Quantization in Append-Only Collections

Salman Mohammed, Matt Crane, and Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada
{salman.mohammed,matt.crane,jimmylin}@uwaterloo.ca

ABSTRACT

Quantization, the pre-calculation and conversion to integers of term/document weights in an inverted index, is a well studied aspect of search engines that substantially improves retrieval efficiency. Previous work has considered the impact of quantization on effectiveness–efficiency tradeoffs in retrieval, for example, exploring the relationship between collection size and quantization range in static web collections. We extend previous work to append-only collections and examine whether quantization settings derived from prior time periods can be applied to future time periods. Experiments confirm that previous results generalize to a collection with different characteristics and with a different ranking function, and that in an append-only collection, we can use previous quantization settings in future time periods without substantial losses in either effectiveness or efficiency.

1 INTRODUCTION

The query latency of search engines is dominated by the time taken to traverse postings in the inverted index to compute the relevance score of documents with respect to queries. This can be reduced in several ways, for example, by using a postings traversal algorithm that skips documents that cannot enter the final ranked list (such as Wand [2]) or using an anytime algorithm to enforce an upper bound on the number of postings processed (such as Jass [9]); see Crane et al. [4] for a recent comparison of these approaches.

Quantization describes another class of techniques for improving the efficiency of query evaluation. This approach has a long history: Persin et al. [12] proposed ordering postings by decreasing term frequency. This not only allows documents with the highest term frequencies to be processed first, but also amortizes the cost of computing the term frequency component of the ranking function across documents sharing the same term frequency.

Moffat et al. [10] observed that approximations to components of the ranking function (in particular, the document length) are as effective as exact values. Anh et al. [1] further observed that term/document weights can be pre-computed. However, such values are floating point and do not compress well, which is addressed by quantizing scores into q-bit integers. The result is a so-called *impact-ordered* index. Several approximations were explored by Anh et al. [1], who showed that either a linear mapping, or alternatively skewing to provide extra granularity to lower scores, works well. With quantization, query evaluation reduces to integer additions. Subsequently, Crane et al. [5] showed that there is a relationship between the size of the document collection, query efficiency, and the number of quantization bits needed to maintain the power of the search engine to discriminate between documents (by reducing the number of scoring ties).

To our knowledge, prior work in quantization has focused on static collections. That is, the entire collection has been indexed and quantization can occur with the benefit of knowing that no more documents are to be added, which would change the quantization range. In this paper, we extend previous work to handle append-only collections, which we define as collections that can be split into temporal blocks that accumulate over time, each of which can be considered static.

We frame our research around the question of whether we can calculate quantization parameters using an earlier temporal block and apply the parameters to future blocks without negatively impacting effectiveness. For simplicity, we use uniform linear scaling of the term/document weights across the collection proposed by Anh et al. [1]. As our question is primarily framed in terms of effectiveness, we conducted our experiments, without loss of generality, using the ATIRE search engine [13]. This approach also allows for direct comparisons to the results of Crane et al. [5].

2 APPEND-ONLY COLLECTIONS

Problematically for collections that grow over time, such as in a typical web search engine (where a crawler continuously adds new pages) or Twitter (where new tweets are constantly being posted), quantization will degrade search effectiveness over time. Based on Crane et al. [5], while eight bits are sufficient for a collection of 25M documents, nine bits are needed for 50M documents. A naïve solution would be to re-quantize the entire collection when such bit thresholds are crossed. However, in the process of quantization, the information that is needed to re-quantize the index is discarded. It is possible to re-quantize to a smaller number of bits, but this is the opposite of what we desire.

However, if we consider append-only collections where past documents are considered static and there is a method by which the collection can be partitioned into blocks, then it may be possible to learn quantization settings on one block and apply those settings to future blocks. This would allow quantization during indexing, rather than as a separate post-indexing process, meaning that the benefits of quantization can apply to non-static collections.

To study whether this approach is feasible, we experimented on the Tweets2013 collection used for the TREC 2013 and 2014 Microblog Tracks [7, 8]. The collection contains approximately

ICTIR'17, October 1–4, 2017, Amsterdam, The Netherlands.

978-1-4503-4490-6/17/10…$15.00
DOI: http://dx.doi.org/10.1145/3121050.3121092

Documents	254,456,742
Size	11.4GB
Unique Terms	89,759,296
Total Terms	3,221,307,052
Date Range	February, 1 2013 to March, 28 2013
TREC Queries	MB 111–225

Table 1: Tweets2013 collection statistics.

