Speeding up Similarity Search by Sketches

Vladimir Mic(✉), David Novak, and Pavel Zezula

Masaryk University, Brno, Czech Republic
xmic@fi.muni.cz

Abstract. Efficient object retrieval based on a generic similarity is one of the fundamental tasks in the area of information retrieval. We propose an enhancement for techniques that use the distance-based model of similarity. This enhancement is based on sketches–compact bit strings compared by the Hamming distance which represent data objects from the original space. The sketches form an additional filter that reduce the number of accessed data objects while practically preserving the search quality. For a certain class of state-of-the-art techniques, we can create the sketches using already known information, thus the time overhead is negligible and the memory overhead is subtle. According to the presented experiments, the sketch filtering can reduce the number of accessed data objects by 60–80 % in case of *M-Index*, and 30 % in case of *PPP-Codes* index while hurting the recall by less than 0.4 % on 10-NN search.

1 Introduction

Similarity retrieval represents a fundamental challenge of modern data processing. Handling objects according to their mutual similarity closely corresponds to the human perception of reality [6], therefore similarity retrieval can provide a natural way to access various types of data. Independently of a specific measure of similarity, we focus on the efficient similarity-based retrieval in large data collections. We adopt the broad model of the *metric space* that considers a data domain D together with a distance function $d : D \times D \mapsto \mathbb{R}$ to express the dissimilarity of two data objects from domain D. The distance function must satisfy properties of non-negativity, identity, symmetry and triangle inequality [19]; the triangle inequality is not explicitly utilized by the proposed technique. We assume an approximate evaluation of k-NN queries. The approximation quality is measured by *recall*, i.e. the relative size of the intersection of the approximate k-NN answer with the precise one. In our case the recall is equal to precision. We focus on speeding up existing distance-based similarity indexes. Majority of these indexes break the data collection down into disjoint partitions that are supposed to contain data objects mutually similar. Given a query object $q \in D$, the most promising partitions are identified; the union of data objects in these partitions forms a *candidate set* for the query q. Data objects x from the candidate set are accessed and distances $d(q, x)$ are evaluated to return the k most similar objects; this phase is further denoted as *refinement* of the candidate set. In high dimensional spaces, majority of data objects in the candidate set

L. Amsaleg et al. (Eds.): SISAP 2016, LNCS 9939, pp. 250–258, 2016.
DOI: 10.1007/978-3-319-46759-7_19

are usually non-relevant [15], but these are uncovered only during the generally expensive refinement phase.

Objectives, Approach and Related Work

We propose to enrich a generic indexing technique with compact bit strings, called *sketches*, for all data objects and use them to reduce the candidate set. In order to be effective, this additional filter is supposed to work on a different space partitioning principle. Experiments with two indexes on two high-dimensional datasets show that the effect of the sketch filtering can be radical.

The basic reasoning behind our proposal is the following:

– The quality and efficiency of the approximate similarity search is closely related to the candidate set, i.e. the set should contain all relevant data objects and be as small as possible. By filtering out non-relevant data objects from the set, we can save the evaluations of the distance function d (which can be expensive) during the refinement phase and, if the data objects are stored on the disk, reduce the I/O costs.
– Many successful indexing techniques use a static set of reference objects (*pivots*) and the distances between each data object o and each pivot are evaluated during the preprocessing phase [1,5,12,15,17]. In these cases, we can create sketches using the already known object-pivot distances (see Sect. 2 for details). Then the time overhead introduced by the additional sketch filtering is negligible and the memory overhead subtle in comparison with the potential gain.

Combining different space partitionings is not a new idea and it seems to be a viable approach to fight the curse of dimensionality. A typical approach is to apply the same space partitioning principle but with different randomized parameters (e.g. sets of pivots) [5,10,13,15]. The data objects are either kept in memory and only accessed "from different points of view" [10], replicated on the disk [5,13], or kept separately from the memory indexes on an SSD disk [15]. There are also techniques that combine completely different principles of space partitioning into a single index, for instance the Pivoting M-Tree [16]. We do not know about any work that would propose a secondary filtering for approximate similarity search that would use the same pivot set for a different partitioning.

