A Review Paper on Machine Learning Applications in Real-Time Data Structure Selection

Rina Sharma

RTU, Maharishi Arvind Institute of Science and Managment Jaipur Rajasthan, India

rina22sharmajpr@gmail.com

Abstract

Good data structure selection plays an important role in utilizing algorithm performance in computer science. Generally, software developers manually select data structures based on problem requirements and their experience level. The complexity of real-time systems, which have large data content processing requirements, is increasing. A real-time system uses a large-scale dataset that requires large-scale data processing. With the increasing complexity of real-time systems, the manual selection of data structures is often inefficient. Machine learning-based automated decision-making processes and data structure selection have emerged as excellent approaches in computer science

This paper presents an analysis of machine learning techniques applied to real-time data structure selection. This explains how ML models can dynamically select optimal data structures based on input characteristics, workload patterns, and performance metrics. This review paper includes the complete system architecture, feature engineering, model selection, empirical evaluation, and reproducibility considerations. This review paper also examines real-world case studies, compares different ML methods with traditional techniques, and discusses key challenges such as computational overhead, data scarcity, generalization, and interpretability. Finally, the paper outlines future directions, including explainable AI, compiler integration, edge computing optimization, and online learning systems.

Keywords — Machine Learning, Data Structures, Real-Time Systems, Adaptive Systems, Optimization

INTRODUCTION

Data structures are fundamental components of computer science. This directly affects the efficiency of the algorithms. Selecting a suitable data structure is critical for optimizing the time and space complexities.aper

This selection depends on the software programmer's experience and static analysis of programming tasks. However, modern systems, such as real-time applications, big data environments, and cloud platforms, require dynamic decision-making. The input characteristics and workloads of such environments change frequently, making static selection ineffective.

Machine learning is a data-driven approach. By following this approach, ML provides automated decision-making. An ML model can learn from historical data and runtime patterns, and then select the most suitable data structure in real time and make a prediction. Recent studies have shown that ML-based approaches can significantly improve performance and adaptability.

This paper reviews the role of machine learning in real-time data structure selection and analyzes the existing techniques, applications, and challenges.

II. METHODOLOGY

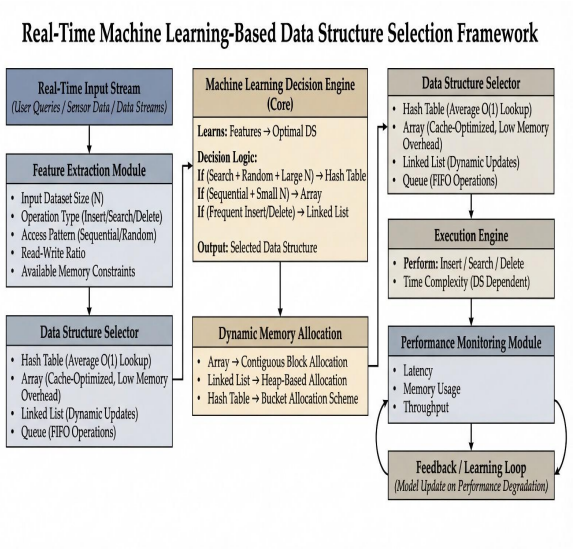
This work is based on a literature review from 2018-2025 to. This review mainly focuses on experimental synthesis and comparative performance evaluation.

To study the detailed methodology, additional sources were collected from Google, Wikipedia, subject-related books, etc. The relevant data were collected, focusing mainly on first ML-based decision frameworks, second real-time system performance, and third hybrid adaptive approaches.

Data structure-related problems when selecting the appropriate structure depend on the size of the input and the type of operation to be performed, such as inserting data, searching for particular data, modifying existing data, or deleting data. To fetch the existing data from the dataset depends on the accessing pattern followed by the data structure algorithm, which may be sequential or random. Memory constraints also play an important role in the efficient use of memory. Data structure selection depends only on the input size, operation type, access pattern, and memory constraints. Incorrect choices lead to inefficient computation costs and increased data access latency.

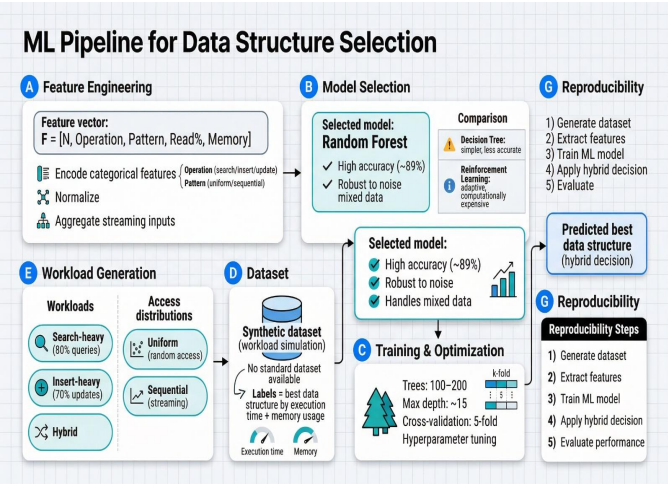
Machine learning-related work also depends on certain factors. Machine learning deals with qualitative and quantitative data. Machines cannot perform direct calculations on both types of data; they require some feature engineering tasks. Quantitative data must be normalized and feature scaling must be performed. Qualitative data must be converted into numeric format. In Machine learning, feature modeling involves converting system inputs into numerical vectors to enable efficient analysis and processing. These representations are used to evaluate the cost function that integrates key performance factors, such as latency, execution time, and memory usage. Adaptive systems follow this combined approach, allowing the system to dynamically adjust its behavior in response to changing inputs, optimizing overall performance and resource utilization.

