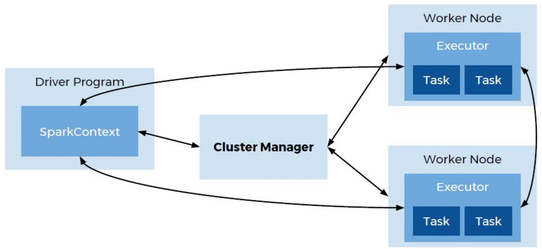
An AI Companion for Developers to Tune Complex Technologies

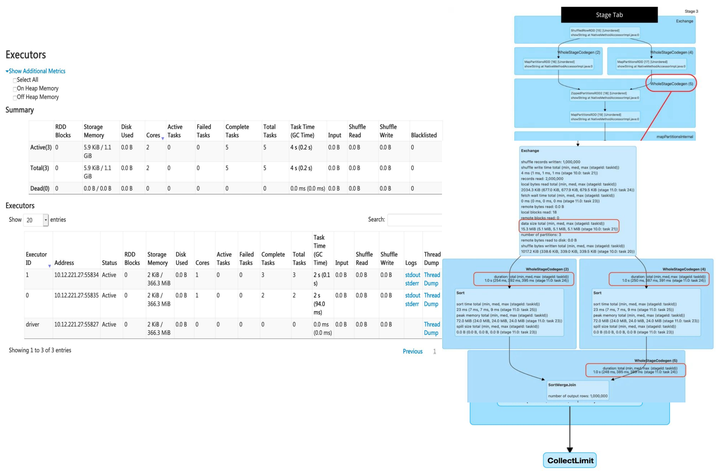
Yusuf Suat Polat
Onur Dilsiz
Supervisor: Bahri Atay Özgövde
Co-Supervisor: Yunus Durmuş

Motivation

In recent years, the progress of Large Language Models (LLMs) has transformed the software development. Although their capabilities have expanded rapidly, their effectiveness in debugging, optimizing, and reviewing code for complex technologies—such as Apache Spark and Kubernetes remains largely unproven. Our project examines the analysis capabilities of LLMs within the domain of Apache Spark, a widely adopted platform for parallelizing Big Data processing. With Apache Spark at its core, **Databrick**s' rapid growth and **$62 billion** valuation in December 2024 highlight the popularity and adoption of Spark as a tool for data processing.

In this work, we focus on how efficient LLMs can **review** and suggest optimizations for Spark code by addressing commonly encountered bad practices that degrade performance in terms of time and cost. By combining different models, prompting techniques, and architectures, we aim to determine the most effective ways to leverage LLMs for both **code analysis and optimization** in Spark development.

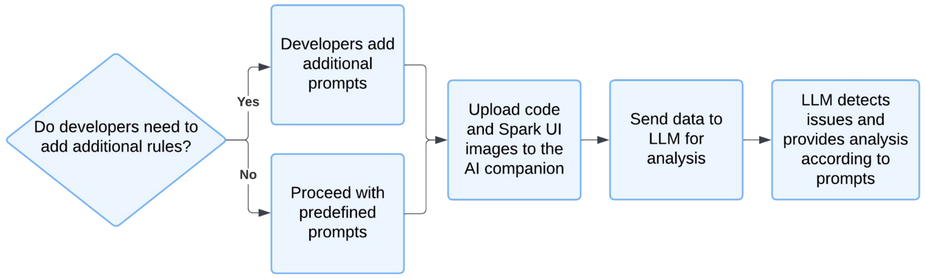




Challenges Encountered

- Authentic data scarcity
- Creating the test environment for various models
- Variety of real life configurations, lack of access to them
- Extracting information from Spark UI
- LLM Limitations for Specialized Domains
- Low Repeatability and Robustness of LLM performance

