Maintaining Discriminatory Power in Quantized Indexes

Matt Crane
Department of Computer Science
University of Otago
Dunedin, New Zealand
mcrane@cs.otago.ac.nz

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand
andrew@cs.otago.ac.nz

Richard O'Keefe
Department of Computer Science
University of Otago
Dunedin, New Zealand
ok@cs.otago.ac.nz

ABSTRACT

The time cost of searching with an inverted index is directly proportional to the number of postings processed and the cost of processing each posting. Dynamic pruning reduces the number of postings examined. Pre-calculation then quantization of term / document weights reduces the cost of evaluating each posting. The effect of quantization on precision, latency, and index size is examined herein. We show empirically that there is an ideal size (in bits) for storing the quantized scores. Increasing this adversely affects index size and search latency; decreasing it adversely affects precision. We observe a relationship between the collection size and ideal quantization size, and provide a way to determine the number of bits to use from the collection size.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software – Performance evaluation (efficiency and effectiveness)

General Terms

Experimentation, Performance, Reliability

Keywords

Quantization, Impact Ordered Indexes

1. INTRODUCTION

Search engine response time can be dominated by the time taken to calculate the score of a term with respect to a document. This can be reduced in several ways; for example by examining fewer postings (pruning), reducing the evaluation cost of the ranking function, or both. In this investigation we are interested in the second—reducing the cost of evaluation without loss of precision.

Persin *et al.* [5] proposed ordering the postings lists by decreasing term frequency (impact ordering). This not only allows those documents with the highest term frequencies to be processed first, but it also means the cost of computing the term frequency component of the ranking function can be amortized by computing it once for a set of documents sharing the same frequency. This ordering is also a form of compression as the term frequency is only stored once for a set of documents rather than once for each.

Moffat *et al.* [4] observed that approximations to components of the ranking function were as effective as exact values (the document length can be approximate). Anh *et al.* [1] observed that the term / document weight could be pre-computed and stored in the impact ordered index; during indexing they evaluated the ranking function for every term with respect to every document and stored that result rather than term frequency. Doing so reduced the search-time ranking function to a series of additions.

However, such pre-calculated values are floating point and do not compress well, so Moffat *et al.* [4] quantized these scores into b-bit integers. Several approximations were explored by Anh *et al.* [1], who show that either a linear mapping, or alternatively skewing to provide extra granularity at the lower scores works well.

Herein we examine the number of bits needed for quantization. We do this by exploring the relationship between the number of bits and retrieval precision, index size, and search latency. We find that as the number of bits increases the precision trends towards that seen in an index storing term frequencies. We observe that as the number of bits increases so too does the index size and processing time.

Problematically, the number of bits needed for quantization must be known while indexing. A value too small results in a loss of precision (*reductio ad absurdum*, 1-bit is Boolean) whereas a value too large increases latency. This leads to our research question *"What is the minimum number of bits needed for a quantized index to perform comparably to a term-frequency index"*. We measure comparable as no statistically significant loss in precision, which in turn we measure with non-interpolated MAP and P@20. P@20 accurately reflects early precision but MAP is more stable when a small number of queries with binary assessments are used [2]. We wish to minimize size so as to keep latency low.

Our question is not phrased in terms of query processing, only in terms of precision. Consequently the choice of term-at-a-time, document-at-a-time, or score-at-a-time processing of postings lists is inconsequential. Without loss of generality we use term-at-a-time processing of impact ordered postings quantized with the *uniform* quantization method of Anh *et al.* [1], which is an index-wide linear scaling of

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
Copyright 2013 ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2507860.

