SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Improved Symbol Table Construction for FSST Compression

Hedi Chehaidar

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Improved Symbol Table Construction for FSST Compression

Verbesserte Symboltabellenkonstruktion für die FSST-Komprimierung

Author:	Hedi Chehaidar
Examiner:	Prof. Thomas Neumann
Supervisor:	Mihail Stoian
Submission Date:	January 7, 2026

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, January 7, 2026 Hedi Chehaidar

Acknowledgments

First and foremost, I would like to thank God for the guidance, strength and perseverance that made it possible to complete this work.

I would also like to express my sincere gratitude to all the people who supported, inspired, and challenged me throughout the work on this thesis.

I would like to thank my supervisor Mihail Stoian for the continuous guidance, insightful feedback, and trust in my ideas. The discussions we had, ranging from high-level algorithmic ideas to low-level implementation details, were invaluable and significantly shaped the direction and quality of this work. I am especially grateful for the encouragement to explore optimizations deeply and to question existing assumptions about compression performance.
I am deeply thankful for the practical insights, constructive criticism, and technical expertise, particularly regarding algorithmic correctness and performance evaluation. The feedback I received helped turn initial ideas into a well-structured and measurable contribution.

I am grateful to the broader research and open-source community whose work laid the foundation for this thesis. The availability of high-quality implementations, documentation, and discussions around compression techniques made it possible to build upon existing work and focus on meaningful improvements.
A special thanks goes to everyone who shared tools, datasets, or prior experience that helped accelerate experimentation and evaluation.

Finally, I want to thank my friends and family for their constant support, patience, and encouragement throughout this journey. Their motivation helped me stay focused and enthusiastic, especially during long debugging sessions and performance tuning phases.

This thesis would not have been possible without the collective contributions of all these people, and I am truly thankful for their support and inspiration.

Abstract

Modern analytical database systems process large volumes of string data, where efficient compression is essential for reducing memory footprint and improving performance. While the Fast Static Symbol Table (FSST) compression scheme provides excellent decompression speed and random-access capabilities, its effectiveness heavily depends on the quality of the constructed symbol table.

The original FSST algorithm relies on heuristic, greedy symbol selection, which can lead to suboptimal symbol choices and limit achievable compression ratios.

This thesis presents several enhancements to the FSST compression method that improve symbol table construction while preserving FSST's core advantages. The central contribution is a refined symbol selection process that systematically identifies more effective symbols and avoids redundant or conflicting choices.

First, a dynamic programming approach is introduced to evaluate and select higher-quality symbols within each generation, enabling a more globally informed optimization compared to the original greedy strategy. Second, an additional frequency counter is incorporated to accelerate the discovery of longer symbols and to explicitly favor them in subsequent generations, improving the exploitation of longer recurring patterns in the data. Third, a symbol pruning mechanism is applied to eliminate conflicting and redundant symbols, ensuring a more compact and effective symbol table.

Together, these techniques significantly improve the robustness and quality of the symbol table generation process. Experimental evaluation demonstrates that the proposed enhancements lead to consistently improved compression ratios compared to the original FSST algorithm, while maintaining its fast decompression and random-access properties. The results show that careful algorithmic refinement of symbol selection can yield substantial gains without altering the lightweight and practical nature of FSST, making the improved approach well suited for use in modern analytical systems.

Contents

1 Introduction

1.1 Motivation

Modern data management systems increasingly operate on large-scale, string-heavy datasets. In analytical databases, strings are omnipresent and appear in many forms, including URLs, file paths, identifiers, log messages, categorical attributes, and semi-structured data originating from web, cloud, and enterprise applications. Studies of real-world database workloads show that string columns often constitute a substantial fraction of the overall data volume, both in terms of storage and memory consumption. Prior analyses of analytical benchmarks and production systems report that strings can account for a significant portion of columnar storage, frequently dominating memory usage in dictionary-encoded or compressed representations [1], [2].

The growing prevalence of string data places strong demands on compression techniques used in analytical systems. Effective compression reduces memory footprint, improves cache utilization, and lowers memory bandwidth pressure, all of which are critical for high-performance query processing. At the same time, analytical workloads require fast random access to individual values, as queries often scan and decode only a subset of columns or rows. This combination of requirements makes general-purpose, block-based compressors such as Zstandard or LZ4 less suitable despite their strong compression ratios, as they typically require decompressing entire blocks before accessing individual strings.

