Efficient Document Analytics on Compressed Data: Method, Challenges, Algorithms, Insights

Feng Zhang, Jidong Zhai, Xipeng Shen, Onur Mutlu, Wenguang Chen

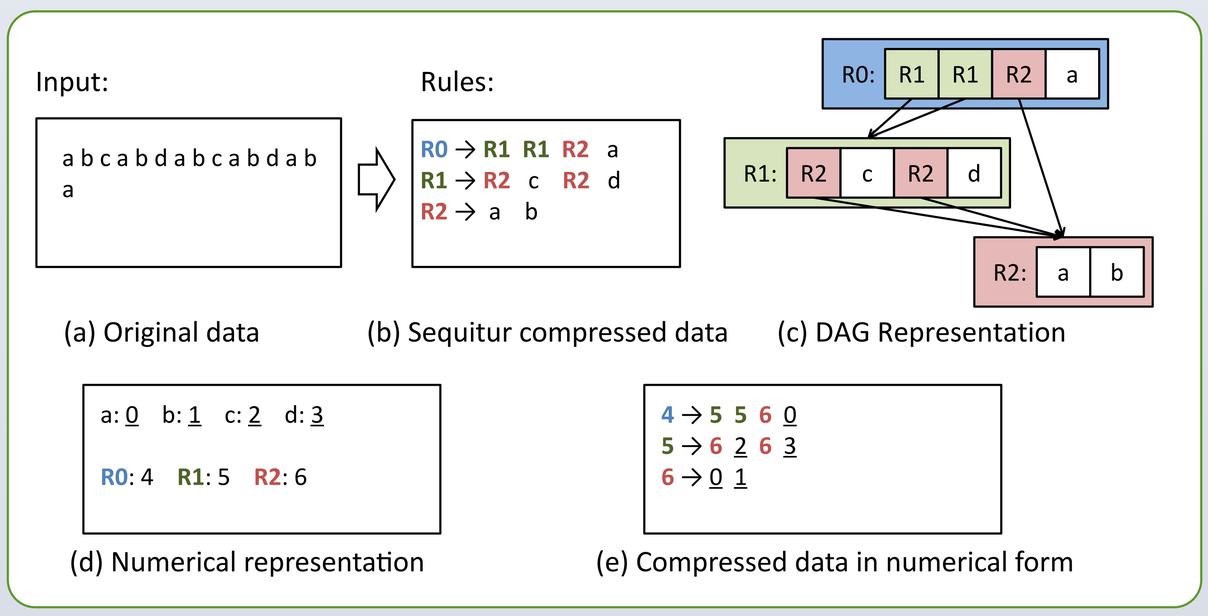
Renmin University of China, Tsinghua University, North Carolina State University, ETH Zürich

ABSTRACT

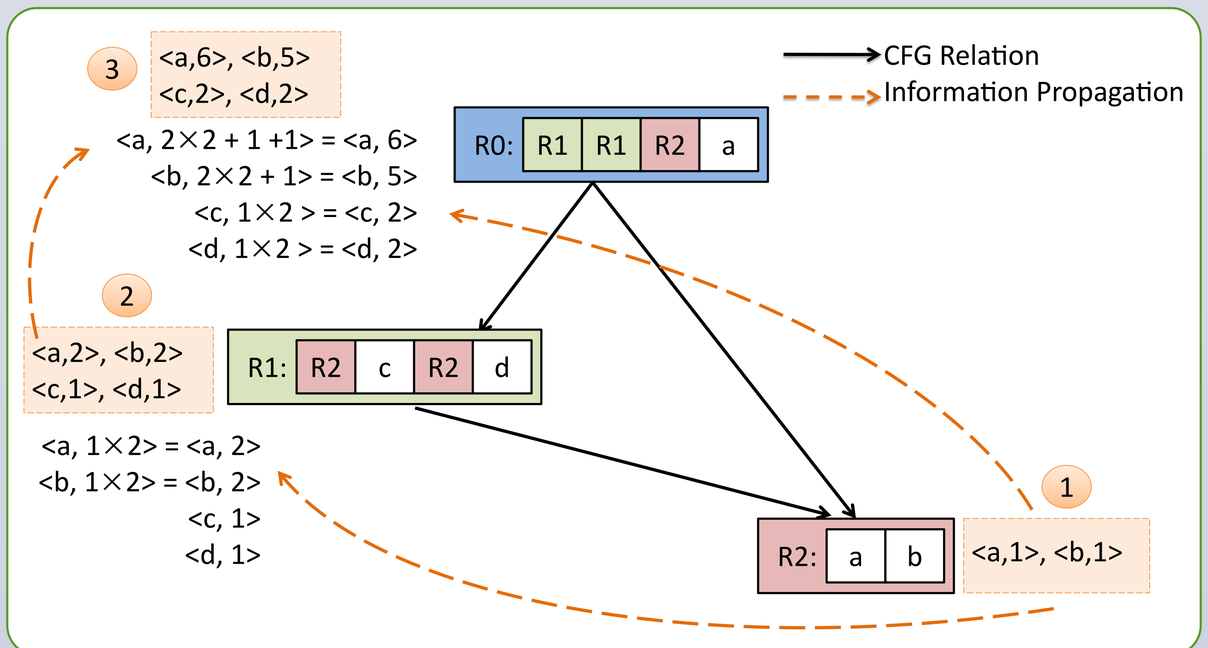
Today's rapidly growing document volumes pose pressing challenges to modern document analytics, in both space us- age and processing time. In this work, we propose the concept of compression-based direct processing to alleviate issues in both dimensions. The main idea is to enable direct document analytics on compressed data. We present how the concept can be materialized on Sequitur, a compression algorithm that produces hierarchical grammar-like representations. We discuss the major challenges in applying the idea to various document analytics tasks, and reveal a set of guidelines and also assistant software modules for developers to effectively apply compression-based direct processing. Experiments show that our proposed techniques save 90.8% storage space and 77.5% memory usage, while speeding up data processing significantly, i.e., by 1.6X on sequential systems, and 2.2X on distributed clusters, on average.