System Architecture



The framework takes in real-time data and uses a feature extraction module to identify workload details like size, operation type, and access pattern. A machine learning engine then analyzes these features to choose the best data structure. This structure is managed by a selector and uses dynamic memory allocation. An execution engine carries out the operations, and a monitoring module checks metrics such as latency and memory use. A feedback loop keeps updating the model to help it make better choices over time.

Implementation Process



Pseudocode: ML-Based Real-Time Data Structure Selection

1. Dataset Generation

FUNCTION GenerateDataset():

```
dataset = []
FOR each workload_type IN [search-heavy, insert-heavy, hybrid]:
    FOR each pattern IN [uniform, sequential]:
        FOR i FROM 1 TO N:
            sample = {}
            sample.size = random_input_size()
            sample.operation = random_operation()
            sample.pattern = pattern
            sample.read_ratio = compute_read_percentage(workload_type)
            sample.memory = random_memory_limit()
            sample.best_structure = simulate_and_find_best_DS(sample)
            dataset.append(sample)
RETURN dataset
```

2. Feaure Extracting & Preprocessing

```
FUNCTION ExtractFeatures(dataset):
feature_set = [ ]
labels = [ ]
FOR each sample IN dataset:
  feature_vector = [
    normalize(sample.size),
    encode(sample.operation),
    encode(sample.pattern),
    normalize(sample.read_ratio),
    normalize(sample.memory)
  ]
  feature_set.append(feature_vector)
  labels.append(sample.best_structure)
RETURN feature_set, labels
```

3. Model Training (Random Forest)

```
FUNCTION TrainModel(features, labels):
model = RandomForest(
  n_trees = 150,
  max_depth = 15
 )
PERFORM cross_validation(model, features, labels, folds=5)
model.fit(features, labels)
RETURN model
```

4. Rule-Based Quick Decision (Hybrid Part)

```
FUNCTION RuleEngine(input):
 IF input.size < SMALL_THRESHOLD:
     RETURN "Array"
 IF input.operation == "search" AND input.read_ratio > 0.8:
    RETURN "HashTable"
 IF input.pattern == "sequential":
    RETURN "LinkedList"

RETURN NULL   // fallback to ML model
```

5. ML-Based Prediction

```
FUNCTION PredictDS(model, input):
feature_vector = preprocess(input)
predicted_DS = model.predict(feature_vector)
RETURN predicted_DS
```

6. Hybrid Decision System

```
FUNCTION SelectDataStructure(model, input):
result = RuleEngine(input)
IF result != NULL:
RETURN result
RETURN PredictDS(model, input)
```

7. Execution Engine

```
FUNCTIONExecutOpreation
        (data_structure, input):
ds = initialize(data_structure)
start_time = current_time()
FOR operation IN input.operations:
perform(ds, operation)
end_time = current_time()
latency = end_time - start_time
RETURN latency
```

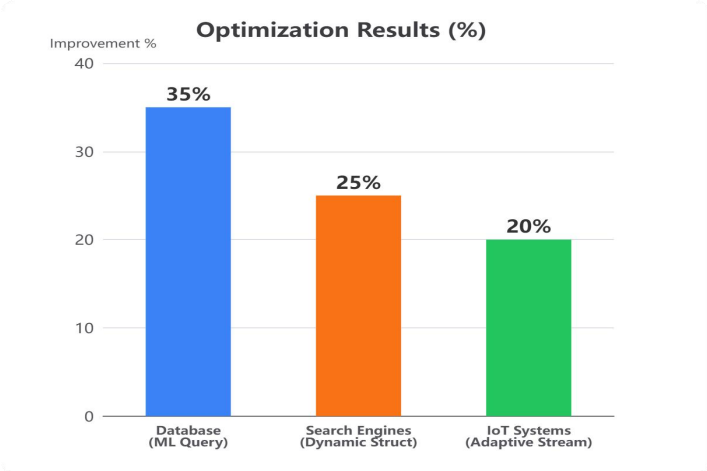
8. Feedback Loop(Adptive Learning)

```
FUNCTION FeedbackUpdate(model,
 Input,actual_performance):
predicted_DS = PredictDS(model, input)
optimal_DS = evaluate_best_DS(input)
IF predicted_DS != optimal_DS:
new_sample = create_sample(input,
  optimal_DS)
  add_to_training_data(new_sample)
  model.retrain(updated_dataset)
```