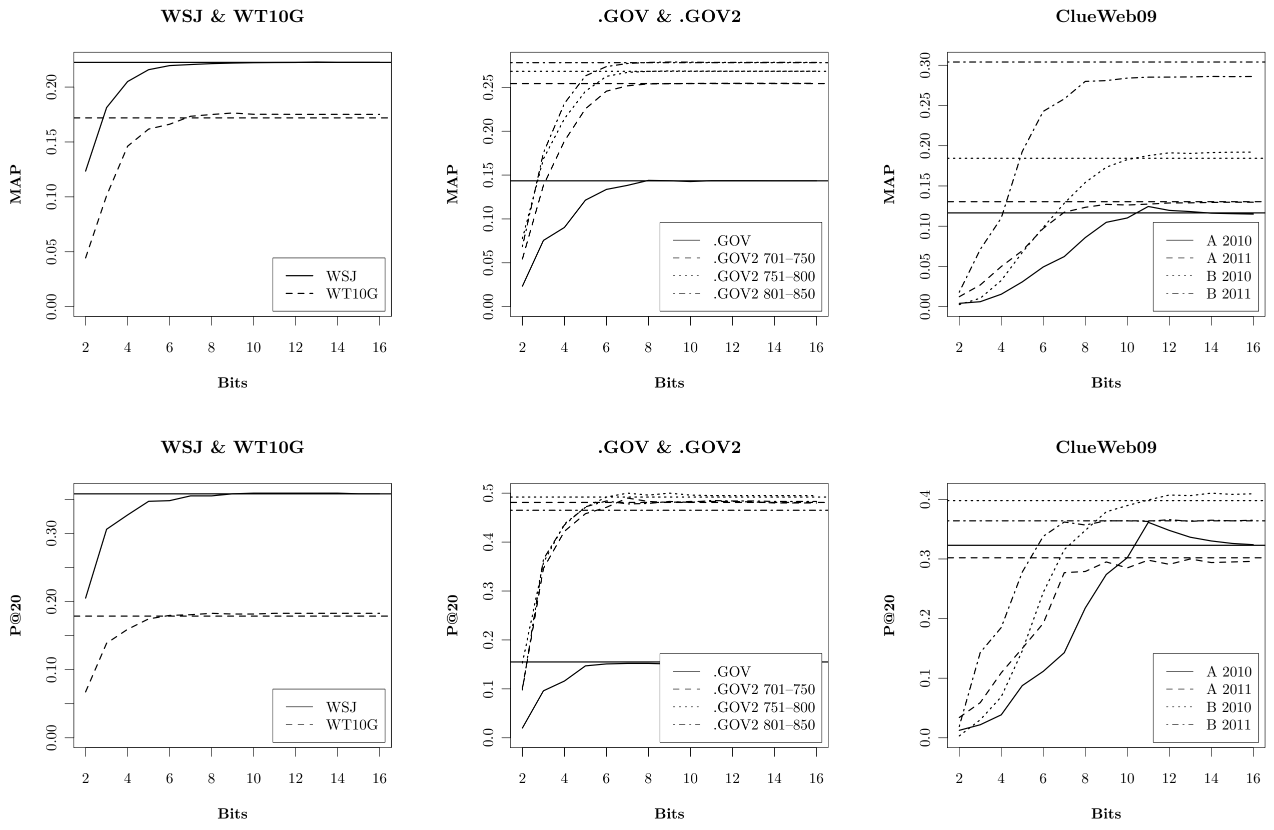

Figure 1: Precision effect due to quantization into $\left[1..2^{bits}\right)$ (MAP at top, P@20 at bottom), horizontal lines are tf indexes

the term / document weight. A heap is used to store the top-k accumulators; these were all already implemented in the ATIRE [6] search engine.

Our experiments were conducted on 5 TREC document collections of varying size and 8 query sets. From these we observe a relationship between the ideal number of bits and collection size. We use this to estimate the number of bits needed for the ClueWeb12 collection, which we find to be metric dependent.

Collection	Documents	Size	TREC Topics
WSJ	173,252	650MB	51–100
.GOV	1,247,753	19GB	551–600
WT10G	1,692,069	10GB	451–500
.GOV2	25,205,179	426GB	701–750, 751–800, 801–850
ClueWeb09 B	50,220,423	1.5TB	51–100 (2010)
ClueWeb09 A	150,954,279	12.5TB	101–150 (2011)

Table 1: Collection stats, ClueWeb09 A minus (70%) spam

2. COLLECTIONS

Table 1 presents details of the document collections used. The first column gives the name, the second gives the number of documents, the third is the size, and the fourth gives the topic sets. For example, the 1.6M document WT10G collection is 10GB and against it we used TREC topics 451–500. ClueWeb09 was used in two configurations: Category B, and Category A with 70% spam reduction (using the list developed by Cormack *et al.* [3]).

The experiments were conducted on a quad-CPU AMD Opteron 6276 2.3GHz 16-core PC with 512GB PC12800 memory, 6×600GB 10000 RPM hard drives, running Linux kernel 2.6.32. All experiments were single threaded.