254 million tweets gathered over February and March 2013, and provides an obvious temporal partitioning—this is exactly the implementation used by Twitter's Earlybird search engine [3]. Although Twitter recently switched to relevance ranking of search results (from reverse chronological order), it appears that the implementation still depends on retrieving candidate tweets and then reranking them [6]. Although the Tweets2013 collection includes information about tweets that were subsequently deleted by users, we ignore this information and include all tweets. This is justified because deletes are typically handled via so-called tombstones: they are kept in the index but filtered from query results.

For queries, we used topics from the TREC 2013 and 2014 Microblog Tracks, each of which includes a query time. In the track setup, results exploited term statistics of tweets posted after the query time,[1] but the track guidelines required that results only contain tweets posted before the query time [7, 8]. This requirement was relaxed in our experiments, as we are not directly interested in absolute effectiveness, but rather only in relative comparisons to a non-quantized index. To measure effectiveness, the relevance judgments were also split to contain judgments for only those tweets that occur within the period being assessed. This means that our results are not directly comparable to other results reported in the literature that conformed to the official track guidelines.

Table 1 shows collection statistics for the Tweets2013 collection.[2] We see that this collection is quite different from web collections that have been previously used to study quantization—two obvious differences are the number and length of documents, which affects the vocabulary size as well. Crane et al. [5] used BM25 as the ranking function, but it is widely known that BM25 does not work well for tweet collections since tweets are relatively uniform in length, which negates the effects of the length normalization factor in BM25. Instead, for our experiments we used the LMJM ranking function [16], with $\lambda = 3000$ (the ATIRE default). Ranking functions based on language modeling have been shown to perform better for tweets [11].

We additionally examined the ClueWeb09 Category B (CW09B) collection, which contains 50 million documents crawled from the web in early 2009. For queries we used TREC queries 51–150. The collection was distributed in crawl order, which provides an approximate temporal ordering. One can imagine a web search engine performing quantization as crawling and indexing proceeds. We tested only on folders in the collection that start with the `en00` prefix, as the `enwp` files are injected outside of crawling.[3]

[1] Although it was shown that use of this "future information" had no significant impact on retrieval effectiveness [14].

[2] Note that we discarded the last few days of the complete collection as to work with complete weeks in our partitioning scheme.

[3] `http://www.lemurproject.org/clueweb09/datasetInformation.php`

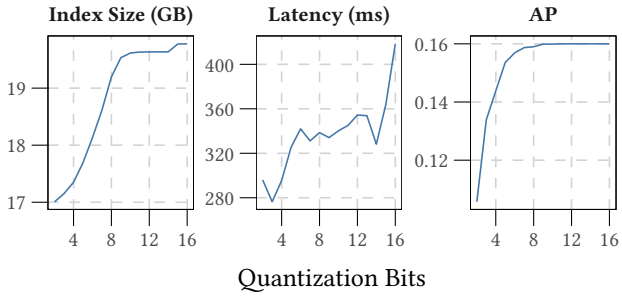

Figure 1: Effects of the number of quantization bits using LMJM on the Tweets2013 collection with respect to index size (measured in GB), mean query latency ($k = 1000$, measured in ms), and effectiveness (measured using AP).

3 EXPERIMENTS

Our experiments were conducted on an otherwise idle server with two Intel E5-2670 2.60GHz CPUs (8 cores) with 256GB of memory, running Linux kernel 3.13.0. All implementations were single threaded. Source code for our experiments is available online for reproducibility purposes.[4]

3.1 Analysis of the Complete Collection

In light of the different collection characteristics and the ranking function, our first experiment aimed to verify that the effects of quantization observed by Crane et al. [5] generalize. Figure 1 shows the effects that the number of quantization bits has on the index size (measured in GB), query latency for $k = 1000$, and search effectiveness (measured using AP) over the entire collection. The query latency is represented as the average time for queries in the test collections, and for an individual query we selected the minimum time from five runs. The minimum was selected because noise introduced in timing can only be additive, and thus the minimum is closer to the ground truth than either the mean or median. While recent work by Crane et al. [4] have suggested that only showing the mean latency has a number of drawbacks, we are merely attempting to replicate results from prior work. Indeed, what we see is entirely consistent with what has been reported in the literature.

