ChatGPT

+--+ | **Methodology for Adaptive Compression of AI Chat Logs in JSON Format** | | | | **Abstract**: This paper presents a robust, adaptive methodology for compressing AI | | chat logs into a JSON-based memory database. The process reduces storage footprint | | while maintaining retrievability and decoding accuracy. It employs a two-tiered | | compression technique: dynamic number substitution for high-frequency tokens and | | consonant-focused reduction for the remainder. A meta index and global dictionary | | versioning ensure long-term consistency and optimization. | | | | **Introduction**: AI chat logs (~10,000 words, ~60,000 characters) pose challenges | | in storage, retrieval, and semantic indexing. Inspired by OpenAI's JSON chat | | exports, this system summarizes logs to ~1,000 words and compresses them via a | | dual-method strategy that preserves semantic value while optimizing byte efficiency.| | Output is stored in a structured JSON format for robust decoding and AI-friendly | | querying. | | | | **Methodology**: | | | | 1. **Summarization**: | | - **Objective**: Reduce a 10,000-word log to ~1,000 words (~6,000 characters). | | - **Process**: A large language model is prompted to summarize the log, | | capturing main ideas, queries, responses, and insights (~84% reduction). | | - **Output**: Summary text for compression. | | | | 2. **Compression**: | | - **Dynamic Method A: Number Substitution**: | | - **Scope**: Identify and replace top 20 most frequent words with integers | | (1–20), without spaces. | | - **Mapping**: Stored in a block-specific index (e.g., "the" → "1"). | | - **Savings**: Replaces ~350 words, saving ~875 characters. | | - **Method B: Consonant-Based Reduction**: | | - **Scope**: Applies to remaining ~650 words (~3,960 characters). | | - **Process**: Remove vowels (except first, if ambiguous), trim repeated | | consonants (e.g., "tt" → "t"), add space. Uses a global dictionary of | | ~1,000 entries for reliable reconstruction. | | - **Savings**: ~1.7 characters/word on average (~1,105 characters total). | | - **Meta Indexing**: | | - Block-specific index maps Method A tokens. | | - Global dictionary includes version control via SHA-256 checksum. | | - **Output**: Compressed text (~3,900 characters), block map (~0.5KB), | | dictionary ref (~1KB), metadata. | | | | 3. **Storage and Decoding**: | | - **Storage Format**: Store as JSON object: | | |

```
json
{
    "chat_id": "abc123",
    "create_time": "2025-06-09T12:00:00Z",
    "compressed_text_content": "1qustn2ans...",
    "block_map": { "1": "the", "2": "and", ... },
    "global_dict_version_id": "sha256:abcd...",
    "char_count": 3900,
    "compression_ratio": 0.65,
    "meta": { "source_length": 6000 }
  }
```

| | | - **Decoding Process**: | | - Tokenize compressed text. | | - If token is a number (1–20), map via block_map (Method A). | | - Else, map consonant clusters via global dictionary (Method B). | | - Reassemble into readable summary; flag ambiguous mappings. | | | | **Results**: This system compresses a 1,000-word summary by ~33% (from ~6,000 to | | ~3,900 characters), with transparent decoding and indexable storage. | | | | **Conclusion**: The JSON-based compression framework effectively reduces storage | | while preserving information integrity. Future enhancements include entropy-based | | token ranking, language-aware stripping rules, semantic embeddings, and adaptive | | summarization prompting for compression-aware outputs. | | | | **Word Count**: 441 | +----------+