3. PRECISION

In this section we show that as the number of bits used to store the quantized weight increases from 2, the precision of the search engine tends towards that observed when term frequencies are used. Indeed, when 1 bit is used the index simply stores the presence or absence of the term in the document (the index is Boolean) but the accuracy is equivalent when the resolving power is equal to that of the floating point numbers used to compute the weight at search time.

To demonstrate this we compared the precision (MAP and P@20) seen in a quantized index to that seen in a term frequency (tf) index. The number of bits used for quantization varied from 2 to 16 in steps of 1. We used the BM25 ranking function trained on maximizing ERR-IA@20 on ClueWeb09 B and TREC 2009 topics 1–50. These topics are otherwise not used herein.

The results are presented in Figure 1. Horizontal lines show the precision seen when a tf index is used and curved lines show the precision as the number of bits increases. By visual inspection, and for the smaller collections, 8 bits is adequate if MAP is used and 5 for P@20 (notwithstanding significance tests, see Section 5), but is woefully inadequate for ClueWeb09.

As a sanity test, and to eliminate the possibility that the observed behaviour is due to the precision metrics, the experiment was reconducted using just ClueWeb09 A and the diversity metric ERR-IA@20. The result, presented in Figure 2, shows a similar pattern to Figure 1.

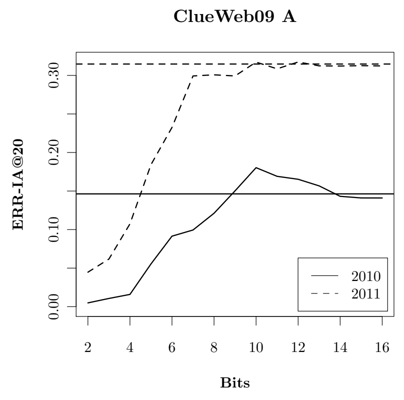

Figure 2: Precision effect (ERR-IA@20) due to fixed-range quantization

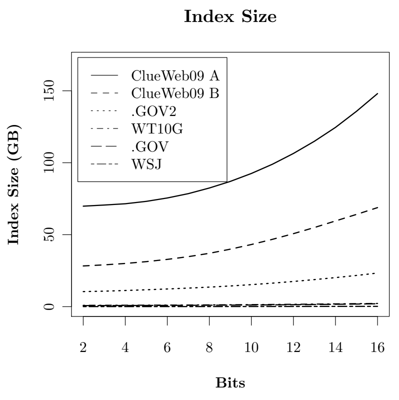

Figure 3: Index size effect due to fixed-range quantization

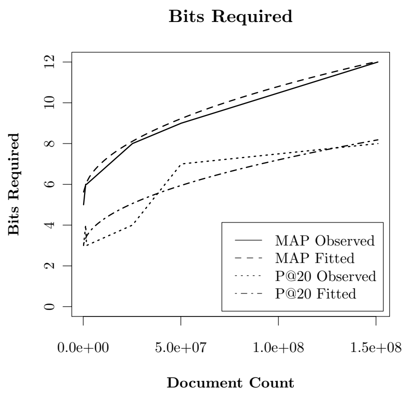

Figure 4: Collection size / bits for quantization relationship

4. SIZE

In this section we show that as the number of bits used to store the quantized score increases, so too does the index size. Indeed, this tends to infinity as some rational numbers cannot be stored in a finite number of bits (quantization is necessarily lossy).

The postings lists were impact ordered, sorted first on decreasing weight and then within each group (quantum) by increasing document id. The weight was stored once for each group and document ids were stored difference (delta) encoded then v-byte compressed. These indexes are small (for .GOV2 it is about 3% of collection size).

To conduct the experiment we examined the index sizes from Section 3. The results are presented in Figure 3 which shows that regardless of collection the index size increases as the number of bits increases. The increase can be substantial, for example the 8-bit ClueWeb09 A index is 82GB but the 16-bit index is 148GB.

This increase is intuitively correct and due to three effects. First, as the number of bits increases the space required for storing them increases. Second, as the number of bits increases the number of unique quantized values tends to a maximum of one per document id. Third, as the number of document ids in each quantum decreases, their deltas increase, and compress less effectively.