Section 2 of this paper describes the proposed approach and its properties in detail. Section 3 contains evaluation of the effectiveness of the sketch filtering on two indexes M-Index and PPP-Codes (Sect. 3.1) and on two datasets (CoPhIR and DeCAF – Sect. 3.2). The paper is concluded in Sect. 4.

2 Bit-String Sketches for Candidate Set Reduction

The sketch of data object $o \in D$ is a bit string in Hamming space which approximates the location of o in the original space (D, d). In particular, each bit value of $sketch(o)$ limits a subspace in (D, d) where the object o is located by means of

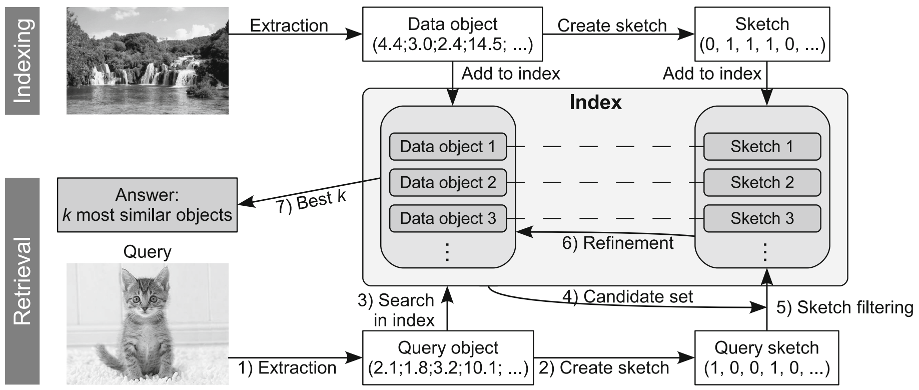

Fig. 1. Similarity search with additional sketch based filtering

a certain space partitioning. A typical example is a *generalized hyperplane partitioning* (GHP) which divides the data objects to two parts by means of their distances from two selected pivots $p_1, p_2 \in D$ [8,11,19]. Several authors study the similarity search based on sketches [4,8,11,18], and they usually achieve promising results for different data types, data dimensions, and distance functions.

In our previous paper [8], we analysed three properties of sketches desirable for efficient similarity search purely based on sketches. Having a set of sketches created for a given dataset, we have shown that (1) bit values of each $sketch(o)$ should strongly depend on the position of object o in space (D, d), (2) the sketches should have *balanced bits*, i.e. each bit should be set to 1 in one half of the sketches, and (3) bits of sketches should be mutually as low correlated as possible. The first requirement is satisfied e.g. by sketches created using the GHP partitioning and other requirements can be satisfied by proper selection of specific pivots for GHP.

In this paper, we propose to use such sketches to enrich practically any distance-based search indexing technique. In particular, we propose to maintain a $sketch(o)$ for each indexed data object o and use these sketches to reduce the candidate set to be refined (see Fig. 1). Given a query object q, the $sketch(q)$ is created and we filter out some candidate objects o based on Hamming distance between $sketch(q)$ and $sketch(o)$. Either, we can remove all data objects o with the sketch distance higher than some threshold, or we can filter out a given percentage of the candidate objects with the highest sketch distances. The distance threshold approach can be applied already during the primary candidate set generation. On the other hand, the percentage to be filtered can be determined without any knowledge about the Hamming space. According to our rigorous testing, the results of these approaches are comparable and thus we present only results of the second one which are slightly better.

