Software Engineering Module Assignment

Kevin Glocker
Linköping University

1 Introduction

Most well known for their use in recent chatbots such as OpenAI's ChatGPT, Google's Gemini, and Anthropic's Claude, large language models (LLMs) have achieved state-of-the-art results for a wide range of tasks in the field of Natural Language Processing and as general-purpose chatbots through scaling to billions of parameters and training on web-scale text corpora. Scaling has led to the emergence of strong zero-shot task-solving abilities. However, LLMs are computationally expensive to employ in practical applications and notoriously difficult to control. Furthermore, a crucial issue inherent in LLM-based chatbots is generating "hallucinations" of factually false or incoherent information [1].

The primary aim of my research is to investigate methods to improve the computational efficiency of LLM training while making models more interpretable and controllable. In particular, the ongoing research projects that I am involved in encompass investigating general language generation across multiple languages, domain-specific representations, and functional capabilities, such as solving knowledge-intensive tasks.

In LLMs, knowledge can be integrated in two forms. Primarily, LLMs learn to encode factual information from large-scale training data stored implicitly in their parameters, known as parametric knowledge. Secondly, LLMs can be combined with a search engine to find relevant external documents to use as context in a process known as retrieval-augmented generation (RAG) [2]. Although RAG substantially improves the factual correctness of LLMs by grounding to given external sources, they still sometimes rely on their internal parametric knowledge and hallucinate [3], [4]. Secondly, while recent LLMs, including open-weight models, have increasingly included more diverse multilingual data in their pretraining mix, the details of the data mix are often proprietary [5], [6]. Most importantly, the true extent of these LLMs' generalization capabilities to other, particularly lower-resource, languages is still poorly understood. This is mainly due to the fact that the majority of reported benchmarks are still largely focused on English and a select few high-resource languages such as Mandarin Chinese.

Currently, I am working on the development of efficient multilingual models that can aid in the understanding of which factual information is present in LLM pretraining data across languages. Furthermore, by extracting rich, interpretable metadata from pretraining data, the development of more reliable methods to integrate knowledge into LLMs and reduce the aforementioned problems with RAG becomes possible. In parallel, I am working on a new, highly multilingual benchmark derived from existing curated data sources to evaluate the model in development and compare the performance of multilingual LLMs on the same task. Moreover, as part of an ongoing project with other researchers in my group, I am investigating how to efficiently adapt smaller language models to multiple languages with varying amounts of available data. The goals are to understand cross-lingual transfer mechanisms better and to further the development of smaller models with strong linguistic capabilities in multiple languages. The latter is particularly beneficial for local deployment on edge devices.

2 Lecture Principles

A foundational principle in the lectures is how validation and verification should be understood in the context of machine learning. Particularly when working with models intended for zero-shot use in a wide variety of scenarios, such as LLMs, emergent model behavior is hard or impossible to fully control and

verify, as in traditional software development. Even in more specialized supervised models, it is usually impossible to fully account for every possible input. While there are some cases where strict grammar constraints are used during LLM inference to enforce, e.g., a verifiable JSON schema, the remaining unconstrained contents of a JSON object, validation can only be performed depending on the input data. Validation in this case can be performed, e.g., using carefully curated benchmarks for downstream tasks.

The second principle that was particularly relevant to my research is how to properly curate a validation set. Especially since I am currently developing a benchmark myself based on a variety of data sources in multiple languages, built on linguistically diverse documents and annotated with different label sets and annotation guidelines, it is important to understand that not every input document and output label can be treated equivalently. While designing the evaluation metrics for my benchmark, I am therefore reporting scores for different related label categories and languages separately to appropriately represent the performance of models in diverse settings. A simple approach to only report aggregated scores across the entire dataset and to consider all labels to be equal would hide this diversity. Furthermore, it would obscure any potential biases of evaluated models to specific clusters of inputs or labels.

3 Guest-Lecture Principles

The first principle from the guest lecture I found particularly relevant is the impact of requirements on a project and how these requirements can be defined, expressed, and enforced. My initial impression of requirement engineering was that these would mainly be conceived in the form of documents that are designed upfront. Since I do not currently collaborate with industry, I am not bound to requirements from specific, external stakeholders, such as customers of a software product. Most larger NLP research projects are dynamic and evolve rapidly. Perhaps even more so than in a "traditional" software project, a full requirements document for an ongoing research direction or its associated software would be work-intensive initially, while also requiring constant revision. However, a key point I took away from both the first guest lecture and the preceding lecture is that since full specifications are unfeasible, requirements could be better expressed in other ways. For instance, the requirements for our research software could be represented as integration or system tests that ensure certain invariants hold.