5. LATENCY

In this section we demonstrate a search latency effect due to the number of bits used for quantization. Intuitively a 1 bit index should be faster than a 16 bit index, however this is contingent on many factors including the index location (disk or memory), index structure (impact or document ordered), early termination, pruning, skipping, *etc.* But regardless of these factors an effect on latency is expected because the amount of information being processed differs. Two impact ordered approaches were tested: term-at-a-time processing to completion; and top-k ($k = 20$) using a heap.

To conduct this experiment we used the indexes from Section 3. For queries we used the titles from the TREC topics in Table 1. The index was on disk and caches were not flushed between runs. Each experiment was conducted 10 times and the median was taken because it discards outliers and more realistically (than the mean) represents expected behaviour. The baseline used term frequencies processed to completion using identical BM25 parameters. Reported times are mean time per query (from the median run).

The results are presented in Figure 5, which shows the effect of the number of bits on search time. The baseline (horizontal line) is the time taken for the tf index. The curved line is the time taken as the number of bits increases. The figure shows that, in general, as the number of bits increases so too does the time to search (latency).

It is intuitively correct that a quantized index should take less time than a tf index. In the former the ranking function is simple integer addition whereas in the latter it requires many floating point operations. It is also reasonable to expect that as the number of bits increases so too should the execution time (to a limit, as observed): the postings lists are longer as there are more integers to process. For example, with .GOV2 (751–800) and 2 bits, 308,110,592 integers were decompressed by comparison to 311,947,378 integers when 16 bits are use. But with top-k search the heap is touched less often with a larger number of bits: with ClueWeb A (2011 topics) and 2 bits it is touched 1,157,182,513 times and 2,628,019 times for 16 bits. There is a tradeoff evident in the dip seen in ClueWeb graph of Figure 5. When not performing top-k search a final sort is performed and it takes longer as the number of unique accumulator values increases.

6. IDEALITY

In the previous sections we showed an effect on precision, index size, and search latency due to the number of bits used. We wish to maximize throughput without loss of precision, and in doing so must choose the number of bits to be large enough for there to be no precision effect, but no larger.

In this experiment we re-examined the results from Figure 1 and selected, for each collection, those runs that were were not statistically significantly (2-tailed pairwise t-test, $p = 0.01$) different from the baseline. From these runs we selected for, each collection, the run that used the smallest number of bits. If this differed between topic sets we chose the larger. This requires a number of significance tests, but no adjustment (e.g. Bonferroni) was made as the purpose of the tests was selection and a small error could be tolerated.

The results are shown in Figure 4 where the relationship between the minimum number of bits needed and the col-