The expenses of the sketch filtering are the following: (1) CPU time to investigate pivot pairs suitable for the sketches; this step is performed during the preprocessing. (2) CPU time to obtain $sketch(o)$ for each object o and query sketch *sketch*(q); using the GHP partitioning, this means evaluation of $2 \cdot b$ distances to pivots, where b is the sketch length; in indexes with a fixed set of pivots all the object-pivot distances are often known. In these cases sketches are created practically for free. (3) CPU time to evaluate the Hamming distances; this operation is very efficient on modern CPUs in comparison with often expensive evaluations of distance d. (4) Memory overhead of keeping *sketch*(o) for each data object o; we show that even short sketches with lengths $b = 32$ or $b = 64$ can be very effective.

In general, the additional filtering can be employed in the following two ways. Either we can (1) reduce the number of refined data objects while, in ideal case, preserving the search quality (recall), (2) or we can preserve the number of refined data objects by adding more data objects instead of the ones filtered out by sketches; in this way, we can improve the search quality. Results presented in Sect. 3 can be interpreted in both ways.

3 Evaluation

In this section, we present evaluation of the similarity search with the additional sketch filtering. We conducted experiments using two different indexing techniques (see Sect. 3.1) and two real-life datasets (see Sect. 3.2). The sketch filtering was evaluated using sketches of lengths 32 and 64 bits and the results are presented and discussed in Sect. 3.3.

3.1 Similarity Indexes

The first index structure is the M-Index [12]. It uses a fixed set of pivots to perform Voronoi partitioning and each Voronoi cell is recursively partitioned by the same principle. The depth of this partitioning is determined dynamically according to occupation of the leaf partitions. Each data object is then stored according to its several closest pivots (according to prefix of a *pivot permutation*). Given a query object, the M-Index uses the query-pivot distances to determine the most promising partitions to contain query-relevant data objects [12].

The PPP-Codes search structure [14,15] has a slightly different architecture. It maintains a memory index which determines the set of candidate objects by their unique IDs and this set is retrieved from a disk storage (SSD) and refined. The space partitioning is also based on pivot permutation prefixes (PPP) but the candidate set can be an order of magnitude smaller than for M-Index [15]. The effect is achieved by decomposing the pivot set into several sets and thus creating several PPPs for each data object. Given a query object, the candidate set is determined by a selective combination of candidate sets from individual pivot spaces. In this way, PPP-Codes can appropriately filter out objects that seem to be relevant in one pivot space but the other spaces indicate the opposite. The process of candidate set formation is more computationally demanding.

3.2 Testing Data

The experiments are conducted on two real-life data collections, both consisting of visual descriptors extracted from images. The first set is formed by *DeCAF* descriptors [3] – 4096-dimensional vectors taken as an output from the last hidden layer of a deep convolutional neural network [7]. These descriptors were extracted from a 1M subset of the *Profiset collection*[1]. The DeCAF descriptors are compared by the Euclidean distance to form the metric space (D, d).

The second dataset consists of a combination of five MPEG-7 visual descriptors [9] as provided by the CoPhIR[2] data collection [2]. Each of these descriptors is accompanied with a suitable distance function [9] and the descriptors extracted from each image are combined into a single metric space (D, d) by a weighted sum of individual distances [2]. In total, this representation can be viewed as a 280-dimensional vector. We take a 1M subset of CoPhIR.

For each dataset, we have randomly selected a set of 512 pivots that are used by M-Index and PPP-Codes for indexing. In order to create sketches, we investigate all $\binom{512}{2}$ pivot pairs. Each pair has been used to partition a random subset of 100,000 data objects by GHP and in this way we have identified those pairs that divide the data into parts balanced at least 55 % to 45 %. From these balanced pivot pairs (≈8,000) we further select those producing sketches with low correlated bits using a heuristics described in [8]. Both 1M subsets of datasets, query objects and pivots were selected randomly and are publicly available.