3.2 Prediction of Quantization Settings

Having confirmed that the trends observed in prior work apply to this tweet collection and ranking function, we now turn our attention to the dynamic nature of the collection. To test this we split the collection into weeks (recall that we only examined the eight full weeks in the collection). In addition, we also performed experiments on the CW09B collection, split using the folder layout described in the previous section.

When documents from the current period are being indexed, we can use the quantization settings learned from the previous period to quantize term/document weights on the fly, rather than as a post-processing step. The quantization settings are the number of bits and the range of the observed weights. The concerns with this are two-fold: First, the range of the values may be substantially different across time periods, affecting the uniformity of the quantization.

[4] `https://github.com/snapbug/quant-time`

		Terms		LMJM Score		Required Bits		Crane et al. [5]	
Subset	Documents	Unique	Total	min	max	$p < 0.05$	$p < 0.01$	AP	P@20
Complete	254,456,742	89,759,296	3,221,307,052	0.092340	21.275933	6	6	15	10
Week 1	30,010,597	18,247,199	379,188,879	0.092185	19.136566	13	3	9	6
Week 2	26,213,500	16,449,208	329,256,187	0.091704	18.995803	5	4	9	6
Week 3	30,050,788	18,431,524	380,556,865	0.092369	19.140002	3	3	9	6
Week 4	30,531,252	19,089,511	387,982,579	0.092652	19.159073	3	3	9	6
Week 5	30,241,679	18,949,096	384,368,344	0.092664	19.149702	4	4	9	6
Week 6	31,054,512	19,333,063	394,602,513	0.092643	19.175997	3	3	9	6
Week 7	31,053,073	19,455,396	394,973,784	0.092724	19.176866	3	3	9	6
Week 8	31,581,904	19,615,690	398,909,012	0.092153	19.187290	5	4	9	6

Table 2: Collection statistics and quantization range for the entire collection and weekly subsets of Tweets2013. Final columns show the required number of bits for different significance levels with respect to AP (experimentally determined), and values predicted by Crane et al. [5] for $p < 0.01$.

Second, the number of bits devoted to quantization may be either too low—resulting in a significant loss in effectiveness—or too high—resulting in a loss in efficiency.

Table 2 shows collection statistics for the different weekly temporal subsets of Tweets2013 as well as the quantization settings. For example, there are 30 million tweets from the first week of February, which contain 380 million occurrences of 18 million unique terms. The range of LMJM term/document weights for this week was 0.092185–19.136566. As shown in the table, the range of the weights is remarkably consistent, within the same granularity, across different temporal subsets.

The table also shows the minimum number of bits required to maintain effectiveness with respect to average precision. Following Crane et al. [5], we selected the smallest number of bits needed to maintain effectiveness that is not significantly different at both $p < 0.05$ and $p < 0.01$ from a non-quantized index (calculated using a two-tailed paired t-test). We also require that no higher bit setting has a statistically significant difference: since quantized ranking converges to results from a non-quantized index as the number of bits increases, if there is a higher bit setting that is significantly different, then the non-difference at the lower setting is likely due to noise, rather than any "real" difference.[5]

The final two columns in Table 2 show the estimated numbers of bits that are needed in terms of AP and P@20 using the relationship and parameters described by Crane et al. [5]. Clearly, this vastly overestimates the required number of bits compared to the $p < 0.01$ setting (the setting used to derive the relationship) when targeting either metric. There is an obvious need to consider collection characteristics and the ranking function, which Crane et al. [5] did not do. The predicted values more closely match the results we derive for subsets of the CW09B collection, which is unsurprising given the collections the formula was derived from.

Figure 2 shows the effects of quantization on mean latency and AP per week. For example, the minimum number of bits required, at the $p < 0.05$ level, for Week 7 is three, while for Week 8 it is five. Applying the value learned from the week 7 subset to the week 8 subset would result in a loss in AP, compared to the oracle minimum bits, of 0.0272, bringing it to a level that is significantly different from a non-quantized index. Accompanying this is a mean query latency decrease of 5% (2ms).

Figure 3 shows the changes in AP and mean latency when using the learned settings from the previous period, with $p < 0.05$, compared to the oracle setting. Whether the setting was over, equal to, or under the oracle value for the current period is identified by both shape and color. The figure clearly shows that in general whenever the bit setting was greater than needed, the mean latency suffered. While the AP values are higher, these are not significantly different from a non-quantized index. When the prediction was lower than the oracle, the mean efficiency improved at the cost of statistically significant differences in AP.

