K-medoids LSH: a new locality sensitive hashing in general metric space

Eliezer S. Silva, Eduardo Valle

RECOD Lab – DCA/FEEC, University of Campinas, Brazil
{eliezers, dovalle}@dca.fee.unicamp.br

Abstract. The increasing availability of multimedia content poses a challenge for information retrieval researchers. Users want not only have access to multimedia documents, but also make sense of them. The ability of finding specific content in extremely large collections of textual and non-textual documents is paramount. At such large scales, Multimedia Information Retrieval systems must rely on the ability to perform search by similarity efficiently. However, Multimedia Documents are often represented by high-dimensional feature vectors, or by other complex representations in metric spaces. Providing efficient similarity search for that kind of data is extremely challenging. In this article, we explore one of the most cited family of solutions for similarity search, the Locality-Sensitive Hashing (LSH), which is based upon the creation of hashing functions which assign, with higher probability, the same key for data that are similar. LSH is available only for a handful distance functions, but, where available, it has been found to be extremely efficient for architectures with uniform access cost to the data. Most of existing LSH functions are restricted to vector spaces. We propose a novel LSH method for generic metric space based on K-medoids clustering. We present comparison with well established LSH methods in vector spaces and with recent competing new methods for metric spaces. Our early results show promise, but also demonstrate how challenging is to work around those difficulties.

Categories and Subject Descriptors: H.2 [**Database Management**]: Miscellaneous; H.3.1 [**Content Analysis and Indexing**]: Indexing methods; H.3.3 [**Information Search and Retrieval**]: Miscellaneous

Keywords: hashing, metric space indexing, nearest neighbor search, similarity search

1. INTRODUCTION

Content-based Multimedia Information Retrieval (CMIR) is an alternative to keyword-based or tag-based retrieval, which works by extracting *features* based on distinctive properties of the multimedia objects. Those features are organized in *multimedia descriptors*, which are used as surrogates of the multimedia object, in such a way that the retrieval of similar objects is based solely on that higher level representation, without the need to refer to the actual low-level encoding of the media. The descriptor can be seen as a compact and distinctive representation of multimedia content, encoding some invariant properties of the content. For example, in image retrieval the successful *Scale Invariant Feature Transform* (SIFT) [Lowe 2004] encodes local gradient patterns around Points-of-Interest, in a way that is (partially) invariant to illumination and geometric transformations.

The descriptor framework also allows to abstract the media details in multimedia retrieval systems. The operation of looking for similar multimedia documents becomes the more abstract operation of looking for multimedia descriptors which have small distances. The notion of a "feature space", that organizes the documents in a geometry, putting close-by those that are similar emerges. Of course, CMIR systems are usually much more complex than that, but nevertheless, looking for similar descriptors often plays a critical role in the processing chain of a complex system.

Although the operation of finding descriptors which have small distances seems simple enough,

E.S. Silva is supported by a CAPES/PROEX grant.

performing it fast for multimedia descriptors is actually very challenging, due to the scale of the collections, the dimensionality of the descriptors and the diversity of distance functions [Akune et al. 2010]. The literature on the subject is very extensive, but in this article, we focus on one of the most cited family of solutions, the *Locality-Sensitive Hashing* (LSH) [Indyk and Motwani 1998; Gionis et al. 1999; Datar et al. 2004], proposing *K-medoids LSH* as an extension to metric space.

2. LOCALITY SENSITIVE HASHING (LSH)

The LSH indexing method relies on a family of locality-sensitive hashing function H, [Indyk and Motwani 1998], to map objects from a metric domain U in a D-dimensional space (usually $\mathbb{R}^d$) to a countable set C (usually $\mathbb{Z}$), with the following property: nearby points in high dimensional space are hashed to the same value with high probability.

DEFINITION 1. *Given a distance function $d : U \times U \to \mathbb{R}^+$, a function family $H = \{h : U \to C\}$ is (r, cr, p_1, p_2)-sensitive for a given data set $S \subseteq U$ if, for any points $p, q \in S$, $h \in H$:*

—If $d(p, q) \leq r$ then $Pr_H[h(q) = h(p)] \geq p_1$ (probability of colliding within the ball of radius r),
—If $d(p, q) > cr$ then $Pr_H[h(q) = h(p)] \leq p_2$ (probability of colliding outside the ball of radius cr)
—$c > 1$ and $p_1 > p_2$

The basic scheme [Indyk and Motwani 1998] provided locality-sensitive families for the Hamming distance on Hamming spaces, and the Jacquard distance in spaces of sets. For several years those were the only families available, although extensions for L_1-normed (Manhattan) and L_2-normed (Euclidean) spaces were proposed by embedding those spaces into a Hamming space [Gionis et al. 1999].

