs from the private data which can then be used for prompt augmentation [B1155, B1156].

There is a great amount of discussion of specialized databases, particularly vector databases. Specialized databases are key to enabling RAG. These databases support semantic search by indexing unstructured data like text and images, making FM-based retrieval faster and more accurate [B1571]. Advanced features include multimodal search, where users retrieve image or video content using text queries [B5035]. This shift from traditional keyword-based search to semantic search also integrates with SQL queries for managing both structured and unstructured data [B1426]. As data management moves beyond traditional data types (e.g., rows, columns, JSON), vector-based storage and retrieval systems are becoming more important [B5370]. Likewise, *embedding as feature engineering* is becoming increasingly important in FM-based systems, enabling text, images, and structured data to be converted into numerical vectors that FMs can process [B1982]. Multimodal embeddings, which map both text and images into a shared vector space, are particularly useful in cross-modal applications such as text-to-image search or video retrieval [B5815].

Synthetic data generation provides scalable, domain-specific data without privacy risks, reducing reliance on real-world datasets. As FM data requirements grow, synthetic data is increasingly used to address the challenges of high-quality *data collection* [B35, B4210]. In parallel, *automated data labelling* is being transformed by model-assisted approaches. For example, the Recognize Anything Model (RAM) [49] can automatically label visual datasets, enabling users to search for images or videos using natural language queries [B3649]. Additionally, human-in-the-loop is applied for combining model-generated annotations with manual oversight to ensure accuracy while reducing the time and cost associated with traditional labelling methods [B5181].

There is a noticeable push on data privacy in data cleaning & preparation. *Data cleaning & preparation* remains essential, but data privacy has become an important focus. Since the data used for customizing FMs could contain Personally Identifiable Information (PII), data anonymization techniques such as differential privacy [B5934] are used to remove PII [B1286, B1309, B3213]. Beside data privacy, removing duplicate is frequently applied to preprocess the dataset to ensure data quality [B188, B804, B2326].

D. Model customization

Model customization is achieved through fine-tuning techniques for adapting FMs to specific application needs.

Fine-tuning methods such as supervised fine-tuning (SFT) tune the entire model on domain-specific data, rather than train it from scratch. Enterprise platforms now support no-code fine-tuning, simplifying the process and accelerating development [B4777]. Open source libraries such as Hugging Face’s PEFT [27] support both full fine-tuning and *Low-Rank Adaptation (LoRA)* [B353]. LoRA is an efficient approach where original model parameters are frozen and injected with trainable matrices. LoRA reduces the number of trainable parameters and lowers GPU requirements, making it cost-effective [B109]. With different LoRA adapters, a single FM can adapt to handle different tasks. Platforms support dynamic loading and caching of LoRA adapters, offering flexibility and optimizing performance [B2462]. *RLHF* is used to align models with user preferences to improve their experience [B656] and can also be applied to image models [B2367].

E. Evaluation & quality assurance

SE4FM blog posts outline practical strategies for ensuring the safety, fairness, and trustworthiness of FMs through systematic evaluation & QA processes. With the diverse applications of FMs, establishing robust *model evaluation* frameworks is essential to ensure models meet operational requirements [B898]. For example, for FM-based systems with RAG integration, an evaluation framework includes metrics such as answer relevance, context precision, and recall to assess the effectiveness of model outputs [B1305]. Ensuring *model safety & compliance* is particularly critical in high-stakes industries. Industry blogs highlight the use of adversarial testing to identify model vulnerabilities, while automated raters are often deployed to perform consistent safety assessments [B437]. Standardized benchmarks are also leveraged to ensure models meet security and compliance standards, especially in regulated industries [B845].

Practitioners are increasingly adopting *automated testing strategies for FMs*, which often use academic benchmarks like BIG-bench [B158]. However, custom datasets tailored to specific domain requirements are also vital for evaluating FMs in domain-specific applications [B845]. One emerging approach in this area is the use of LLM-as-a-judge techniques, which leverage LLMs to provide scalable and consistent evaluations [B158]. Additionally, adversarial testing is used to strengthen models against potential threats by uncovering

weaknesses that may not surface under traditional testing methods [B845]. Another emerging task is *model explainability & interpretability*, which is important particularly in sensitive industries. Tools that generate natural language explanations for model outputs are becoming common, helping developers understand why certain test cases pass or fail [B4776]. This increases transparency and aligns FMs with best practices for software engineering.

