An Efficient Approach to Store and Access Wikipedia's Revision History

Amit Arjun Verma
2016csz0003@iitrpr.ac.in
IIT Ropar

S.R.S Iyengar
sudarshan@iitrpr.ac.in
IIT Ropar

Neeru Dubey
neerudubey@iitrpr.ac.in
IIT Ropar

Simran Setia
2017csz0001@iitrpr.ac.in
IIT Ropar

Abstract

For large-scale Wikipedia analysis, efficient retrieval of past states of Wikipedia is a prerequisite. However, the lack of efficient tools for managing the massive amount of provided data acts as a bottleneck. We present a detailed analysis of online algorithms to efficiently compress and retrieve the revision history of Wikipedia articles. We give theoretical evidence that our methods perform efficient compression and extraction while optimizing time and space complexity. Moreover, the experiments on sampled Wikipedia articles using the online parameters extraction method show that our algorithm can compress the dataset up to 93% of its original size.

Keywords: Wikipedia, edit-history, datasets, nlp, compression, algorithm

Introduction

Over the past decade, Wikipedia, the free online collaborative encyclopedia, has been a subject of enormous interest in the various research domains. At the core of all the mentioned research is the Wikipedia's full revision history dataset. Each revision of a Wikipedia article is being stored as the combination of preceding revision and the current edits, resulting in the accumulation of redundant information. Owing to this, the size of each article reaches megabytes or even gigabytes. This massive size of full revision history dataset acts as a bottleneck in performing Wikipedia-based research as the majority of these analysis tasks require a deep investigation of each revision. we present an algorithm to efficiently extract the edits by reconstructing a specific past state of Wikipedia from its edit history. The rationale behind data compression is to store the edits made in the current revision exclusively. This compression is relatively simple and can be achieved by taking the *difference* of the current revision with the previous one using the diff algorithm (Hunt and MacIlroy, 1976). However, it is evident that there is a trade-off between the retrieval time and the compression of revisions. Thus, to accelerate the reconstruction process, every k^{th} revision is stored as a full revision. A similar approach was followed by Ferschke et al. (Ferschke et al., 2011), where the value of k (interval length) was fixed to 1000 irrespective of input article. Following the lines of Ferschke et al.'s work, Verma et al. (Verma et al., 2021) proposed a method for compressing the revision history using the interval length $k = \sqrt{n}$. We propose an online algorithm to compress and retrieve the Wikipedia article's revisions, which can scale up with the Wikipedia dataset size. We show that our approach outperforms the previously established methods in terms of retrieval time and space complexity. With the proposed algorithm, we reduce the required storage space to less than 7% of its original size.

Varibale-Interval Length Compression

Assume there are n revisions for a Wikipedia article, define $\mathcal{R} = \{r_i \mid r_i \text{ is the } i^{\text{th}}\text{revision}, 0 \leq i \leq n\}$ to be the set of all n revisions of the article, where r_0 denotes the empty revision. $\mathcal{R}$ can also be considered as the XML dump of a Wikipedia article. Let dr_i denote the difference between two consecutive revisions i and $i-1$ i.e., $dr_i = r_i \ominus r_{i-1}$, where $\ominus : \mathcal{R} \times \mathcal{R} \rightarrow d\mathcal{R}$ is the *diff* operator which is defined as the smallest set of deletions and insertions required to create one text from the other (Hunt and MacIlroy, 1976). Thus, $d\mathcal{R}$ is the set of all *difference* revision obtained from *diff* operator. Since edits made in the current revision results in the successive revision, r_i can also be written as $r_i = r_{i-1} \oplus dr_i$, where $\oplus : \mathcal{R} \times d\mathcal{R} \rightarrow \mathcal{R}$ restores one of the revisions that generated dr, acting as a *decompressor*. One way of storing the data is to store only the difference dr_i, $\forall i \in [1, n]$ rather than the entire revision. A revision r_i can be retrieved by loading $d\mathcal{R}$ to the main memory and recursively constructing r_1, r_2, $\cdots$, r_i using dr_1, dr_2, $\cdots$, dr_{i-1} respectively. Moreover, the reconstruction time and compression ratio depends on the type of algorithm used to compute the *diff* between the two revisions.

We propose an online algorithm by storing the full revisions at variable interval lengths. We translate this optimization problem in to an ordered set partition problem. More specifically, for a Wikipedia article, we define $S = \{s_i \mid s_i = ||r_i||, 0 \leq i \leq n\}$ as an ordered set of revision sizes, where $||r_i||$ represents the text length of revision $r_i \in \mathcal{R}$. For the simplicity sake assume $||r_i \ominus r_{i-1}|| = |s_i - s_{i-1}|$ ($|.|$ is the absolute function), which means we can compute $||dr_i|| = |s_i - s_{i-1}|$ (we will relax this assumption later). Given a set S for a corresponding Wikipedia article, we define P as a partition of the set S such that:

(a) $P = \{p_1, p_2, p_3, \ldots, p_N\}$, where $N \leq n$.