This leads directly into a second principle I took away from the lecture. Once a test-driven approach to enforce requirements is chosen, it is important to elicit the exact requirements that need to be enforced. This requires both decomposing a wider goal into specific and measurable requirements and identifying cases that need to be avoided. More concretely, in our projects, properties of text data processing pipelines can be tested for. This is particularly important for the benchmark I am developing. In this case, it is key to both verify the correctness of evaluation metrics and ensure that the evaluation data is internally consistent. These specific cases are measurable requirements derived from the general goal of developing a reliable and consistent benchmark. In the case of, e.g., LLM finetuning experiments, it can be validated whether training recipes remain reproducible. In the case of larger-scale LLM pretraining, however, it is only feasible to validate it partially due to resource constraints.

Connecting the two principles, a test-centric approach to requirements is more practical, as continuously ensuring our software implementations fulfill our requirements during an ongoing project can be as easy as running the test suite. After eliciting specific requirements from our project goals, no additional time-intensive requirement document writing is necessary.

4 Data Scientists versus Software Engineers

The textbook starts off mainly characterizing data scientists as working within often fragile Jupyter notebooks and only focusing on model development, going as far as calling individuals with both broad data science and software engineering expertise "unicorns" [7]. Although this is indeed a common workflow for data scientists, at least anecdotally speaking, I have personally met many capable data scientists with strong software engineering backgrounds. Therefore, as the author admits [7], I find this definition oversimplified and only partially agree with it. While even highly specialized data scientists

require at least minimal software engineering knowledge, familiarity with data science is not necessary for software engineers working in unrelated fields that have not traditionally involved machine learning. The textbook defines them by their expertise in balancing different constraints to complete a software project while ensuring usability, maintainability, and robustness [7]. This definition generally matches my own impression of software engineering in industry, which is why I agree with the described contrast with data scientists, for which such skillsets are usually not the main priority.

In my opinion, due to the breadth of data science as a field, the extent to which a deeper understanding of software engineering principles by data scientists is required also depends on the field, scale of operations, and academic or business environment. My opinion in this case is mainly motivated by my perspective of academia, where less funding is available for dedicated, specialized staff, which may also apply to some startups. As such, not all environments can benefit from the strong interdisciplinary collaboration the textbook advocates for [7]. As state-of-the-art machine learning models grow in size and complexity, both in terms of the number of model parameters and data samples, data scientists working on more foundational models may be increasingly required to write more complex and performance-critical software to be able to conduct their work within their available resources. For instance, when working on fundamental improvements of auto-regressive language models with even a small fraction of the parameter count of state-of-the-art LLMs, training on sufficiently large datasets can cost hundreds to thousands of high-end GPU hours. This requires applying robust software engineering practices to ensure the correctness and reliability of the training code, even against potential compute node failures, verified through thorough testing. Furthermore, when modifications are desired that go beyond the most commonly used transformer architectures implemented in popular libraries, an in-depth understanding of software engineering is required to optimize performance-critical code. Even if data scientists do not perform such optimizations themselves, at least an understanding of how a new machine learning solution will perform in practice is required. Here, I agree with the importance of understanding the entire machine learning system as outlined in the textbook chapters [7].

On the software engineering side, I do not think the field will somehow fully merge with data science in the near future. This is particularly the case in low-level engineering for embedded devices on which even simpler machine learning models can not be deployed and where switching to more powerful chips would not be worthwhile. In contrast, as tightly integrating machine learning solutions into software products becomes more popular, I think the software engineering subfield with a focus on a deeper understanding of machine learning and its impacts will continue to grow.

5 Paper analysis

5.1 Privacy and Copyright Protection in Generative AI

The paper titled "Privacy and Copyright Protection in Generative AI: A Lifecycle Perspective" [8] highlights current legal and ethical challenges throughout the entire data lifecycle for generative AI models from the perspective of the GDPR and US copyright law. The authors advocate for a holistic view of data from the initial problem formulation and data collection to model inference and distribution. Understanding the requirements of the GDPR and copyright law is critical for the engineering of AI systems to ensure regulatory compliance and prevent legal action. Due to the complexity of handling data from collection and storage to becoming encoded within the parameters of a generative model, compliance can not simply be an afterthought. Instead, it must be considered from the initial definition of project requirements throughout the software development process, and during the entire lifecycle in production. This is particularly the case when data samples become difficult to track in highly data-intensive applications, such as the generative models the paper focuses on [8].

Since my research involves both pretraining and fine-tuning of language models on large amounts of text data, it is important to consider the provenance of my data when used during model training or even directly redistributed as part of a benchmark. Our research only involves publicly available and permissively licensed datasets. However, the exact legal status of any web-scale dataset we use, such as

FineWeb-2 [9], is unclear and difficult to verify due to its size. As a result, the paper is relevant to my work both in showing where throughout the lifecycle of data usage legal problems may appear and how those could be mitigated.