Some practitioners discuss *model fairness & bias*, as biased outputs from FMs can lead to ethical concerns or operational risks. To reduce bias in FM-generated outputs, techniques such as prompt engineering and scenario testing are employed [B1295]. Adversarial testing and diverse rater systems are also leveraged to ensure fairness and prevent harmful outputs [B437]. Additionally, practitioners highlight the importance of human oversight in critical decision-making processes to mitigate risks associated with harmful or biased outputs [B1064]. To enhance trust in FM-based systems, practitioners conclude and follow a set of best practices for AI security, such as the Secure AI Framework (SAIF) [B697].

F. Prompt construction

Prompt construction receives the least attention, and no discussions were found regarding requirements engineering. *Prompt engineering* techniques are discussed in the blog posts, such as structured prompts [B374] and multi-shot prompts [B4401]. In database contexts, prompts consider schema, query history, and user-specific factors to generate SQL queries [B3601]. In addition, *automated prompt generation* techniques are explored through dynamic metaprompts which are optimized for greater control and adaptability [B1172]. For tasks like text-to-image generation, prompts are improved based on semantic search and user context [B3754]. In addition, prompt compression is proposed to automatically reduce prompt length without sacrificing essential information [B1173].

RQ2 Summary: The most discussed activities in SE4FM blog posts are model deployment & operation, with a focus on cloud hosting and model serving & scaling. With regards to system architecture, trends include prompt chaining, workflow orchestration for FMs, and AI agents. Data management emphasizes RAG and vector databases to support unstructured data and information retrieval. Model customization relies on fine-tuning methods such as full fine-tuning, LoRA, and RLHF. Evaluation & quality assurance practices focus on automated testing, safety, trustworthiness, and bias mitigation, while prompt engineering receives limited attention. Discussion about requirements engineering is not found in SE4FM blog posts.

VII. DISCUSSION OF FUTURE RESEARCH DIRECTIONS

A. Research Directions for FM4SE

Research Direction 1: Using FMs for modernization and transformation of legacy code. Practitioners applied FMs

to translate legacy systems into modern languages such as Java [B769], upgrade to newer language versions [B369], and migrate to cloud-based infrastructures [B6378]. This process mostly relies on FMs for code translation, however, Pan et al. [37] highlight challenges in applying FMs to translate code in real-world projects. To address these challenges, researchers should explore methodologies that enhance FM performance for automating complex system migrations, including the transformation of entire legacy codebases and architectures.

Research Direction 2: Evaluating code assistants in software development workflows for tasks other than code generation. Liang et al. [24] conducted a survey on the usability of code assistants focusing on the code generation task. However, industry discussions highlight that code assistants are employed for other tasks such as code understanding [B5668], code summarization [B3357], and API recommendation [B3177] as well. This broader integration, including their potential role as AI teammates [16], suggests opportunities for researchers to expand studies on code assistants beyond code generation.

Research Direction 3: Real-world validation of research on applying FMs in software management, requirements engineering, and software design. Our analysis of industry blog posts shows that discussions around FM applications in these activities are rare (see Table III). Likewise, though software engineering researchers have explored using FMs for requirements engineering tasks such as requirements classification and traceability automation, such studies remain relatively rare as well [17]. Considering the capabilities of FMs in handling natural languages and programming languages, they should be well-suited for these tasks, hence the lower number of (reported) applications in this area is surprising. Researchers should aim to bridge the gap by identifying the barriers to and developing tools that apply FM techniques in these underreported activities. Such efforts could help bring research advancements in these areas closer to practical industry applications, showcasing the value of FM4SE for a broader range of software engineering activities.

B. Research Directions for SE4FM

Research Direction 4: Researchers should expand research on SE4FM. Our findings in Sections V and VI show that practitioners discussed SE4FM much more often than FM4SE, with 997 blog posts focused on SE4FM compared to 155 blog posts on FM4SE. Activities such as model deployment & operation, system architecture, data management, and model customization are frequently mentioned by more than half of the companies (Table IV). Although the lower number of posts on FM4SE does not necessarily indicate a lack of interest in these topics, there seems to be a disconnect between academic research and practitioner activities. The results of our study highlight the growing importance of adapting SE practices to support the development lifecycle of FM-based systems. Researchers should explore these areas to bridge the gap between industry practices and academic research, thereby supporting the evolving needs of FM-based systems.