3.3 Results

In this section, we evaluate the ability of the sketches to filter out non-relevant data objects from the candidate set. In the first set of experiments, we let the M-Index identify the candidate set with the 50,000 most promising data objects for a representative query object q; such set is denoted *M50K*. Let us first observe the distribution of distances $d(q, x)$ for all 50 K data objects x in this candidate set. This distribution is denoted dd_{M50K} and is depicted by the black curve in Fig. 2. Further, we denote *M50K,S50 %* the same candidate set with 50 % data objects filtered out by sketches (according to Hamming distances from $sketch(q)$); the respective query-object distance distribution $dd_{M50K,S50\,\%}$ is also depicted in Fig. 2. Please, note that these plots are taken for a single query on the DeCAF dataset and that both plots are the same, just the right one uses a logarithmic scale of axis y. The y axis expresses the frequency, formally:

$$\int_{-\infty}^{\infty} dd_{M50K}(x)\,dx = 1, \text{ therefore: } \int_{-\infty}^{\infty} dd_{M50K,S50\,\%}(x)\,dx = 0.5.$$

This example illustrates the ability of the sketches to preserve relevant data objects, since the beginnings of both curves are the same (see marked point from which these curves differs). In particular 499 out of 500 data objects with the smallest distances to the query object are preserved in this example.

[1] http://disa.fi.muni.cz/profiset/.
[2] http://cophir.isti.cnr.it/.

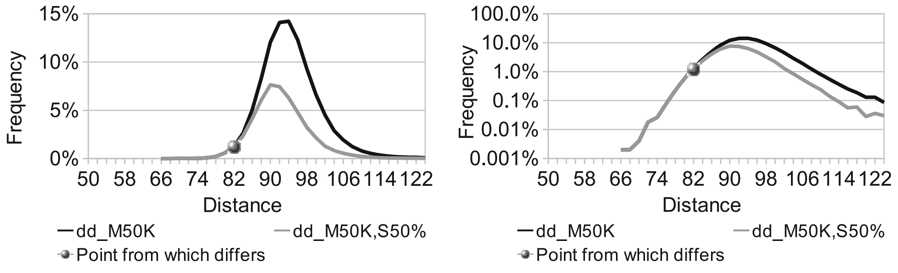

Fig. 2. Distance distributions on data objects M50K and M50K,S50 %, left plot with linear scale and right plot with logarithmic scale

In the rest of this section, we present results averaged over 1,000 randomly selected queries. We focus on the probability of false negatives, i.e. that the actual kth nearest neighbour from the candidate set is filtered out by sketches. Figure 3 depicts these probabilities for the DeCAF dataset, M50K data objects and filtering out 50 % of these data objects by sketches. Results for 32 bit and 64 bit sketches are presented. The genuine results are depicted in a gray color and the trend curves are in black. The results show significantly better ability of 64 bit sketches over 32 bit to preserve the most similar data objects in an answer.

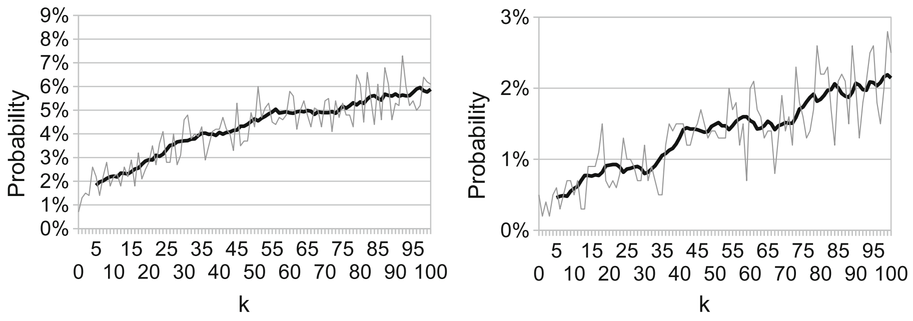

Fig. 3. Chances of omitting the kth most similar object by additional sketch filtering, left plot with 32 bit sketches and right plot with 64 bit sketches