The practical success of LSH, however, came with the E2LSH[1] (Euclidean LSH) [Datar et al. 2004], for L_p-normed space, where a new familiy of LSH functions was introduced:

$$H = \{h_i : \mathbb{R}^D \to \mathbb{Z}\} \tag{1}$$

$$h_i(\mathbf{v}) = \left\lfloor \frac{\mathbf{a}_i.\mathbf{v} + b_i}{w} \right\rfloor \tag{2}$$

$\mathbf{a}_i \in \mathbb{R}^D$ is a random vector with each coordinate picked independently from a Gaussian distribution $N(0, 1)$, b_i is an offset value sampled from uniform distribution in the range $[0, \ldots, w]$ and w is a scalar for the quantization width. Applying h_i to a point or object $\mathbf{v}$ corresponds to the composition of a projection to a random line and a quantization operation, given by the quantization width w and the floor operation.

A function family G is constructed by concatenating M randomly sampled functions $h_i \in H$, such that each $g_j \in G$ has the following form: $g_j(\mathbf{v}) = (h_1(\mathbf{v}), ..., h_M(\mathbf{v}))$. The use of multiple h_i functions reduces the probability of false positives, since two objects will have the same key for g_j only if their value coincide for all h_i component functions. Each object $\mathbf{v}$ from the input dataset is indexed by hashing it against L hash functions $g_1, \ldots, g_L$. At the *search phase* a query object $\mathbf{q}$ is hashed using the same L hash functions and the objects stored in the given buckets are used as the candidate set. Then, a ranking is performed among the candidate set according to their distance to the query, and the k closest objects are returned.

A few recent works approach the problem of designing LSH indexing schemes using only the distance information: M-Index [Novak et al. 2010], DFLSH (Distribution Free Locality-Sensitive Hashing) [Kang and Jung 2012] and [Tellez and Chavez 2010].

[1] *LSH Algorithm and Implementation (E2LSH)*, accessed in 22/09/2013. `http://www.mit.edu/~andoni/LSH/`

K-means LSH. [Paulevé et al. 2010] present a comparison between *structured* (regular random projections, lattice) and *unstructured* (K-means and hierarchical K-means) quantizers in the task of searching high dimensional SIFT descriptors, resulting on the proposal of a new LSH family based on the latter (Equation 3). Results indicate that the LSH functions based on unstructured quantizers perform better, as the induced Voronoi partitioning adapts to the data distribution, generating more uniform hash cells population than the structured LSH quantizers. However, K-means and hierarchical K-means are clustering algorithms for vector spaces, restricting the application of this approach. In order to overcome this limitation we turn to a clustering algorithm designed to work in generic metric spaces – *K-medoids clustering*.

3. K-MEDOIDS LSH

We propose a novel method for locality-sensitive hashing in the metric search framework and compare them with other similar methods in the literature. Our method is rooted on the idea of partitioning the data space with a distance-based clustering algorithm (K-medoids) as an initial quantization step and constructing a hash table with the quantization results. *K-medoids LSH* follows a direct approach taken from [Paulevé et al. 2010]: each partition cell is a hash table bucket, therefore the hashing is the index number of the partition cell.

Definition 2. *Given a metric space (U, d) (U is the domain set and d is the distance function), the set of cluster centers $C = \{c_1, \ldots, c_k\} \subset U$ and an object $x \in U$:*

$$\begin{gathered} h_C : U \to \mathbb{N} \\ h_C(x) = \text{argmin}_{i=1,\ldots,k}\{d(x, c_1), \ldots, d(x, c_i), \ldots, d(x, c_k)\} \end{gathered} \tag{3}$$

PAM (Partitioning Around Medoids) [Kaufman and Rousseeuw 1990] is the classic algorithm for K-medoids clustering. In that work a medoid is defined as the most centrally located object within a cluster. The algorithm initially selects a group of medoids at random (or using some heuristics), then iteratively assigns each non-medoid point to the nearest medoid and update the medoid set, looking for the optimal set of medoid points minimizing the quadratic sum of distance. PAM features a prohibitive computational cost, since it performs $O(k(n-k)^2)$ distance calculation for each iteration. There are a few methods designed to cope with that complexity constraint. Park and Jun [Park and Jun 2009] propose a simple approximate algorithm based on PAM with significant performance improvements. Instead of searching the entire dataset for a new optimal medoid, the method restricts the search for points within the cluster. We choose to apply this method for its simplicity and performance.