For instance, the paper highlights current attempts by major LLM offerings to prevent certain privacy-sensitive or copyrighted information from being output by extensive post-training of the LLM itself or dedicated external models [8]. However, these methods are often insufficient and can be circumvented by users with creative prompts [8]. While the authors suggest continuously monitoring models and updating preventative methods, this would likely continue in an arms race of prevention methods and prompt hacking methods to circumvent them. As such, it could ultimately not solve the underlying issue. Furthermore, the paper covers the issue of appropriate license tracking from data collection through to model training [8]. In particular, it highlights that even if all best practices are followed, new data consent tracking methods are implemented, and licenses are properly tracked for each data sample, it is difficult or impossible to trace back an output of the final model to a particular data sample. This is particularly problematic when data owners make use of the GDPR Right of Erasure and request deletion of their data since a trained model could potentially still generate it given the right prompt [8]. Machine unlearning techniques have been proposed for this scenario to avoid retraining that is too costly to be practically feasible in many cases. However, such methods have not yet proven effective for large-scale models [8].

Motivated by these key issues, a concrete project in which the paper's ideas could be applied is a general-purpose European chatbot. In this hypothetical project, the chatbot would consist of an LLM that is pretrained on all major languages spoken within Europe. It could additionally offer services to companies to fine-tune and serve models for them using their own proprietary data. For the pretraining process, there are multiple possible ways in which the copyright and GDPR concerns outlined in the paper could be handled. First, data collection could be performed by carefully considering licensing of public text data and collecting appropriate consent from copyright holders as appropriate. Closely following recommendations from the paper, appropriate measures for consent tracking of data samples could be followed to keep track of data licensing and ownership throughout the data preparation and model training process [8].

Finally, a solution needs to be developed for handling GDPR data deletion requests. As an alternative to either costly retraining or complex unlearning techniques, new modular architectures could be developed in which potentially sensitive data is not directly included in the pretraining data of a monolithic model, but rather distributed across smaller dedicated submodules that are substantially cheaper to retrain. Alternatively, such data could be entirely held out of the training process and only retrieved from a database as needed in a RAG-like setup. My research in understanding the factual information present in LLM pretraining data could fit into this project by aiding prefiltering of only the most relevant data to maintain diversity and quality, while reducing the amount of training data that needs to be tracked. Additionally, findings from my research could be key to developing the alternative, modular LLM architecture I briefly described.

Implementing a comprehensive consent tagging scheme for large-scale data, as suggested in the paper [8], is currently out of scope for my PhD research and would likely require more systematic efforts throughout our field. However, I can concretely apply measures to prioritize data sampled from sources that are less likely to contain sensitive data. Furthermore, I will follow best practices to keep track of the data I am using and its provenance throughout my data processing, training, and evaluation pipelines. Finally, if the scope of my research project allows, architectures for more reliable and efficient handling of GDPR deletion requests after model training would be an interesting direction.

5.2 RAGProbe: Breaking RAG Pipelines with Evaluation Scenarios

RAGProbe [4] proposes evaluation scenarios for complete LLM RAG pipelines across different applications and use cases. A scenario consists of specific configurations of the RAG pipeline. The options include chunking and sampling methods of the documents that the knowledge base consists of, and the desired evaluation metrics, such as correctness and consistency of responses. Most importantly, RAGProbe generates synthetic question-answer pairs for evaluation using only sampled documents. This approach

requires no hand-crafted example data and only a scenario-specific prompt for a specific use case. Results show that this approach is more effective at identifying failure cases in RAG than the current state-of-the-art approach, RAGAS [10]. For example, one of the evaluated scenarios tests whether an LLM correctly refuses to answer a question when no relevant context is required instead of responding regardless based on its parametric knowledge [4]. This scenario is particularly relevant, since the responses could either be entirely hallucinated or based on outdated or unrelated information.

Evaluation of RAG behavior is relevant to all AI systems that integrate LLMs with either internal or external data sources, such as most standalone or integrated chatbots intended for question answering. For example, customer service or technical support bots that require access to internal knowledge bases, or programming assistants that index code bases and library documentation.

While my own research is more foundational and focused on LLMs themselves and their data rather than full downstream applications, evaluation as part of a RAG pipeline can yield a more comprehensive picture of my work's utility. The proposed RAGProbe method, together with other RAG evaluation methods, such as RAGAS [10], could add valuable insights to the wider impact of, e.g., architectural and data quality improvements, and provide new opportunities for more in-depth analysis of the interaction between LLMs and RAG components.

