A Lossy Compression Method on Positional Index for Efficient and Effective Retrieval

Shuni Gao, Jipeng Liu, Xiaoguang Liu*, Gang Wang*
College of Computer Science, KLMDASR, Nankai University, Tianjin, China
{gaoshn,liujp,liuxg,wgzwp}@nbjl.nankai.edu.cn

ABSTRACT

In query processing, incorporating proximity between query terms is beneficial for effective retrieval. However, it brings inevitable storage and computing costs by using positional data in inverted indexes. In this paper, we propose a lossy method for compressing term position data in the case of utilizing term proximity. Our method exploits clustering property of term occurrences, adaptively clusters the nearby occurrences, and replaces the clustered positions with a centralized value. Experimental results show that our adaptive method is competitive with respect to index size, ranking efficiency and effectiveness.

CCS CONCEPTS

• **Information systems → Proximity search**; **Data compression**; **Retrieval effectiveness**.

KEYWORDS

Query Processing, Term Proximity, Positional Index Compression

ACM Reference Format:
Shuni Gao, Jipeng Liu, Xiaoguang Liu*, Gang Wang*. 2019. A Lossy Compression Method on Positional Index for Efficient and Effective Retrieval. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358125

1 INTRODUCTION

Inverted index compression has drawn lots of attention in Information Retrieval. Typically, an inverted index consists of document IDs (docIDs), in-document frequencies, and positions of term occurrences within the document. Most previous works [5, 6] have concentrated on organization and compression on docIDs.

Positional index is widely used in *term dependency* models, in which the proximity between terms is integrated into document ranking. Term dependency models achieve higher ranking accuracy than *bag-of-words* models in most cases. Meanwhile, it brings inevitable storage and computational costs with the use of position data. Moreover, positional indexes are about 3 to 5 times larger than the non-positional ones [1], thus significantly decrease system throughput. Consequently, the compression of positional index is becoming increasingly important recently. Various researches have been proposed for the compression of positional index, in a lossless way [1, 7].

The main idea of our work is lossy compression, namely information cannot be fully recovered. The technique is commonly used in image compression, which allows a certain loss of pixels. We bring lossy method into positional data compression. Different from docID compression, which need to be fully and correctly recovered for further AND/OR/WAND operation, not all positions are contributive to proximity estimation in term dependency models. Therefore, the special application allows a certain loss of positional information.

An important observation of occurrences of a term within document is the tendency of cluster, i.e., positions are crowded together. The idea in this work is that clustered occurrences can be compressed to a centralized value, which is lossy. Moreover, we hypothesize that positions in different positional index should be clustered in different granularity. As a result, we propose an adaptive-granularity method when clustering term positions.

In the remainder of this paper, we introduce related work in Section 2, review term dependency models in Section 3, and describe the proposed adaptive lossy compression approach in Section 4. Experimental results are presented in Section 5, and we suggest some ideas for future work in Section 6.

2 RELATED WORK

The idea of *approximate positional index* was first proposed by Elsayed et al. in [3]. They divided documents into coarse-grained buckets and represented each bucket with its ID. They introduced two approaches to dividing documents: fixed-width, with fixed bucket size for each document; and variable-width, which divided each document into a fixed number of buckets. The methods achieved smaller positional indexes and less query execution time, but uncompetitive retrieval effectiveness compared with uncompressed indexes. The weakness is that all the documents are divided by single granularity (a fixed bucket-width or a fixed number of buckets). A position list records the occurrences of a term within a document, which means both document and term information should be considered, rather than just document information. In our work, granularity is adaptively determined for each position list according to the term and document it belongs to. A series of experiments on methods proposed in [3] are conducted as baselines in Section 5. To the best knowledge of the authors, there is no other work that introduce lossy-compressed positional index.