To address this gap, lightweight string compression schemes have been proposed that prioritize fast decompression and random access. One prominent example is the FSST compression algorithm [3]. FSST compresses strings by replacing frequent byte sequences with compact symbols from a statically constructed symbol table. During decompression, symbols can be expanded independently, allowing direct access to individual strings without scanning neighboring data. As a result, FSST achieves decompression speeds that are competitive with, and often superior to, more heavyweight compression schemes, making it attractive for use in modern analytical databases.

However, the compression effectiveness of FSST critically depends on the quality of its symbol table. The original FSST algorithm constructs this table using a greedy approach that iteratively selects symbols based on local heuristics. While this strategy is computationally efficient and aligns with FSST's design goal of lightweight processing, it can lead to suboptimal symbol choices. In particular, greedy selection may favor short or locally frequent symbols that conflict with longer or more informative sequences, limit the discovery of beneficial longer symbols, or introduce redundancy within the symbol table. Furthermore, once a symbol is selected, its impact on future generations of symbols is not globally optimized, which can prevent the algorithm from converging toward a symbol set that maximizes overall compression gain.

These limitations suggest that there is room for improvement in FSST's symbol table construction without compromising its core advantages. By revisiting the greedy nature of symbol selection and incorporating more informed decision-making into the construction process, it is possible to improve compression ratios while retaining FSST's fast decompression and random-access properties. This thesis explores such improvements, focusing on enhanced symbol selection strategies that address the shortcomings of the original greedy approach.

1.2 Thesis Outline

This thesis is structured as follows.

Chapter 2 (Background) introduces the fundamental concepts required to understand the techniques developed in this thesis. It begins with an overview of dynamic programming and contrasts it with greedy algorithmic approaches, highlighting their respective strengths and limitations. The chapter then introduces the core data structures used throughout the thesis, namely tries and max-heaps, which play a central role in symbol generation and selection.

Chapter 3 (Related Work) reviews existing work in the area of data compression. It first discusses widely used block-based compression algorithms such as LZ4 and Zstandard, explaining their general design principles and trade-offs. The chapter then presents the original FSST compression algorithm, detailing its symbol table construction and encoding process, which form the basis for the improvements proposed in this thesis.

Chapter 4 (Approach) describes the main contributions of this work. It introduces three enhancements to the FSST symbol table construction process: a dynamic programming-based method for improved symbol selection, the introduction of a third frequency counter to accelerate the discovery and prioritization of longer symbols, and a symbol pruning strategy to eliminate conflicting and redundant symbols. Each technique is explained in detail and integrated into the overall compression pipeline.

Chapter 5 (Evaluation) evaluates the proposed enhancements experimentally. It presents the datasets, experimental setup, and benchmarking methodology used in the evaluation. The chapter analyzes the impact of the proposed techniques on compression ratio and runtime, comparing the improved FSST variants against the original algorithm.

Chapter 6 (Conclusion) summarizes the contributions and findings of the thesis. It reflects on the achieved improvements and discusses directions for future work, with emphasis on potential optimization opportunities to reduce the runtime overhead introduced by the enhanced symbol selection techniques.

2 Background

2.1 Dynamic Programming vs. Greedy

2.2 Datastructures

2.2.1 Max Heap

2.2.2 Trie

3 Related Work

3.1 Block-based Compressors

3.2 FSST

4 Approach

4.1 Dynamic Programming

4.2 3^{rd} Counter

4.3 Symbol Pruning

5 Evaluation

5.1 Environment

5.2 Benchmarking Data

5.3 Results

6 Conclusion

6.1 Summary

6.2 Future Work

List of Figures

List of Tables

Bibliography

[1] Peter A. Boncz, Martin L. Kersten, "Monet: An impressionist sketch of an advanced database system," *Proc. IEEE BIWIT workshop, San Sebastian (Spain)*, pp. 240–251, 1995.

[2] D. J. Abadi, S. R. Madden, and N. Hachem, "Column-stores vs. row-stores," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, L. V. S. Lakshmanan, R. T. Ng, and D. Shasha, Eds., New York, NY, USA: ACM, 2008, pp. 967–980. DOI: 10.1145/1376616.1376712.

[3] P. Boncz, T. Neumann, and V. Leis, "Fsst: Fast random access string compression," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2649–2661, 2020. DOI: 10.14778/3407790.3407851.