An example project in which RAGProbe could be applied is an LLM-powered assistant that aids in the migration of legacy software projects to a new platform, such as a cloud provider. The LLM could be used to analyze the legacy code base and provide guidance in understanding its architecture and any components that could be challenging to migrate. This could involve answering questions related to its functionality. Furthermore, the tool could include test case generation and cross-reference the original code base and the migrated version, identifying potential inconsistencies between implementations. Such a tool would require heavy use of RAG to find and process code chunks throughout potentially large, production code bases developed over multiple years. Here, RAGProbe scenarios could be defined for each sub-component to evaluate its reliability. For example, it could help identify whether the retrieval mechanism reliably finds all relevant code segments to answer a given question.

Missing a crucial section of the code base that would have to be considered when making architectural changes for the target platform could potentially be catastrophic. In the worst case, work could have to be reverted due to incorrect decisions made with incomplete information. Furthermore, the ability of RAGProbe to identify cases where the model does not correctly recognize that it is unable to respond to questions is immensely valuable during the development of the system's RAG pipelines. For instance, hallucinated responses could cause developers to be misled about potential faults, which could waste a substantial amount of valuable development time or lead to wrong architectural decisions by developers who trust the tool. In contrast, false refusal when relevant context actually exists within the code base or documentation of the legacy software product could lead to key components being missed. Applying RAGProbe during the tool's development would involve gradually changing and adjusting each component in the LLM pipeline, including the prompts, to identify the optimal configuration for the task. Manually collecting, curating, and annotating relevant question-answer pairs for this process would incur substantial costs. In contrast, RAGProbe's standardized schemas have the potential to substantially reduce time and cost while also being more flexible as requirements evolve over time and changes in the evaluation scenarios may be required.

While we use code data as a part of our pretraining data mixes in our research, developing models that reason about or generate code is not my specialty. However, the methods I am developing to gain more insights into factual knowledge distribution within pretraining data could potentially aid RAGProbe in identifying more diverse, relevant, and challenging contexts to use for generating its evaluation data. This could, in turn, lead to more reliable evaluation results with broader coverage of potential failure modes. As mentioned, my own research can be adapted to integrate this method by including RAGProbe scenarios in downstream evaluation setups. The addition of this method could provide a broader understanding of the impact of my methodology on practical LLM pipelines for different tasks.

6 Research Ethics & Synthesis Reflection

For selecting relevant papers, I first narrowed down my search to the CAIN conference programs from 2024 and 2025. I chose these years since they have seen the most extensive impact of LLMs on software engineering, as increasingly powerful LLMs have been made commercially available and adapted in software engineering workflows in the months and years following ChatGPT's first public release at the end of 2022.

After manually scanning through the topics and paper titles in the CAIN 2024 and 2025 conference programs, I found myself gravitating towards topics related to sustainability, energy efficiency, privacy and copyright, and applications to testing and evaluation. I chose to focus on these topics since they are most relevant to my own research. From the papers I investigated more closely, I did not find any misleading titles. When I realized a paper was less relevant after closer inspection than initially anticipated, I moved on to the next paper on my shortlist.

I ensured originality by not using LLMs for generating any ideas or content during the writing process. I only used NLP tools to identify typographic or grammatical errors in my draft. Furthermore, I did not directly copy content from any of the sources. Where the assignment instruction explicitly required restating ideas from external sources, I rephrased and contextualized them rather than repeating them directly from the source, added my own thoughts, and referenced them appropriately.

References

[1] Z. Xu, S. Jain, and M. Kankanhalli, "Hallucination is inevitable: An innate limitation of large language models," *arXiv preprint arXiv:2401.11817*, 2024.

[2] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[3] Y. Zhao, A. Devoto, G. Hong, *et al.*, "Steering knowledge selection behaviours in llms via sae-based representation engineering," *arXiv preprint arXiv:2410.15999*, 2024.

[4] S. Sivasothy, S. Barnett, S. Kurniawan, Z. Rasool, and R. Vasa, "Ragprobe: Breaking rag pipelines with evaluation scenarios," in *2025 IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN)*, 2025, pp. 60–71.

[5] G. Team, A. Kamath, J. Ferret, *et al.*, "Gemma 3 technical report," *arXiv preprint arXiv:2503.19786*, 2025.

[6] A. Yang, A. Li, B. Yang, *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[7] C. Kastner, *Machine learning in production: from models to products*. MIT Press, 2025.

[8] D. Zhang, B. Xia, Y. Liu, *et al.*, "Privacy and copyright protection in generative ai: A lifecycle perspective," in *2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, 2024, pp. 92–97.

[9] G. Penedo, H. Kydlíček, V. Sabolčec, *et al.*, *Fineweb2: One pipeline to scale them all – adapting pre-training data processing to every language*, 2025. arXiv: 2506.20920 [cs.CL].

[10] E. Shahul, J. James, L. E. Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," in *Conference of the European Chapter of the Association for Computational Linguistics*, 2023.