3 TERM DEPENDENCY MODEL

Metzler and Croft [4] proposed sequential dependence model (SDM) and full dependence model (FDM), which model term dependencies

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358125

via Markov random fields. SDM incorporates sequential dependence of adjacent terms, while FDM incorporates full dependence of all term pairs in a query. The two dependency models use the same form of scoring function below.

$$S(\mathbf{q}, \mathbf{d}) := \sum_{t \in \mathbf{q}} \lambda_T f_T(t, \mathbf{d}) + \sum_{<t_i, t_j> \text{ in } \mathbf{q}} \left(\lambda_O f_O(t_i, t_j, \mathbf{d}) + \lambda_U f_U(t_i, t_j, \mathbf{d})\right),$$

where the second sum is over consecutive term pairs in SDM and all term pairs in FDM. f_T, f_O, and f_U are feature functions for unigrams, ordered sequences of term pair, and term pairs co-occurring within a window, respectively. λ_T, λ_O, and λ_U are the corresponding weighting parameters.

4 THE PROPOSED METHOD

In this Section, we first introduce the procedure of generating lossy-compressed positional data, then we introduce how we decide the lossy granularity (threshold) for each position list.

Algorithm 1 Adaptive Lossy Compression Method.

Input:
$pl = \{p_0, p_1, \ldots, p_n\}$, a sorted position list containing the positions a term appears in a document
$threshold$, the lossy granularity for the position list
Output:
$lossy\_position\_list$, the lossy-compressed positions

```
c = {p_0}
for i = 1, 2, ..., n do
    p = c.last() /* the last number in c */
    if p_i − p < threshold then
        c.add(p_i)
    else
        s = c.centroid()
        lossy_position_list.add(s)
        c = {p_i}
    end if
end for
```

Algorithm 1 shows the procedure of generating lossy-compressed position list. We first cluster the nearby positions by their gaps, and then use a centralized position (centroid) to represent clustered positions. The centroid of a cluster is calculated by first summing up all values in the cluster, and then divided by cluster size.

Table 1 gives an example of our adaptive lossy compression method and Var(64) proposed in [3]. In this example, the clustering threshold of our method is 11. Var(64) equally divides the document (653 words) into 64 buckets and represents each bucket with its ID. We see that Var(64) scales positions into a smaller range, while our centroid-based representation retains the approximate position of a cluster and keep the overall distribution of all clusters.

Table 1: An example of adaptive lossy compression.

Uncompressed	2, 10, 17, 22, 66, 71, 82, 93, 100, 125, 561, 641, 643
Adaptive(11)	12, 68, 82, 96, 125, 561, 642
Var(64)	0, 1, 2, 6, 8, 9, 12, 54, 62, 63

The key point of our method is to adaptively choose appropriate granularity (clustering threshold) for each position list. Since a position list is the positions that a term occurs in a document, we assume that the threshold should be determined by two factors: the importance of the term and the length of the document. Based on the above consideration, we propose formula (1)-(3) to determine the threshold using Inversed Document Frequency(IDF) and document length. IDF is commonly used to evaluate the importance of a term. $IDF(t) = log(N/N_t)$, where N_t is the number of documents in which the term t appears, and N is the number of documents in the whole collection.

$$document\_factor = [log_{10}(document\_length)]^a \quad (1)$$

$$term\_factor = \frac{IDF}{b} + c \quad (2)$$

$$threshold = \frac{document\_factor}{term\_factor} \quad (3)$$

Document length varies from tens to tens of thousands, so we first scales document length by logarithmic and power function. Parameter a is used to adjust the influence of the two functions in formula (1). IDF is linearly transformed to a reasonable range in formula (2). Formula (3) combines the two factors together. Intuitively, it is better to keep more positional information if the position list belongs to an important term. That is to say, the more important a term is, the smaller its threshold should be. Consequently, threshold is inversely proportional to IDF.

a, b, c are tunable parameters, which are trained by hill climbing search to directly optimize mean average precision. The parameters are trained by starting at $a = b = 1$, $c = 0$. We sequentially split 10% of query set for parameter training. In our experiments, a, b, c is 3, 4, 0.5 respectively.

5 EXPERIMENTAL RESULTS

5.1 Experimental setup

We use two query sets, which is the Million Query 2007 (MQ2007) Million Query 2008 (MQ2008). The Million Query track used GOV2 collection of documents. The two query sets consists of queryID, docID and <queryID, docID> relevance judgment. We filter out invalid queries that have no relevant document. Information of dataset is listed in Table 2.