In the final set of experiments, we use 64 bit sketches and we focus on overall k-NN recall for $k = 10$ (denoted as recall@10). Figures 4 and 5 show results for CoPhIR and DeCAF datasets, respectively, always for candidate sets from both M-Index and PPP-Codes. The vertical axes represent the average recall@10 after the sketch filtering. The horizontal axes show the relative size of the candidate set with respect to the size of dataset (which is 1 million). Individual curves correspond to percentages of this candidate set filtered out by the sketches (curves *sketch filter 0 %* show original results of M-Index and PPP-Codes).

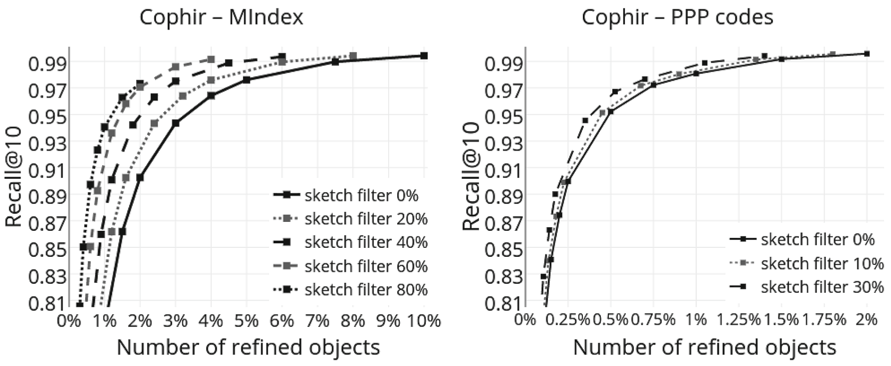

Fig. 4. Recall of double filter and refine similarity retrieval on CoPhIR dataset

We can read these graphs in two ways: (1) For a given number of refined objects, we can observe which combination of original candidate set size and percentage of sketch filtering gives the highest recall, and (2) for a required recall level, we can look for parameters leading to the smallest number of refined objects. For instance, for recall below 0.95, the most efficient is letting the sketches filter out even 80 % of the M-Index candidates and over 30 % in case of PPP-Codes. For a highly accurate retrieval with recall about 0.99, it is better to use a bigger original candidate set and filter out about 50–60 % in case of M-Index and 30 % objects for PPP-Codes. Let us observe specific selected values for DeCAF (Fig. 5): The pure M-Index must refine 100,000 objects to achieve recall 97.57, while the sketches can filter out all but 40,000 objects and preserve average recall of 97.21. In case of PPP-Codes, only 14,000 objects instead of 20,000 have to be refined while the recall value decreases from 97.32 to 97.13.