4 CONCLUSIONS AND FUTURE WORK

We have shown that the overall results of Crane et al. [5] extend to another collection (with very different characteristics) as well as a different ranking function. We were then able to divide collections into temporal subsets and show that by using settings obtained for one period, we can effectively quantize the next time period without substantial loss in effectiveness or efficiency.

There are a number of open questions worth exploring: The formula provided by Crane et al. [5] vastly overpredicts the number of bits required for the Tweets2013 collection, since it does not account for collection characteristics and the ranking function. The number of bits required for this collection is surprisingly low, especially given the gains observed by resolving tie-breaks [15]. Finally, we only explore the scenario where judgments are applied to a future time period, as opposed to the entire collection cumulatively. While this is not unrealistic for tweets since users generally care about the most recent posts, such a setup may be questionable in the web scenario. We save these issues for future work.

REFERENCES

[1] Vo Ngoc Anh, Owen de Kretser, and Alistair Moffat. 2001. Vector-Space Ranking with Effective Early Termination. In *SIGIR*. 35–42.

[2] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient Query Evaluation using a Two-Level Retrieval Process. In *CIKM*. 426–434.

[3] Michael Busch, Krishna Gade, Brian Larson, Patrick Lok, Samuel Luckenbill, and Jimmy Lin. 2012. Earlybird: Real-time Search at Twitter. In *ICDE*. 1360–1369.

[5] This requirement was also used by Crane et al. [5], although the description is missing from that paper.

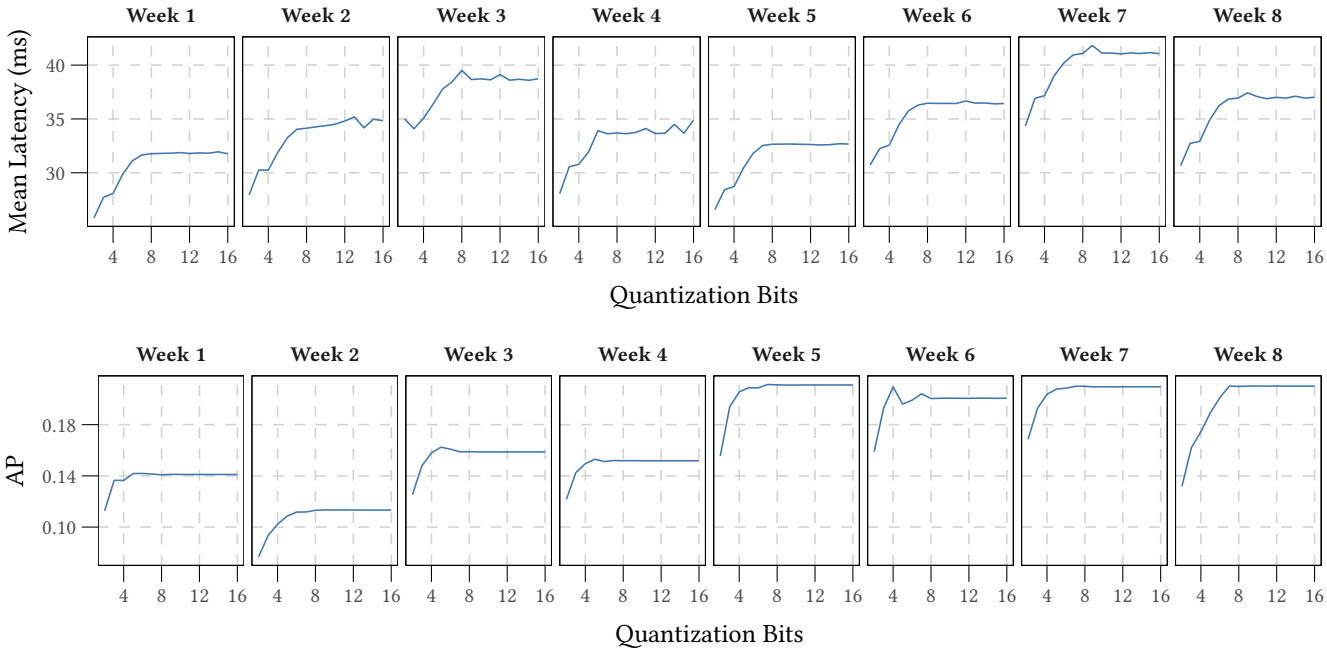

Figure 2: Effects of quantization bits on mean latency (top) and AP (bottom) for weekly subsets of the Tweets2013 collection.