KEY IDEA

A compression example with Sequitur.

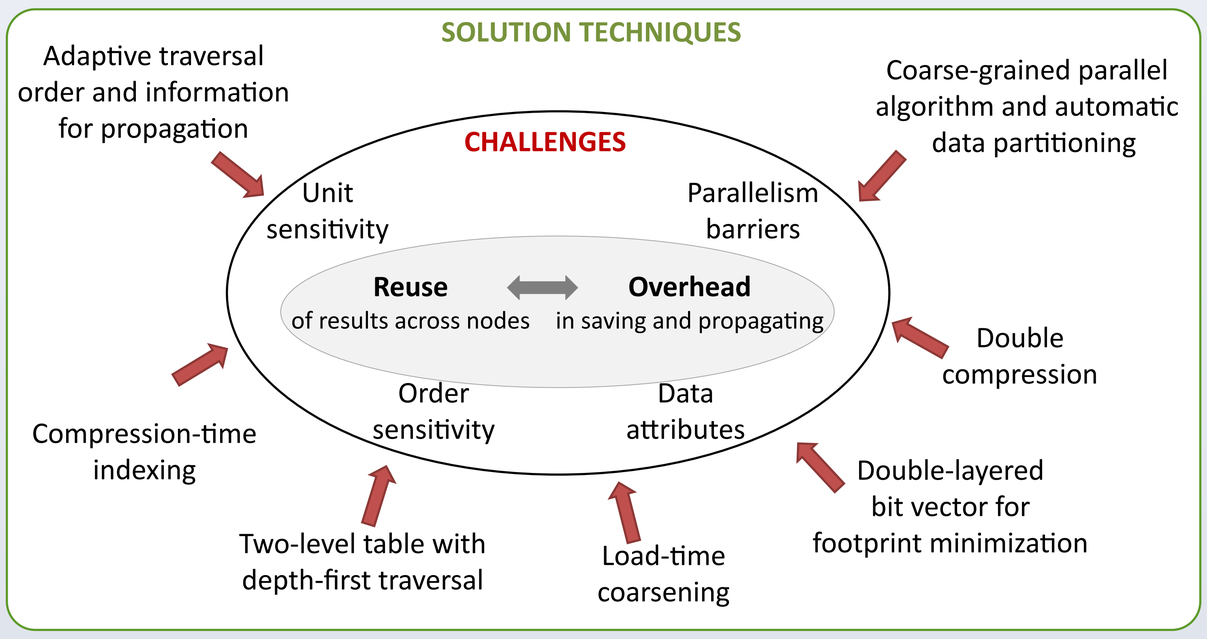


A DAG from Sequitur for "a b c a b d a b c a b d a b a", and a postorder traversal of the DAG for counting word frequencies.



CHALLENGES

Effectively materializing the concept of compression-based direct processing on document analytics faces a number of challenges. These challenges center around the tension between reuse of results across nodes and the overheads in saving and propagating results. Reuse saves repeated processing of repeated content, but at the same time, requires the computation results to be saved in memory and propagated throughout the graph. The key to effective compression-based direct processing is to maximize the reuse while minimizing the overhead.



GUIDELINES

Guideline I: Try to minimize the footprint size of the data propagated across the graph during processing.

Guideline II: Traversal order is essential for efficiency. It should be selected to suit both the analytics task and the input datasets.

Guideline III: Coarse-grained distributed implementation is preferred, especially when the input dataset exceeds the memory capacity of one machine; data partitioning for load balance should be considered, but with caution if it requires the split of a file, especially for unit-sensitive or order-sensitive tasks.

Guideline IV: For order-sensitive tasks, consider the use of depth-first traversal and a two-level table design. The former helps the system conform to the word appearance order, while the latter helps with result reuse.

Guideline V: When dealing with analytics problems with unit sensitivity, consider the use of double-layered bitmap if unit information needs to be passed across the CFG.

Guideline VI: Double compression and coarsening help reduce space and time cost, especially when the dataset consists of many files. They also enable that the thresholds be determined empirically (e.g., through decision trees).

RESULTS