Research Direction 5: Performance engineering for FM-based systems. Performance optimization of FM-based systems [15] is a critical area that practitioners frequently discussed (in several tasks). Techniques such as model compression and model serving & scaling are employed to reduce the computational resources needed for deploying FMs [B701, B5463]. Researchers could contribute by formalizing and standardizing such techniques, as well as exploring new methods to reduce the computational overhead of fine-tuning and deploying FMs. The resource-intensive nature of FMs introduces challenges related to managing large-scale parallelism and ensuring memory efficiency, especially for models with billions of parameters [B5192]. Extending traditional load testing methods [18] to FM inference pipelines could help ensure that these models scale effectively and meet the performance requirements of real-time applications.

Research Direction 6: Investigate the impact of the shift from full model training to fine-tuning on software engineering activities such as dependency and asset management. Industry blog posts highlight the increasing reliance on libraries for model fine-tuning and model inference, such as PEFT from Hugging Face [B353] and LLaMA.cpp [B1355]. Researchers should investigate how these emerging tools and techniques integrate into existing SE workflows and assess their impact on model quality, performance, and usability. For example, the reduced set of trainable parameters used by LoRA can be stored separately as an adapter [B2462], which introduces new challenges in asset management. Researchers should investigate how to manage and optimize these adapters within the context of FM-based systems.

Research Direction 7: Supporting FM workflow orchestration and AI agents through software engineering activities. Practitioners are integrating FMs not as standalone components but as part of complex workflows that chain together multiple models, prompts, and external APIs [B881, B1357]. Furthermore, AI agents that autonomously manage tasks and orchestrate tools are becoming prevalent in industry [B414, B1166]. Researchers should explore design patterns, best practices, and frameworks that support the development of these FM-based systems.

Research Direction 8: Supporting the evolving data pipelines in FM-based systems. While traditional data collection, cleaning, and labelling methods remain essential, industry practices are increasingly incorporating synthetic data generation and automated labelling using FMs [B4210, B5181]. Synthetic data generation is used to produce large volumes of domain-specific data without privacy concerns [B35]. However, as reliance on synthetic and automatically labeled data increases, researchers must investigate the trade-offs between data quality, model performance, and ethical considerations. Human-in-the-loop approaches, where models assist with but do not completely replace manual labelling, also offer a promising area for further exploration [B5181]. Additionally, the growing use of specialized databases, such as vector databases for efficient retrieval of unstructured data [B1571], demands research into how these evolving data management

strategies affect the development lifecycle of FM-based systems, particularly when integrating multimodal data [B5035].

VIII. THREATS TO VALIDITY

Internal Validity. Industry blogs may not fully represent a company’s official or comprehensive views, as posts are often authored by individuals or specific teams. This introduces a risk of bias since not all companies maintain blogs which could potentially skew our dataset. In addition, there could be marketing trends or recent product launches, which could further skew the dataset. To mitigate this threat, we not only report the number of blogs mentioning a specific activity, but also the number of companies discussing those activities in our analysis. Additionally, we realize that any overview of discussed topics in fast moving areas such as FM4SE and SE4FM is only a snapshot of the situation at that time, which can quickly change. We want to emphasize that our FM-powered surveying approach is automated and flexible, and can be rerun easily for different companies, blog posts and possibly even different areas.

The prompt optimization approach in Section IV is currently manual. In future work, we plan to explore automated prompt optimization frameworks, such as `dspy` [19], [20].

External Validity. The FM/LLM Jury comprises models of different sizes. If using smaller models, they might lack emergent abilities [45] found in larger ones which could affect the accuracy of the labels. Furthermore, our findings are based on blogs from large companies and may not generalize to smaller ones which are limited by resource constraints.

IX. CONCLUSION

This study bridges the gap between academic research and industry practices by analyzing 1,152 industry blog posts related to FM4SE and SE4FM from leading technology companies. We used an FM/LLM Jury of multiple FMs to automatically label and summarize the contents. We uncovered key activities and trends discussed by industry practitioners in the intersection of SE and FMs. Our main findings are:

- Although code generation is the most prominently discussed task in FM4SE blog posts, FMs are used for many other tasks in the software development process, including code completion, code understanding, and code summarization. Software engineering researchers should investigate how to leverage FMs to support an even broader range of software engineering activities.
- In SE4FM blog posts, model deployment & operation and system architecture & orchestration are the most frequently discussed activities. Software engineering researchers should investigate how these activities can be further supported through software engineering.