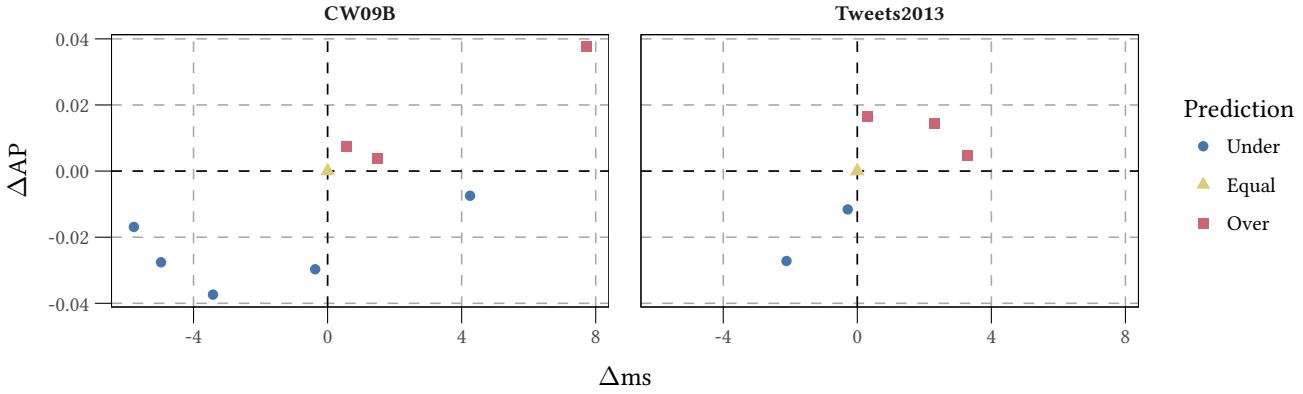

Figure 3: Changes in AP and mean latency using settings from the previous block (CW09B) or week (Tweets2013) compared to the oracle setting (when requiring *n.s.* at $p < 0.05$ compared to a non-quantized index).

[4] Matt Crane, J. Shane Culpepper, Jimmy Lin, Joel Mackenzie, and Andrew Trotman. 2017. A Comparison of Document-at-a-Time and Score-at-a-Time Query Evaluation. In *WSDM*. 201–210.

[5] Matt Crane, Andrew Trotman, and Richard O'Keefe. 2013. Maintaining Discriminatory Power in Quantized Indexes. In *CIKM*. 1221–1224.

[6] Lisa Huang. 2016. Moving Top Tweet Search Results from Reverse Chronological Order to Relevance Order. (19 Dec. 2016). Retrieved July 31, 2017 from https://blog.twitter.com/2016/moving-top-tweet-search-results-from-reverse-chronological-order-to-relevance-order

[7] Jimmy Lin and Miles Efron. 2013. Overview of the TREC-2013 Microblog Track. In *TREC*.

[8] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2014. Overview of the TREC-2014 Microblog Track. In *TREC*.

[9] Jimmy Lin and Andrew Trotman. 2015. Anytime Ranking for Impact-Ordered Indexes. In *ICTIR*. 301–304.

[10] Alistair Moffat, Justin Zobel, and Ron Sacks-Davis. 1994. Memory Efficient Ranking. *IP&M* 30, 6 (1994), 733–744.

[11] Jesus A. Rodriguez Perez, Andrew J. McMinn, and Joemon M. Jose. 2013. University of Glasgow (UoG_TwTeam) at TREC Microblog 2013. In *TREC*.

[12] Michael Persin, Justin Zobel, and Ron Sacks-Davis. 1996. Filtered Document Retrieval With Frequency-sorted Indexes. *JASIS* 47, 10 (1996), 749–764.

[13] Andrew Trotman, Xiang-Fei Jia, and Matt Crane. 2012. Towards an Efficient and Effective Search Engine. In *OSIR Workshop*.

[14] Yulu Wang and Jimmy Lin. 2014. The Impact of Future Term Statistics in Real-time Tweet Search. In *ECIR*. 567–572.

[15] Yue Wang, Hao Wu, and Hui Fang. 2014. An Exploration of Tie-breaking for Microblog Retrieval. In *ECIR*. 713–719.

[16] Chengxiang Zhai and John Lafferty. 2001. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *CIKM*. 334–342.

Acknowledgments. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, with additional contributions from the U.S. National Science Foundation under CNS-1405688.