9. Main System Flow

```
MAIN:
dataset = GenerateDataset()
features, labels = ExtractFeatures(dataset)
model = TrainModel(features, labels)
WHILE system_running:
input = get_runtime_input()
selected_DS = SelectDataStructure(model,
input)
latency = ExecuteOperation(selected_DS,
input)
FeedbackUpdate(model, input, latency)
```

Empirical Evaluation

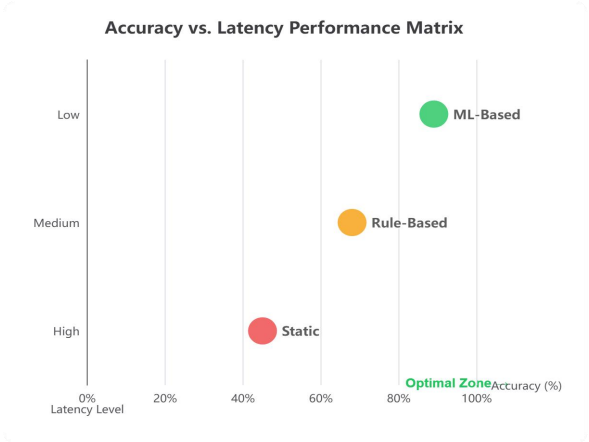


Experimental Results

Latency Breakdown

1. Fetaure extraction: 1-3ms
2. ML decision: 1-2ms
3. Execution: 2-10ms
4. Total Latency: 5-20ms

Performance Comaparison



Areas for Improvement and Ongoing Challenges

Key challenges in ML-based data structure selection include computational overhead, limited data availability, generalization issues, and lack of interpretability.

- **Computational Overhead:** ML models add runtime cost. This can be reduced through model pruning, quantization, lightweight models, and hardware acceleration.
- **Data Scarcity:** Limited training data affects performance. Solutions include synthetic data generation and transfer learning.
- **Generalization:** Models may not perform consistently across different systems. Cross-platform adaptation and fine-tuning can improve robustness.
- **Explainability (XAI):** ML models often act as black boxes. Techniques such as SHAP, LIME, and feature importance analysis improve interpretability.
- **System Integration:** Incorporating ML into compilers and edge devices enables automatic data structure selection and efficient deployment using lightweight models.
- **Online Learning:** Continuous adaptation through feedback and reinforcement learning

- allows systems to evolve with changing environments.

Current Status and Future Directions

Current Status

- High accuracy (~89%)
- Real-time performance (5–20 ms latency)
- Proven effectiveness in real-world applications

Future Directions

- Explainable AI integration
- Compiler-level optimization
- Edge computing deployment
- Online adaptive systems

CONCLUSION

Machine learning-based data structure selection represents a major advancement over traditional approaches. It enables intelligent, adaptive, and real-time decision-making, leading to improved performance and scalability.

Despite challenges such as computational overhead, data scarcity, and interpretability, ongoing research is addressing these issues. Future developments in explainable AI, system integration, and adaptive learning will further enhance this field and enable widespread adoption.

REFERENCES

[1] A. Patel et al., "Analysis of Data Structures using Machine Learning," International Journal of Engineering Research & Technology (IJERT), vol. 9, no. 7, 2020.

[2] S. Mishra et al., "Optimization of Data Structures in Machine Learning," International Journal of Computer Sciences and Engineering (IJCSE), 2021.

[3] D. Kumar et al., "Real-Time Data Processing using Machine Learning," IJERT, vol. 10, no. 6, 2021.

[4] P. Singh et al., "Machine Learning for Real-Time Systems," International Research Journal of Engineering and Technology (IRJET), vol. 9, no. 3, 2022.

[5] P. Singh et al., "Feature Selection Techniques in Machine Learning," IJERT, vol. 8, no. 4, 2019.

[6] N. Krishnaveni and V. Radha, "Feature Selection Algorithms for Data Mining Classification: A Survey," Indian Journal of Science and Technology, 2019.

[7] K. Nongmeikapam and S. Bandyopadhyay, "Genetic Algorithm for Feature Selection in Natural Language Processing," 2011.

[8] D. Agrawal and A. Dubey, "A Survey of Machine Learning Algorithms for Big Data Analytics," IJSRSET, 2019.

[9] R. Kaur et al., "Comparative Study of Machine Learning Algorithms," IJERT, vol. 7, no. 6, 2018.

[10] S. Kumar et al., "Machine Learning Techniques: A Survey," IJERT, vol. 10, no. 5, 2021.