MA-DPR and BM25 Retrieval Recall Correlation with Code Generation Correctness Beyond 32K Tokens

Assignee Research

June 2, 2026

Abstract

This report synthesises findings from 15 peer-reviewed papers addressing the following research question: How does the retrieval recall of MA-DPR versus BM25 correlate with code generation correctness when scaling context windows beyond 32k tokens in RAG systems. 9 claims were extracted from source literature; 1 was independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 4.8/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Improving BM25 Code Retrieval Under Fixed Generic Tokenization: Adaptive q-Log Odds as a Drop-In BM25 Fix. Research question: How does the retrieval recall of MA-DPR versus BM25 correlate with code generation correctness when scaling context windows beyond 32k tokens in RAG systems?.

2 Methodology

Systematic literature search across multiple databases yielded 15 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 4.8/10.

3 Results

15 papers retrieved. 9 claims extracted; 1 independently verified. Quality review score: 4.8/10.

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
On CoIR-Go 182K under frozen generic tokenization, q-log at q = 0.05 improves NDCG@10 from 0.2575 to 0.4874, an absolute	✓	0.24
Java, Ruby, and JavaScript are significantly positive at their per-language qopt; PHP is not (CI spans zero); Python sel	×	0.03
Using the corpus-adaptive predictor of section 6 (qpred = 1-7.28 htok, fit once and frozen), a single-pass deployment re	×	0.07
Go captures 82.7% of the oracle gap at qpred = 0.54 (+0.190 absolute).	×	0.02
Java, PHP, and JavaScript land within 4% of the oracle setting, so the predictor realises almost the full oracle delta.	×	0.02
Python sits at qpred = 0.88, with a predicted delta within implementation noise of zero.	×	0.02
At q = 1 the expression (x1-q -1)/(1 -q) is 0/0 indeterminate. Differentiating numerator and denominator with respect to	×	0.07
For every q $\in$R, lnq is strictly increasing on x > 0, and the RSJ odds (N-nt+0.5)/(nt+0.5) is strictly decreasing in nt,	×	0.04
When nt N, the RSJ odds x $\rightarrow$N/nt grows large, and lnq(x) $\approx$ x1-q/(1 -q) for q < 1 (power-law amplification), log x for q	×	0.07

References

- http://arxiv.org/abs/2306.03091v2
- http://arxiv.org/abs/2605.18561v1
- http://arxiv.org/abs/2404.07220v2