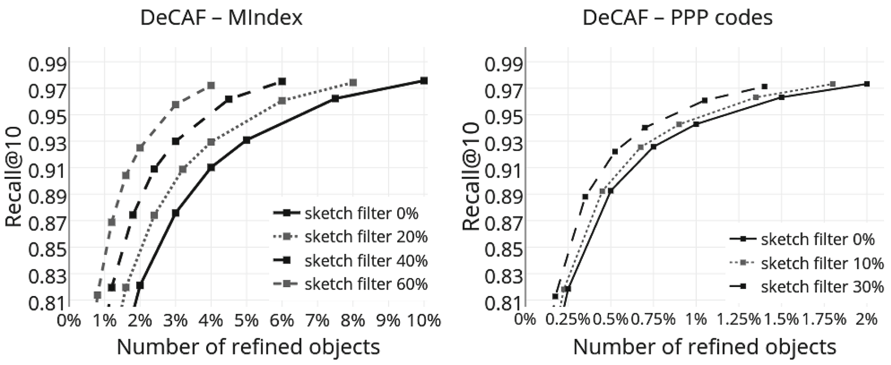

Fig. 5. Recall of double filter and refine similarity retrieval on DeCAF dataset

4 Conclusions

We have proposed and evaluated an enhancement of traditional similarity search techniques with an additional sketch-based filtering. Sketches, compact binary strings, can be created practically for free for techniques that use a static set of pivots. Their contribution to the quality of filtering can be huge, as shown on two state-of-the-art indexes M-Index and PPP-Codes and two real-life datasets. We have demonstrated the ability of the sketches to filter out many non-relevant data objects while preserving almost all relevant ones. In case of M-Index, the number of refined data objects can be reduced by 60–80 % and in case of PPP-Codes by 30 % while the decrease of the recall@10 was only negligible.

Acknowledgements. This work was supported by the Czech Science Foundation project GA16-18889S.

References

1. Amato, G., Gennaro, C., Savino, P.: MI-File: using inverted files for scalable approximate similarity search. Multimedia Tools Appl. **71**(3), 1333–1362 (2014)
2. Batko, M., Falchi, F., Lucchese, C., Novak, D., Perego, R., Rabitti, F., Sedmidubsky, J., Zezula, P.: Building a web-scale image similarity search system. Multimedia Tools Appl. **47**(3), 599–629 (2010)
3. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: DeCAF: a deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 (2013)
4. Dong, W., Charikar, M., Li, K.: Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In: Proceedings of ACM SIGIR 2008, pp. 123–130. ACM (2008)
5. Esuli, A.: Use of permutation prefixes for efficient and scalable approximate similarity search. Inf. Process. Manage. **48**(5), 889–902 (2012)
6. Kemler, D.G.: Classification in young and retarded children: the primacy of overall similarity relations. Child Dev. **53**(3), 768–779 (1982)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
8. Mic, V., Novak, D., Zezula, P.: Improving sketches for similarity search. In: Proceedings of MEMICS 2015, pp. 45–57 (2015)
9. MPEG7: Multimedia content description interfaces. part 3: Visual (2002)
10. Muja, M., Lowe, D.G.: Scalable nearest neighbour algorithms for high dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **36**(11), 1–14 (2014)
11. Muller-Molina, A.J., Shinohara, T.: Efficient similarity search by reducing i/o with compressed sketches. In: Proceedings of SISAP 2009, pp. 30–38. IEEE Computer Society (2009)
12. Novak, D., Batko, M., Zezula, P.: Metric index: an efficient and scalable solution for precise and approximate similarity search. Inf. Syst. **36**(4), 721–733 (2011)
13. Novak, D., Zezula, P.: Performance study of independent anchor spaces for similarity searching. Comput. J. **57**(11), 1741–1755 (2014)

14. Novak, D., Zezula, P.: Rank aggregation of candidate sets for efficient similarity search. In: Decker, H., Lhotská, L., Link, S., Spies, M., Wagner, R.R. (eds.) DEXA 2014. LNCS, vol. 8645, pp. 42–58. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10085-2_4
15. Novak, D., Zezula, P.: PPP-codes for large-scale similarity searching. In: Hameurlain, A. (ed.) TLDKS XXIV. LNCS, vol. 9510, pp. 61–87. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49214-7_2
16. Skopal, T., Pokorny, J., Snasel, V.: PM-Tree: pivoting metric tree for similarity search in multimedia databases. In: Proceedings of ADBIS 2004, pp. 99–114 (2004)
17. Tellez, E.S., Chavez, E., Navarro, G.: Succinct nearest neighbor search. Inf. Syst. **38**(7), 1019–1030 (2013)
18. Wang, Z., Dong, W., Josephson, W., Lv, Q., Charikar, M., Li, K.: Sizing sketches: a rank-based analysis for similarity search. SIGMETRICS Perform. Eval. Rev. **35**(1), 157–168 (2007)
19. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: the Metric Space Approach. Advances in Database Systems, vol. 32. Springer Science & Business Media, New York (2006)