Our findings offer valuable insights into how industry leaders are leveraging FMs and SE. We provide researchers with eight promising research directions to explore. Additionally, we demonstrate the potential of using an FM-powered surveying approach for automating grey literature surveys in rapidly evolving fields like SE and ML.

REFERENCES

- [1] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software Engineering for Machine Learning: A Case Study," in *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019, pp. 291–300.
- [2] A. A. Bangash, H. Sahar, S. Chowdhury, A. W. Wong, A. Hindle, and K. Ali, "What do Developers Know About Machine Learning: A Study of ML Discussions on StackOverflow," in *IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, pp. 260–264.
- [3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.
- [4] G. Butler, *Think write grow: how to become a thought leader and build your business by creating exceptional articles, blogs, speeches, books and more*. John Wiley & Sons, 2012.
- [5] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang *et al.*, "A Survey on Evaluation of Large Language Models," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 3, Mar. 2024.
- [6] H. Chase, "LangChain," <https://github.com/langchain-ai/langchain>, last visited: Oct 11, Oct. 2022.
- [7] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating Large Language Model Decoding with Speculative Sampling," *arXiv preprint arXiv:2302.01318*, 2023.
- [8] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez *et al.*, "Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference," *arXiv preprint arXiv:2403.04132*, 2024.
- [9] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The Llama 3 Herd of Models," *arXiv preprint arXiv:2407.21783*, 2024.
- [10] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, "From local to global: A graph rag approach to query-focused summarization," 2024. [Online]. Available: <https://arxiv.org/abs/2404.16130>
- [11] K. E. Emam, "Benchmarking Kappa: Interrater Agreement in Software Process Assessments," *Empirical Software Engineering*, vol. 4, no. 2, pp. 113–133, Jun 1999.
- [12] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large Language Models for Software Engineering: Survey and Open Problems," in *IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, May 2023.
- [13] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, S. Yih, L. Zettlemoyer, and M. Lewis, "InCoder: A Generative Model for Code Infilling and Synthesis," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.
- [14] Google Cloud, "Prompt iteration strategies," <https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/prompt-iteration>, last visited: Oct 11, 2024.
- [15] A. E. Hassan, D. Lin, G. K. Rajbahadur, K. Gallaba, F. R. Cogo, B. Chen, H. Zhang, K. Thangarajah, G. Oliva, J. J. Lin *et al.*, "Rethinking Software Engineering in the Era of Foundation Models: A Curated Catalogue of Challenges in the Development of Trustworthy FMware," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE)*, 2024, p. 294–305.
- [16] A. E. Hassan, G. A. Oliva, D. Lin, B. Chen, Z. Ming, and Jiang, "Towards AI-Native software engineering (SE 3.0): A vision and a challenge roadmap," *arXiv preprint arXiv:2410.06107*, 2024.
- [17] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large Language Models for Software Engineering: A Systematic Literature Review," *ACM Trans. Softw. Eng. Methodol.*, Sep. 2024.
- [18] Z. M. Jiang and A. E. Hassan, "A Survey on Load Testing of Large-Scale Software Systems," *IEEE Transactions on Software Engineering*, vol. 41, no. 11, pp. 1091–1118, 2015.
- [19] O. Khattab, K. Santhanam, X. L. Li, D. Hall, P. Liang, C. Potts, and M. Zaharia, "Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive NLP," *arXiv preprint arXiv:2212.14024*, 2022.
- [20] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, M. Zaharia, and C. Potts, "Dspy: Compiling declarative language model calls into self-improving pipelines," *arXiv preprint arXiv:2310.03714*, 2023.
- [21] T. Kocmi and C. Federmann, "Large Language Models Are State-of-the-Art Evaluators of Translation Quality," in *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*. European Association for Machine Translation, Jun. 2023, pp. 193–203.
- [22] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates Inc., 2020.
- [23] H. Li, C.-P. Bezemer, and A. E. Hassan, "Replication package," <https://github.com/SAILResearch/fmse-blogs>, 2024.
- [24] J. T. Liang, C. Yang, and B. A. Myers, "A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, 2024.
- [25] S. Lin, J. Hilton, and O. Evans, "Teaching Models to Express Their Uncertainty in Words," *Transactions on Machine Learning Research*, 2022.
- [26] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing," *ACM Comput. Surv.*, vol. 55, no. 9, Jan. 2023.
- [27] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "Peft: State-of-the-art parameter-efficient fine-tuning methods," <https://github.com/huggingface/peft>, last visited: Oct 11, 2022.
- [28] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, and S. Wagner, "Software Engineering for AI-Based Systems: A Survey," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 2, apr 2022.
- [29] S. Masuda, K. Ono, T. Yasue, and N. Hosokawa, "A Survey of Software Quality for Machine Learning Applications," in *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2018, pp. 279–284.
- [30] Microsoft, "Prompt flow," <https://github.com/microsoft/promptflow>, last visited: Oct 11, 2024.
- [31] S. J. Mielke, A. Szlam, E. Dinan, and Y.-L. Boureau, "Reducing Conversational Agents' Overconfidence Through Linguistic Calibration," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 857–872, 2022.
- [32] M. M. Morovati, A. Nikanjam, F. Tambon, F. Khomh, and Z. M. J. Jiang, "Bug characterization in machine learning-based systems," *Empirical Software Engineering*, vol. 29, no. 1, p. 14, Dec 2023.
- [33] A. Murphy and M. Schifrin, "Forbes 2024 global 2000 list," <https://www.forbes.com/lists/global2000/>, last visited: Oct 11, 2024.
- [34] OpenAI, "GPT-4o mini: advancing cost-efficient intelligence," <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence>, last visited: Oct 11, 2024.
- [35] —, "Prompt engineering," <https://platform.openai.com/docs/guides/prompt-engineering>, last visited: Oct 11, 2024.
- [36] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35. Curran Associates, Inc., 2022, pp. 27 730–27 744.
- [37] R. Pan, A. R. Ibrahimzada, R. Krishna, D. Sankar, L. P. Wassi, M. Merler, B. Sobolev, R. Pavuluri, S. Sinha, and R. Jabbarvand, "Lost in Translation: A Study of Bugs Introduced by Large Language Models while Translating Code," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024.
- [38] A. Panickssery, S. R. Bowman, and S. Feng, "LLM Evaluators Recognize and Favor Their Own Generations," *arXiv preprint arXiv:2404.13076*, 2024.
- [39] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," *arXiv preprint arXiv:2112.10752*, 2021.
- [40] P. Schmid, O. Sanseviero, A. Bartolome, L. von Werra, D. Vila, V. Srivastav, M. Sun, and P. Cuenca, "Llama 3.1 – 405B, 70B & 8B with multilinguality and long context," <https://huggingface.co/blog/llama3#training-memory-requirements>, last visited: Oct 9, 2024.