Table 2: Dataset information.

	MQ2007	MQ2008
#query	1454	564
#doc	59559	12102
#position list	190386	46399

Indexes were constructed by Lucence. Experiments are performed on a server with Intel Xeon processors (E5-2603 1.6GHz), 256GB RAM. All experiments are performed on a single core.

We choose three types of baselines, which is uncompressed, lossless-compressed and lossy-compressed positional index. For lossless method, we choose Simple16 and VByte, which are commonly used in inverted index compression[2]. For lossy method, we choose previous solutions that split each document into buckets and represent each bucket with its ID. Variable-width method with buckets number b is denoted as Var(b), and fixed-width method with bucket size w is denoted as Fixed(w). The value of b and w is in consistency with the experiment in [3]. Experiments are conducted

by applying SDM and FDM retrieval model on the positional data generated by baselines and our method. For fair comparisons, we tune parameter settings to optimize retrieval effectiveness for both baselines and our method.

Experimental results are evaluated in terms of index size, ranking efficiency and effectiveness[1].

5.2 Index Size

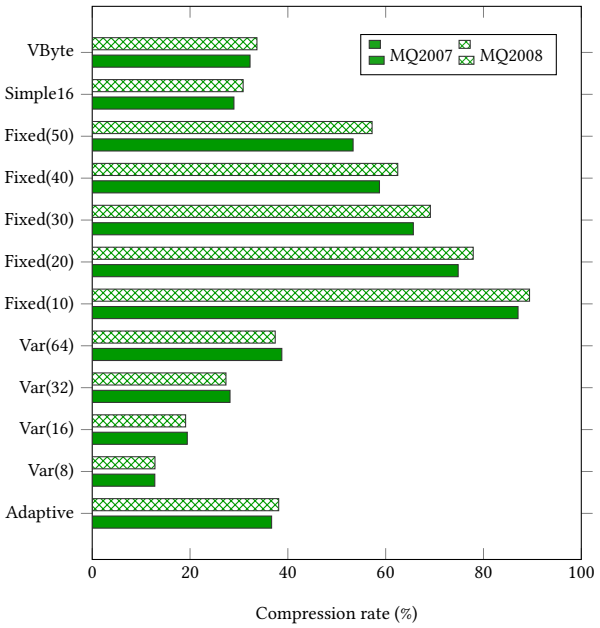

Figure 1: Compression rate of baselines and our method.

Figure 1 shows the compression rate of baselines and our adaptive method. Bars plot compression rate, thus shorter is better. We observe that the compression rate of our method is close to lossless compression method Simple16 and VByte. Among all lossy baselines, Var achieve smaller indexes than Fixed. As expected, index size increases with the growth of bucket number b and the decline of bucket size w respectively. The compression rate of our method is smaller than Fixed and bigger than Var. The two query sets exhibit similar results on compression rate.

A secondary compression by coding bucket ids was performed in [3], which is not included in our experiment. We only evaluate the lossy-compression procedure.

5.3 Ranking Efficiency

In efficiency experiments, we collect the time to compute term proximity, which is when position data are being used. Figure 2 illustrates the averaged time of the proposed method and baselines. The MQ2007 and MQ2008 query sets exhibit similar results, so

[1]We do not compare the results of lossless methods in efficiency and effectiveness experiments for space reasons. In terms of efficiency, lossless compressed index is not competitive compared with uncompressed index because of the decompressing procedure. In terms of effectiveness, lossless compressed index yields the same ranking accuracy with uncompressed one.

