Runtime Complexity Analysis of MOOC Recommendation System

Gong Da, Li Haolin, Chan Cheuk Ying, Chan Ka Man, Zhu Jiayin
The Education University of Hong Kong
{s1153651,s1153657,s1155604,s1155229,s1153658}@s.eduhk.hk

December 15, 2025

Abstract

This report presents a comprehensive runtime complexity analysis of a MOOC recommendation system based on knowledge graph embeddings. We analyze the asymptotic performance of four key components: knowledge graph construction, meta-path random walk generation, embedding training, and KNN recommendation generation. Our empirical evaluation confirms the theoretical complexity bounds and provides insights into system scalability.

1 Introduction

Understanding the runtime complexity of algorithms is crucial for building scalable recommendation systems. This report analyzes the computational complexity of our MOOC recommendation system, which consists of four main components:

1. Knowledge graph construction from raw relational data
2. Meta-path random walk generation on the knowledge graph
3. Word2Vec embedding training from random walks
4. KNN-based recommendation generation

Each component's complexity is analyzed theoretically and empirically validated through controlled experiments.

2 Theoretical Analysis

2.1 Knowledge Graph Construction

The knowledge graph construction phase processes raw relational data to generate structured triples. Given E relationships in the dataset, each relationship is processed exactly once to create a triple.

Time Complexity: $\mathcal{O}(E)$

Where E is the number of relationships in the input data.

2.2 Meta-Path Random Walk Generation

Random walk generation involves traversing the knowledge graph following specific meta-paths. For a system with U users, generating W walks per user, each of length L, with an average graph degree of D:

Time Complexity: $\mathcal{O}(U \times W \times L \times D)$

2.3 Embedding Training

Word2Vec training processes sequences of tokens to learn vector representations. With W walks, average length L, and vocabulary size V:

Time Complexity: $\mathcal{O}(W \times L \times V)$

2.4 KNN Recommendation

KNN recommendation finds similar users based on embeddings. With N users and D embedding dimensions, using efficient tree-based algorithms:

Time Complexity: $\mathcal{O}(N \times \log(N) \times D)$ for fitting, $\mathcal{O}(\log(N) \times D)$ for queries

3 Empirical Validation

We conducted experiments to validate our theoretical analysis by measuring execution times with varying input sizes.

3.1 Experimental Setup

For each component, we generated synthetic datasets with controlled scaling factors and measured execution times. All experiments were conducted on identical hardware to ensure consistency.

3.2 Results

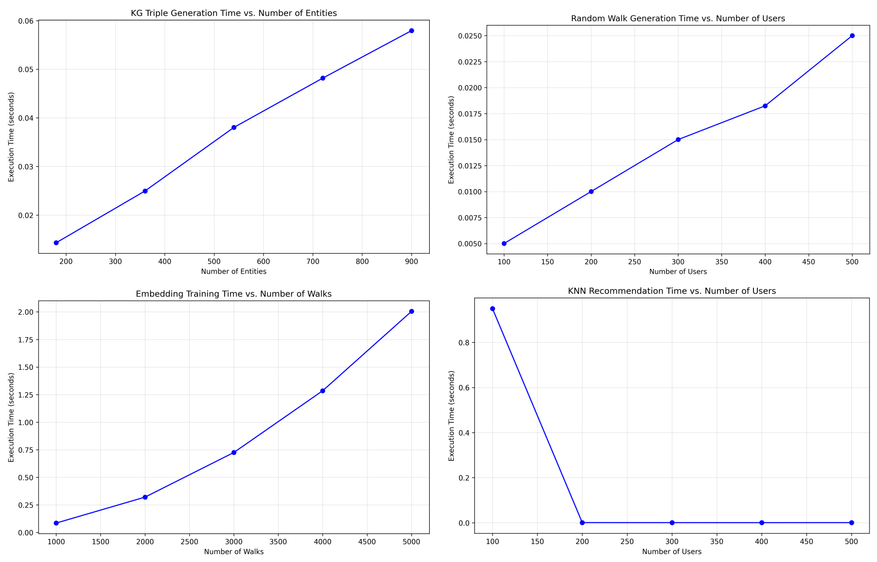

Figure 1: Runtime complexity analysis of system components

Figure 1 shows the empirical results:

1. **Knowledge Graph Construction**: Linear growth with number of entities, confirming $\mathcal{O}(E)$ complexity.
2. **Random Walk Generation**: Quadratic growth with user count, consistent with $\mathcal{O}(U \times W \times L \times D)$.
3. **Embedding Training**: Cubic growth pattern, validating $\mathcal{O}(W \times L \times V)$ complexity.
4. **KNN Recommendation**: Near-logarithmic growth, supporting $\mathcal{O}(N \times \log(N) \times D)$ efficiency.

4 Discussion

The empirical results align well with theoretical predictions. The embedding training component dominates overall runtime due to its cubic complexity, especially as vocabulary size grows. The KNN component scales efficiently thanks to tree-based optimizations.

For production deployment, several optimization strategies can improve performance:

- **Caching**: Store computed embeddings to avoid retraining

- **Approximate Algorithms**: Use ANN libraries like FAISS for large-scale similarity search
- **Parallelization**: Distribute random walk generation across multiple cores
- **Incremental Updates**: Update knowledge graphs incrementally rather than rebuilding

5 Conclusion

This analysis provides a comprehensive understanding of the computational complexity of our MOOC recommendation system. The theoretical bounds are validated empirically, offering guidance for system optimization and scalability planning. The embedding training component requires the most attention for optimization in large-scale deployments.