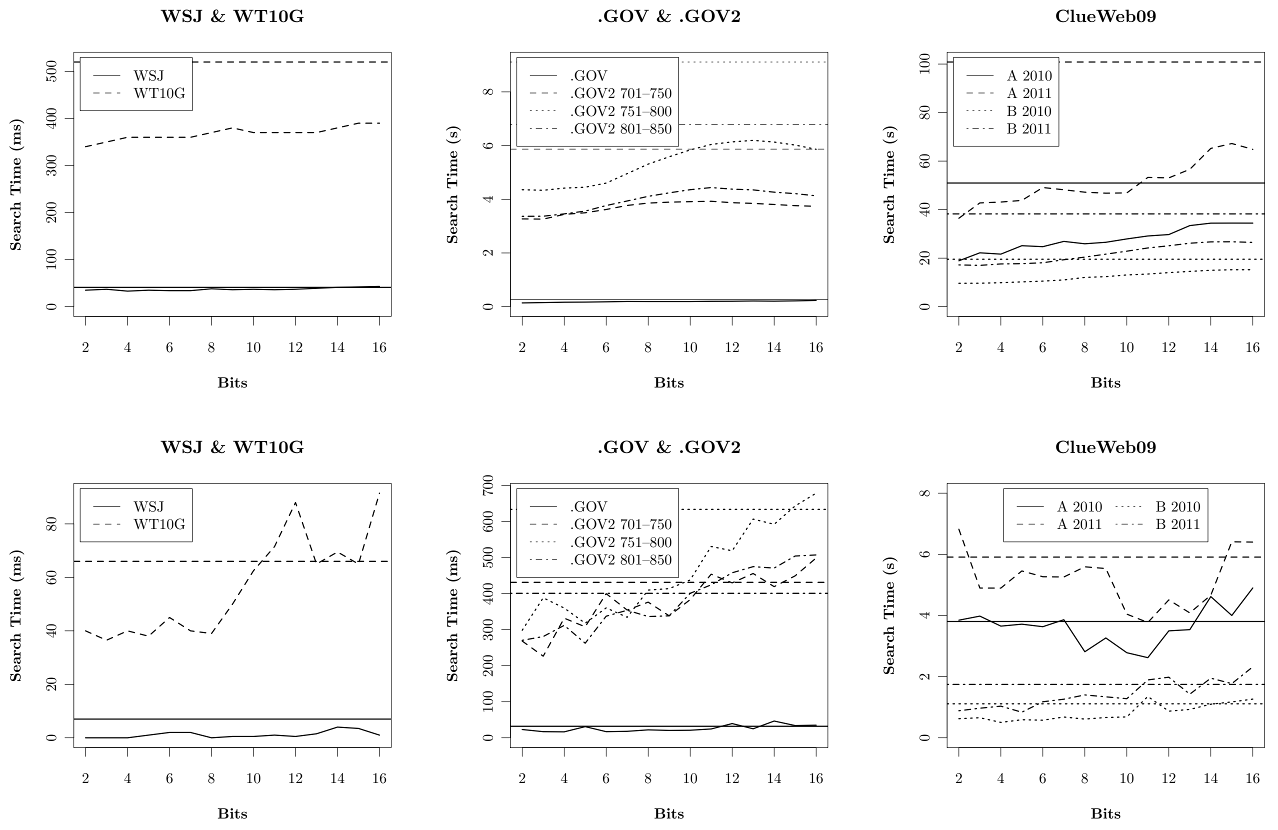

Figure 5: Search time effect due to quantization (search to completion at top, top-20 at bottom), horizontal lines are tf indexes

lection size can be seen. MAP appears to be more stable than P@20 so we fitted a curve to MAP and back fitted to P@20. This curve provides, given the collection size, a conservative estimate (lower bound, due to significance errors) of the number of bits needed without loss of precision. The lines are shown dashed and presented as Equation 1:

$$b = \left\lceil g + h \times 10^{-4} \times \sqrt{|D|} \right\rceil \quad (1)$$

where b is the number of bits, and $|D|$ is the number of documents. For MAP, $g = h = 5.4$; for P@20, $g = 2.9$, $h = 4.3$. Quantization is lossy and the errors sum when ranking and so different results are expected for substantially different query types (e.g. long).

The number of points is small so care should be taken when extrapolating. However, ClueWeb12 contains 1.2 billion documents and substituting this for $|D|$ in Equation 1 suggests the ideal number of bits to use is 25 for MAP and 8 for P@20; fewer will negatively impact precision and more will negatively affect latency and index size. Coincidently, IEEE single precision floats are 24 bit so it may be necessary to use doubles when a tf index is used (which we did).

7. CONCLUSIONS

This work presents a thorough empirical investigation into the effects on index size, search latency, and precision due to the number of bits chosen for quantization of term / document weights. We observe that quantization into 1 bit is equivalent to a Boolean index. As the number of bits increases the precision increases to a point equal to that of a term frequency index. As the number of bits increases so too does the search time and index size. We present intuitive explanations as to why. We show a relationship between the collection size and the ideal number of bits, that number that is as small as possible (minimizing latency) with no (significant) loss in precision. Our experiments suggest that ClueWeb12 will require 25 bits for quantization if MAP is the preferred metric but only 8 if P@20 is used.

8. REFERENCES

[1] V. Anh, O. de Kretser, and A. Moffat. Vector-space ranking with effective early termination. In *SIGIR 2001*, pages 35–42, 2001.

[2] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In *SIGIR 2000*, pages 33–40, 2000.

[3] G. Cormack, M. Smucker, and C. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 14(5):441–465, 2011.

[4] A. Moffat, J. Zobel, and R. Sacks-Davis. Memory efficient ranking. *IP&M*, 30(6):733–744, 1994.

[5] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. *JASIS*, 47(10):749–64, 1996.

[6] A. Trotman, X. Jia, and M. Crane. Towards an efficient and effective search engine. In *SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 40–47, 2012.