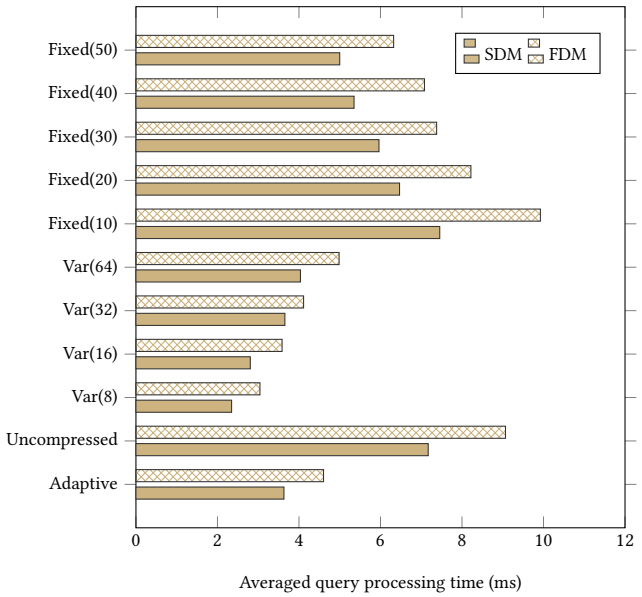

Figure 2: Averaged execution time of baselines and our method.

we only plot results on MQ2007 for space reasons. As expected, execution time is positively correlated with size of positional index. Fewer positions lead to less calculation, thus Var(8) consumes the least execution time. Our adaptive method achieves comparable time to Var, and less time than Fixed and uncompressed strategies.

We also observe that Fixed(10) consumes more time than the uncompressed strategy, although the data size of Fixed(10) is smaller than the uncompressed one. As introduced in Section 3, new feature functions are proposed for Fixed and Var, which include additional calculations on buckets. As a result, the efficiency of computing term proximity is hindered. Compared with Var and Fixed that represent positions by bucket-ids, our centroid-based representation retains the approximate position of a cluster so that the overall distribution of all clusters is kept. As a result, our strategy can be directly used in dependency models without changing feature functions.

5.4 Ranking Effectiveness

Ranking Effectiveness is measured by mean precision at 1 (MP@1) and mean average precision (MAP). The results are listed in Table 3. Significant tests are performed between the proposed method and all baselines.

In Table 3, we observe that the proposed adaptive strategy consistently achieves the highest ranking accuracy. Among the baselines, Fixed outperforms the uncompressed and Var strategies in most cases. It seems that Var is uncompetitive compared with the uncompressed one because it hurts ranking accuracy sometimes.

When we examine overall performance, we find that Var(8) achieves the smallest positional index, the least execution time, but the worst ranking accuracy. In contrast, Fixed(10) achieves the largest positional index, the most execution time, but better ranking accuracy. This is expected that smaller positional index keeps fewer

Table 3: Ranking accuracy (MAP, MP@1) of baselines and the proposed method. Bold, $^+$ and * indicates statistically significant improvements of adaptive method over the uncompressed, Var, and Fixed strategy respectively, according to the Fisher's randomization (p<0.05). Percentage improvements of adaptive method over the uncompressed baseline are shown in parenthesis.

	MQ2007				MQ2008			
	FDM		SDM		FDM		SDM	
	MP@1	MAP	MP@1	MAP	MP@1	MAP	MP@1	MAP
Uncompressed	0.5107	0.5389	0.4962	0.5268	0.5922	0.6561	0.5869	0.6513
Var(8)	0.5107	0.5247	0.4931	0.5176	0.5975	0.6534	0.6046	0.6501
Var(16)	0.5076	0.5307	0.5038	0.5240	0.6152	0.6573	0.6046	0.6550
Var(32)	0.5084	0.5351	0.5084	0.5255	0.5993	0.6557	0.5833	0.6460
Var(64)	0.4985	0.5342	0.4893	0.5219	0.6011	0.6563	0.5887	0.6468
Fixed(10)	0.5268	0.5427	0.4977	0.5247	0.5922	0.6614	0.5887	0.6534
Fixed(20)	0.5145	0.5419	0.4962	0.5292	0.5904	0.6614	0.5816	0.6526
Fixed(30)	0.5145	0.5435	0.4901	0.5297	0.5851	0.6608	0.5922	0.6557
Fixed(40)	0.5245	0.5437	0.4985	0.5314	0.5957	0.6592	0.5993	0.6572
Fixed(50)	0.5321	0.5471	0.5115	0.5361	0.5975	0.6601	0.6046	0.6569
Adaptive	**0.5329**$^+$ (+4.35%)	**0.5475**$^+$ (+1.60%)	**0.5260**$^{+*}$ (+6.00%)	**0.5390**$^{+*}$ (+2.32%)	**0.6401** (+8.09%)	**0.6778**$^{+*}$ (+3.31%)	**0.6312** (+7.55%)	**0.6730**$^{+*}$ (+3.33%)

positions, resulted in less calculation and losing more positional information, while larger index is on the contrary. Compared with Var(64) which is comparable to our method in terms of data size and execution time, our method achieves improvements of up to 7.5% in MP@1 and 4% in MAP. Overall, the proposed adaptive lossy-compression method achieves the best balance on data size, ranking efficiency and effectiveness.