- [41] G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang *et al.*, “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context,” *arXiv preprint arXiv:2403.05530*, 2024.
- [42] P. Verga, S. Hofstatter, S. Althammer, Y. Su, A. Piktus, A. Arkhangorodsky, M. Xu, N. White, and P. Lewis, “Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models,” *arXiv preprint arXiv:2404.18796*, 2024.
- [43] H. Villamizar, T. Escovedo, and M. Kalinowski, “Requirements Engineering for Machine Learning: A Systematic Mapping Study,” in *47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2021, pp. 29–36.
- [44] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, “Software Testing With Large Language Models: Survey, Landscape, and Vision,” *IEEE Transactions on Software Engineering*, vol. 50, no. 4, pp. 911–936, 2024.
- [45] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, “Emergent Abilities of Large Language Models,” *arXiv preprint arXiv:2206.07682*, 2022.
- [46] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language models,” in *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates Inc., 2022.
- [47] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang *et al.*, “Qwen2 Technical Report,” *arXiv preprint arXiv:2407.10671*, 2024.
- [48] P. Yu, H. Xu, X. Hu, and C. Deng, “Leveraging generative AI and large language models: A comprehensive roadmap for healthcare integration,” *Healthcare*, vol. 11, no. 20, 2023.
- [49] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, Y. Qin, T. Luo, Y. Li, S. Liu, Y. Guo, and L. Zhang, “Recognize Anything: A Strong Image Tagging Model,” *arXiv preprint arXiv:2306.03514*, 2023.
- [50] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing *et al.*, “Judging LLM-as-a-judge with MT-bench and Chatbot Arena,” in *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates Inc., 2023.
- [51] X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang, “A Survey on Model Compression for Large Language Models,” *arXiv preprint arXiv:2308.07633*, 2024.