

REAL-TIME CONTENT FILTERING USING RABIN-KARP ALGORITHM

TEAM 8

Introduction

- Content filtering is essential for detecting and removing inappropriate text in real-time applications.
- Various string matching algorithms are used, including Brute Force, Knuth-Morris-Pratt (KMP), Boyer-Moore, and Rabin-Karp.
- This document compares these algorithms and highlights why Rabin-Karp is a better choice.

Algorithm Overview

Rabin-Karp Algorithm

- Hash-based string matching technique.
- Computes hash values for substrings and compares them with the pattern hash.
- Reduces the number of character comparisons.
- Efficient for multiple pattern matching in large texts.

Comparison with Other Algorithms

Rabin-Karp Algorithm

- Best for searching multiple patterns simultaneously.
- Uses hash comparisons, reducing character-by-character checks.
- Best-case time complexity: $O(n + m)$, Worst-case: $O(n * m)$ (hash collisions).
- Space complexity: $O(m)$ (for pattern hashes).
- Ideal for real-time filtering, scalable for large datasets.
- Hash collisions may cause occasional slowdowns.

Knuth-Morris-Pratt (KMP) Algorithm

- Uses a preprocessing phase with a prefix function.
- Works efficiently for single-pattern matching.
- Time complexity: $O(n + m)$ in worst case.
- Space complexity: $O(m)$ (prefix table for pattern).

- Guaranteed worst-case linear performance.
- Not optimized for multiple pattern searches.

Brute Force Algorithm

- Simple character-by-character comparison.
- No preprocessing required.
- Time complexity: $O(n * m)$ in all cases.
- Space complexity: $O(1)$, no extra storage needed.
- Easy to implement but inefficient for large datasets.

Boyer-Moore Algorithm

- Uses heuristic-based pattern preprocessing (bad character and good suffix rules).
- Skips unnecessary comparisons, optimizing search.
- Best-case time complexity: $O(n / m)$, Worst-case: $O(n * m)$ (rare cases).
- Space complexity: $O(m)$ (preprocessing tables).
- Highly efficient for single-pattern searches in large texts.
- Not ideal for multiple pattern matching.

Why Rabin-Karp is Better for Real-Time Filtering

- **Handles multiple patterns efficiently**, unlike KMP and Boyer-Moore.
- **Fast real-time processing**, leveraging hash comparisons.
- **Scalability** for filtering large sets of banned words.
- **Minimal preprocessing** compared to KMP and Boyer-Moore.
- **Works well in streaming applications** where continuous filtering is required.
- **Adaptable to dynamic content filtering** with minimal computational overhead.

Conclusion

- Rabin-Karp is highly effective for real-time content filtering due to its **multi-pattern matching capability**.
- **Fast hash-based comparisons** make it ideal for detecting banned words in chat moderation, spam filtering, and web content filtering.
- Other algorithms have specific advantages, but Rabin-Karp excels in **real-time, large-scale, and dynamic filtering scenarios**.
- Implementing Rabin-Karp in content moderation systems improves **accuracy, speed, and scalability** for efficient text filtering.