(b) p_j (for some j) is either a singleton set or if $s_l, s_m \in p_j$ and if $l < m$, then $s_i \in p_j\ \forall i \in (l, m)$.

(c) Given $p_j = \{s_i \mid l \leq i \leq m\}$, we define function f^- on p_j as

$$f^-(p_j) = \begin{cases} s_i, & \text{if } p_j \text{ is a singleton} \\ s_l + \sum\limits_{i=l}^{m-1} |s_{i+1} - s_i|, & \text{otherwise} \end{cases}$$

(d) Given $p_j = \{s_i \mid l \leq i \leq m\}$, we define function t on p_j as

$$t(p_j) = \begin{cases} 1, & \text{if } p_j \text{ is a singleton} \\ 1 + \sum\limits_{i=l}^{m-1} s_i + |s_{i+1} - s_i|, & \text{otherwise} \end{cases}$$

Given the partition and the function definition, the summation $\sum f^-(p_j), \forall j \in N$ represents the overall size of the set S (i.e. size of the Wikipedia article) after compression. Consider a simple example where a Wikipedia article contains only three revisions and their sizes are $r_1 = 1$, $r_2 = 2$, and $r_3 = 8$. Given the size of each revision, we can represent the set $S = \{1, 2, 8\}$. If we partition this set S such that $P = \{\{1\}, \{2, 8\}\}$ then $\sum f^-(p_j)$ will be 9, which we refer as the total *space cost*. But what about the revision retrieval time? Given a revision r_i and the difference $d_i = r_i \ominus r_{i+1}$, retrieving the revision r_{i+1} will take $\mathcal{O}(||r_i|| + ||dr_i||)$ time. Which means that overall *time cost* for a given set S will be $\sum t(p_j), \forall j \in N$ (in the case of $S = \{1, 2, 8\}$, the time cost is 11, $\mathcal{O}(1)$ unit for r_1 and r_2, whereas $\mathcal{O}(2+6)$ for r_3). Now provided a set S, the problem reduces to finding a partition P such that the overall time cost and the space cost is minimised. More specifically:

- $\sum f^-(p_j), \forall j \in N$ is minimized and,
- $\sum t(p_j), \forall j \in N$ is minimized.

However, the optimization function as the summation of space and time cost ($\sum f^-(p_j) + \sum t(p_j), \forall p_j \in P$) provides a solution other than the original arrangement of the set S, only if there exists at least two consecutive revisions having the difference precisely equal to 1. Moreover, the solution is never unique. To overcome this challenge, we convert the optimization problem into a memory cost minimization problem based on a fixed time cost. More specifically, given a set S and a fixed time cost (as a function of $n = |S|$), the optimization problem reduces to finding a partition P that minimizes the overall space cost. We first start with computing the differences between all the consecutive revisions. We compute the *memory cost saved* as $mr_i = ||r_{i-1}|| - ||dr_i||$, whereas the *time cost* represents the time units required to retrieve a specific revision. Given a fixed time cost, we aim to maximize the memory cost saved. It is easy to verify that this maximization problem can be translated into the 0/1 knapsack problem, where the *time cost* is the knapsack size, and the *memory cost saved* is the profit.

Results

The preliminary results show that optimal compression is performed using the variable-interval length method. Moreover, we observe the optimal compression ratio and revision retrieval time when we fix the *maximum time cost* (C) to *nlogn*. The variable-interval length method ($C = nlogn$) even outperforms the fixed-interval length method in terms of time (0.079 seconds on average) respecting the same compression ratio (ratio of 0.168 on average). The reason behind this optimality is the idea of collating all the consecutive minor edits into a single block. Furthermore, given a *maximum time cost*, the method guarantees the optimal compression.

References

[Ferschke et al.2011] Oliver Ferschke, Torsten Zesch, and Iryna Gurevych. 2011. Wikipedia revision toolkit: efficiently accessing wikipedia's edit history. In *ACL*, pages 97–102. Association for Computational Linguistics.

[Hunt and MacIlroy1976] James Wayne Hunt and M Douglas MacIlroy. 1976. *An algorithm for differential file comparison*. Bell Laboratories Murray Hill.

[Verma et al.2021] Amit Arjun Verma, SRS Iyengar, Simran Setia, and Neeru Dubey. 2021. An open source toolkit to parse and analyze online crowd-sourced portals.