We delve deeper into the reason that our method achieves better effectiveness than uncompressed and other strategies. As we introduced, term dependency models achieve higher ranking accuracy in most cases. Meanwhile, utilizing term proximity inevitably brings noise into document ranking. Previous research [8] held similar opinion that utilizing term proximity was not suitable for all queries. The high ranking accuracy of our method indicates that our approach reduces noise brought by term proximity to a certain extent while retaining important positional information.

As for the two dependency models used in this paper, we see that FDM consumes more time (figure 2), but consistently yields better retrieval effectiveness (table 3). It can be explained that SDM only incorporates proximity of adjacent term pairs of a query, while FDM incorporates all term pairs of a query. Consequently, FDM leverages more proximity features of a query, thus achieves better effectiveness. Meanwhile, FDM is inherently more computationally complex than SDM.

6 CONCLUDING REMARKS

Utilizing term dependence is beneficial for effective retrieval. However, it brings inevitable storage and computing costs by using positional data of inverted indexes. In this paper[2], we hypothesize that not all exact positions have to be stored. Motivated by the clustering property of term positions, we design a lossy compression method for positional index that replaces clustered positions with a centralized value. Moreover, clustering granularity is adaptively determined by term importance and document length.

Experiments are carried by two dependency models on two query sets. Experimental results show that the proposed method achieves the highest ranking accuracy, with competitive index size and execution time. Moreover, further analysis on effectiveness indicates that the proposed method reduces noise brought by term proximity to some extent while retaining important positional information.

In the future, more methods are going to be tried to optimal the formulas. Second, we will try to apply our method into real world applications. Moreover, the proposed lossy method can be further applied to other filed involving position data, such as passage retrieval.

[2]This work is partially supported by NSFC (61872201, 61702521, 61602266, U1833114); STDPof Tianjin (17JCYBJC15300, 16JCYBJC41900, 18ZXZNGX00140, 18ZXZNGX00200); and Fundamental Research Funds for Central Universities, TKLNDST.

REFERENCES

[1] Leonidas Akritidis and Panayiotis Bozanis. 2012. Positional Data Organization and Compression in Web Inverted Indexes. In *Database and Expert Systems Applications - 23rd International Conference, DEXA 2012, Vienna, Austria, September 3-6, 2012. Proceedings, Part I.* 422–429.

[2] Matteo Catena, Craig Macdonald, and Iadh Ounis. 2014. On Inverted Index Compression for Search Engine Efficiency. In *Advances in Information Retrieval - 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings.* 359–371.

[3] Tamer Elsayed, Jimmy J. Lin, and Donald Metzler. 2011. When close enough is good enough: approximate positional indexes for efficient ranked retrieval. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011.* 1993–1996.

[4] Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *Proc. SIGIR.* 472–479.

[5] Alistair Moffat and Lang Stuiver. 2000. Binary Interpolative Coding for Effective Index Compression. *Inf. Retr.* 3, 1 (2000), 25–47.

[6] Giuseppe Ottaviano and Rossano Venturini. 2014. Partitioned Elias-Fano indexes. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014.* 273–282.

[7] Hao Yan, Shuai Ding, and Torsten Suel. 2009. Compressing term positions in web indexes. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009.* 147–154.

[8] Ju Yang, Jiancong Tong, Rebecca J. Stones, Zhaohua Zhang, Benjun Ye, Gang Wang, and Xiaoguang Liu. 2016. Selective Term Proximity Scoring Via BP-ANN. *SIGIR Neu-IR* abs/1606.07188 (2016).