AI Companion Tool Event Flow



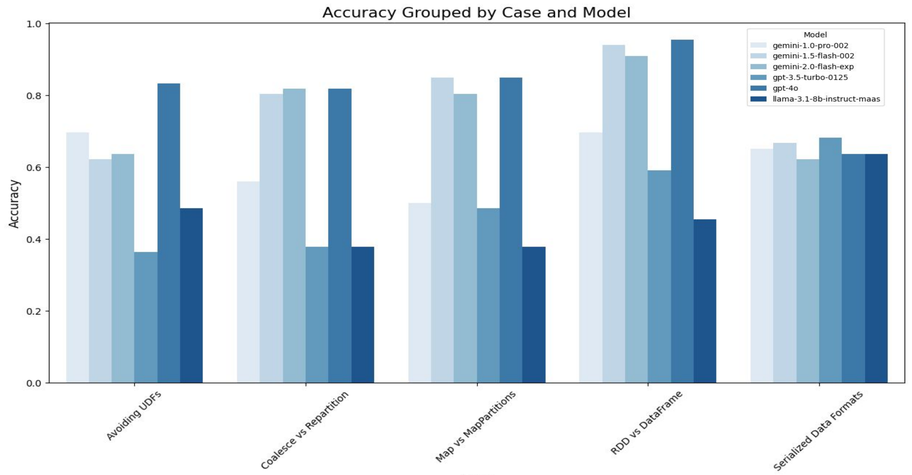
Evaluation Workflow

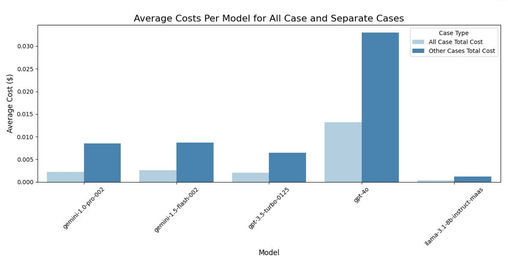
Our evaluation workflow is designed to systematically assess model performance. For each file in the dataset, the code contained within the file is embedded into predefined prompts. These prompts are then processed by various models. The response from each model is structured in JSON format, which is subsequently parsed to extract a binary detection result. This result serves as the basis for evaluating the models' metrics in detecting specific cases.

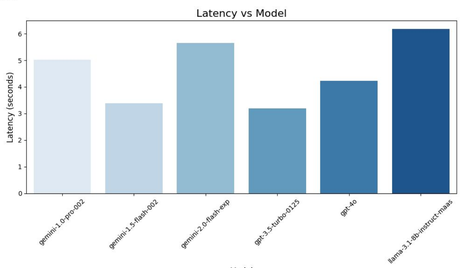
Inspected Cases

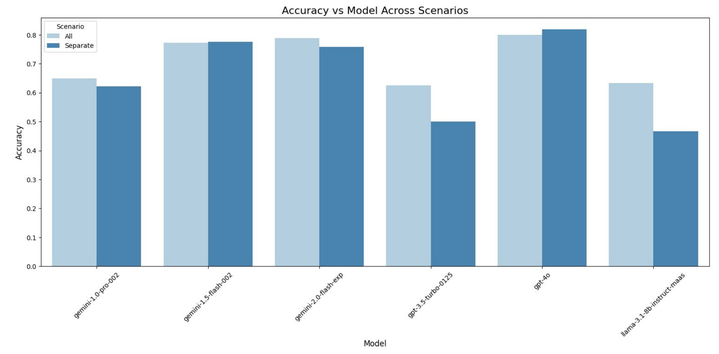
- ***Avoid UDFs:*** Use Spark SQL functions for better performance and optimization.
- ***Prefer coalesce() over repartition():*** Minimize shuffling when reducing partitions.
- ***Use mapPartitions() over map():*** Initialize heavy resources once per partition for better efficiency.
- ***Use DataFrame/Dataset over RDD:*** Leverage Catalyst and Tungsten optimizations; avoid expensive RDD serialization.
- ***Use Optimized Formats (Parquet/Avro):*** Faster, compressed, and optimized for big data analytics.

Results

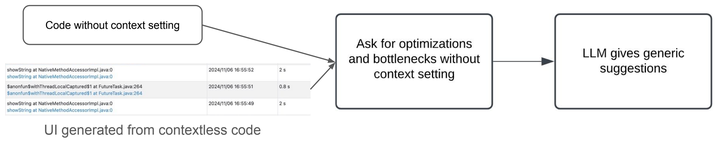




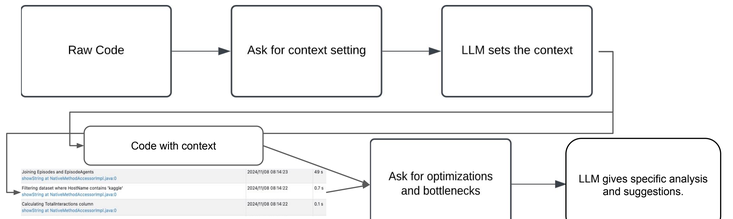




Multimodal Operation of AI Companion



UI generated from contextless code



UI generated from code with context

Conclusion

Based on our evaluation metrics, which primarily emphasize detection accuracy, we observed that the most successful model, GPT-4o, identifies over 80% of bad practices in the specified cases. High-parameter models demonstrate greater specificity and effectiveness, particularly when distinct prompts are tailored for each case. These findings emphasize the potential of LLMs for analysis and optimization, encouraging broader adoption within the industry.

Future Work

In the future, this study can be **expanded** in several ways:

- The companion can be integrated directly into common development environments as a plugin or provide an intuitive GUI, giving developers feedback and suggestions on their code.
- The collection and analysis of metrics can be automated by extending functionality to work seamlessly with the UI, reducing the need for manual screenshots.