ADVANCE DATA COMPRESSION TECHNIQUES:

Title:

Effective Chat History Compression for Messaging Applications

Problem Statement:

Current messaging applications store and send large amounts of chat data with redundant phrases, emojis, and media URLs. This redundancy causes excessive usage of storage, higher bandwidth usage, and longer synchronization time, particularly in resource-constrained environments. Existing compression techniques are not tailored for the distinct, repetitive nature of dialogue data and hence are inefficient for this application.

Objectives:

1. Create a lossless compression method specially designed for chat histories.

2. Find and substitute repeated text, emojis, and media URLs with concise symbolic notation.

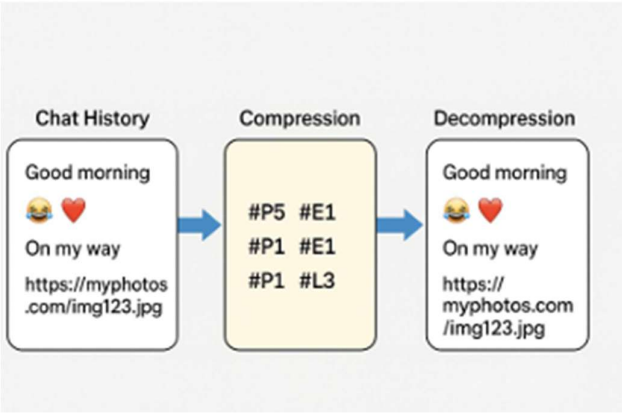
3. Measure performance factors like storage capacity and transmit time prior to and after compression.

4. Present a proof-of-concept demonstration illustrating real-time improvement in efficiency

Abstract:

Messaging apps are now a crucial part of everyday life, producing enormous amounts of stored and transferred chat data. This information tends to involve duplicate text patterns, emojis, and media links, leading to storage wastage and higher network loads. This project suggests a domain-specific lossless compression scheme that identifies and compresses these repetitive elements into compact symbolic codes, which greatly minimizes message size. The process is made for easy incorporation within messaging systems without compromising message readability and format. Experimental assessments will quantify decreases in storage capacity, transmission time, and bandwidth use, showing the promise of this method to improve efficiency in personal and business communication environments.

Diagrammatic view:



Literature survey:

S: No	Paper/Technique	Key Idea	Limitations in chat context
1.	Huffman & Arithmetic Coding	Statistical coding reduces redundancy by assigning shorter codes to frequent characters.	Works well for text, but not efficient for short, repetitive chat-like messages.
2.	Lempel-Ziv-Welch (LZW)	Dictionary-based compression for repeated patterns	Performs poorly with very short repeated tokens (common in chats).
3.	Domain-Specific Compression (e.g., DNA/Log data)	Custom compression tailored to repetitive domain patterns.	Chat data needs domain-specific adaptation for emojis/URLs.
4.	WhatsApp Backup Studies	Large chat backups consume device storage and cloud space.	Current apps do not optimize for redundancy.
5.	Emoji Representation Research	Emojis cause high redundancy; substituting them with symbolic codes improves efficiency.	Generic algorithms don't handle emojis effectively.
6.	Adaptive Dictionaries	Dynamically builds dictionaries from frequent words/phrases	Memory overhead in real-time messaging scenarios.
7.	Log File Compression	Symbolic replacement of repeated patterns in log data	Similar to chat, but lacks handling for emojis/URLs.

8.	Real-Time Compression in Messaging	Attempts at live compression with low latency.	Trade-off between speed and compression ratio.
9.	Hybrid Compression (Pattern + Huffman)	Combines substitution + entropy coding.	Complexity increases for lightweight devices.
10.	IoT / Low Bandwidth Research	Lightweight compression reduces sync times in constrained environments.	Needs to be adapted for chat applications.

Module Preparation:

S:No	Module Name	Description	Input	Output
1.	Preprocessing Module	Cleans and tokenizes chat history, handles timestamps, names, and emojis.	Raw chat logs (WhatsApp/Telegram export)	Tokenized clean data
2.	Pattern Recognition Module	Detects repeated words, phrases, emojis, and media URLs.	Tokenized chat data	Pattern dictionary
3.	Symbolic Substitution Module	Replaces frequent patterns with compact symbolic codes (@E1, @U1, etc.).	Tokenized data + dictionary	Encoded symbolic data
4.	Compression Algorithm Module	Applies chosen algorithm (e.g., Huffman)	Encoded data	Compressed binary file

5.	Decompression Module	Reconstructs original chat without data loss.	Compressed file + codebook	Original chat text
6.	Performance Evaluation Module	Compares storage, bandwidth, and transmission time before vs. after compression.	Raw vs. compressed data	Evaluation metrics & graphs
7.	Integration & Demo Module	Demonstrates integration inside a messaging app prototype.	Chat app + compression module	Proof-of-concept app

Team member:

S. RISHIKESH - 21MID0217