SpikingBrain and Llama 2 Pass@1 Performance on Repository-Level Coding Benchmarks

Assignee Research

June 6, 2026

Abstract

This report synthesises findings from 4 peer-reviewed papers addressing the following research question: What is the difference in pass@1 scores between SpikingBrain and Llama 2 13B when evaluated on multi-file repository-level coding benchmarks. 15 claims were extracted from source literature; 0 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 2.4/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: CodePlan: Repository-level Coding using LLMs and Planning. Research question: What is the difference in pass@1 scores between SpikingBrain and Llama 2 13B when evaluated on multi-file repository-level coding benchmarks?.

2 Methodology

Systematic literature search across multiple databases yielded 4 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 2.4/10.

3 Results

4 papers retrieved. 15 claims extracted; 0 independently verified. Quality review score: 2.4/10.

4 Limitations

This report is a machine-generated literature synthesis and does not constitute original research. Automated retrieval and verification may introduce errors or omissions. Review scores reflect automated assessment, not human peer review. Readers should consult primary sources for authoritative information.

5 Extracted Claims

Claim	Verified	Confidence
The study employs two key metrics to assess CodePlan: Block Metrics and Edit Metrics.	×	0.02
Matched Blocks are defined as code blocks existing in the Source Repository, edited in the Target Repository, and also e	×	0.05
Missed Blocks are code blocks present in the Source Repository and edited in the Target Repository but not edited in the	×	0.06
Spurious Blocks are code blocks found in the Source Repository that were not edited in the Target Repository but were in	×	0.07
The ideal outcome for Block Metrics is a high number of Matched Blocks and low numbers of Missed and Spurious Blocks.	×	0.04
Levenshtein Distance measures the edit distance at the file level between the Predicted Repository and the Target Reposi	×	0.06
A higher Levenshtein Distance indicates that CodePlan did not make the correct changes to the repository.	×	0.05
Diff BLEU is calculated as BLUE(DIFF(Source repo file, Target repo file), DIFF(Source repo file, Predicted repo file)).	×	0.00
Diff BLEU compares the modified sections of code between the Predicted and Target Repositories while disregarding common	×	0.04
When modifications in Predicted and Target Repositories match precisely, Diff BLEU yields a score of 1.0.	×	0.01
Table (p2) contains an original C# function 'func' in 'Create.cs' that returns a tuple containing a complex object and a	×	0.00
Table (p2) shows a modified version of 'Create.cs' where the function 'func' returns a 'Complex' object constructed with	×	0.00
Table (p5) lists 'Plan Graph = nul' and 'Dependency Graph' as entries.	×	0.02
Table (p5) includes the operations 'Classify Changes', 'UpdateDependency Graph', 'GetSpatial Context', and 'Get Temporal	×	0.04
Table (p8) states that for Import/Using statement changes, the system must recompute the edges incident on the import st	×	0.03

References

- http://arxiv.org/abs/2309.12499v1
- http://arxiv.org/abs/1401.2781v4
- http://arxiv.org/abs/2311.00117v3