Therefore, our baseline method consists in the following steps:

(1) Generate L lists of k cluster centers ($C_i = \{c_{i1}, \ldots, c_{ik}\}, \forall i \in \{1, \ldots, L\}$) in L runnings of the Park and Jun fast k-medoid algorithm (as in In Paulevé *et al.* [Paulevé et al. 2010], this clustering is done over a sampling of the dataset).
(2) Index each point x of the dataset using $h_{C_i}(x)$ as the bucket key for each table ($i \in \{1, \ldots, L\}$).
(3) Given a query point q, hash it using $h_{C_i}(q)$ ($i \in \{1, \ldots, L\}$) and retrieve all colliding points in a candidate set of kNN[2] points. Then perform a linear scan over that candidate list.

Initialization. K-means clustering results exhibit a strong dependence on the initial cluster selection. K-means++ [Ostrovsky et al. 2006; Arthur and Vassilvitskii 2007] solve the $O(\log k)$ approximate K-means problem[3] simple by carefully designing a good initialization algorithms. That raises the question of how the initialization could affect the final result of the kNN search for the proposed methods. The

[2]kNN - k nearest neighbors
[3]the exact solution is NP-Hard

K-means++ initialization method is based on metric distance information and sampling, thus can be plugged into a K-medoids algorithm without further changes. [Park and Jun 2009] propose also a special initialization for the fast K-medoids algorithm. We implemented both of those initialization, and the random selection as well and empirically evaluated their impact on the similarity search task.

3.1 Experiments and analysis

Datasets:. for the initial experiments we resorted to a dataset called APM (*Arquivo Público Mineiro* – The Public Archives in Minas Gerais) which was created from the application of various transformation (15 transformation of scale, rotation, etc) to 100 antique photographs of the Public Archives. The SIFT descriptors of each transformed image were calculated, resulting in a dataset of 2.871.300 feature vectors (SIFT descriptor is a 128 dimensional vector). The queries dataset is build from the SIFT descriptors of the original images (a total of 263.968 feature vectors). Each query point is equiped with its set of true nearest neighbors – the ground-truth. For these experiments we used 500 points uniformly sampled from the query dataset and performed a 10-NN search.

Metrics:. We are concerned especially with the relations between the recall and selectivity metric given the variation in the parametric space of the methods. The *recall* metric is the fraction of total correct answers over the number of true relevant answers. The *selectivity*[4] metric is the fraction of the dataset selected for the shortlist processing. Selectivity is an important metric for those methods being considered, since the shortlist processing is a bottleneck in the whole query processing [Paulevé et al. 2010].

K-means LSH, K-medoids LSH and DFLSH:. In order to evaluate the feasibility of K-medoids LSH we compared it with the K-means LSH *et al.* [Paulevé et al. 2010] and the DFLSH [Kang and Jung 2012]. We implemented all the methods in the same framework, namely Java and *The Apache Commons^{TM} Mathematics Library*[5]. K-means LSH is used as a baseline method, since its performance and behavior are well studied and presented in the literature and DFLSH is as an alternative method for the same problem. Figure 1 shows recall, query time and selectivity statistics averaged over 500 queries and using a single hash function[6].

Figure 1a presents results relating recall with selectivity. Notice that high selectivity means that a large part of the dataset is being processed in the final sequential ranking – that is not a desirable point of operation, since the main performance bottleneck on the query processing time is located at that stage(1b). For a selectivity as low as 0.3%, both methods' recall metric is approximately 65% and with 1% selectivity it grows up to a 80%. These results could be improved by using more than one hash function. Clearly K-means LSH presents the best result on the recall metric, however it is important to notice that both DFLSH and K-medoids LSH are not exploiting any vector coordinate properties, relying solely on distance information.

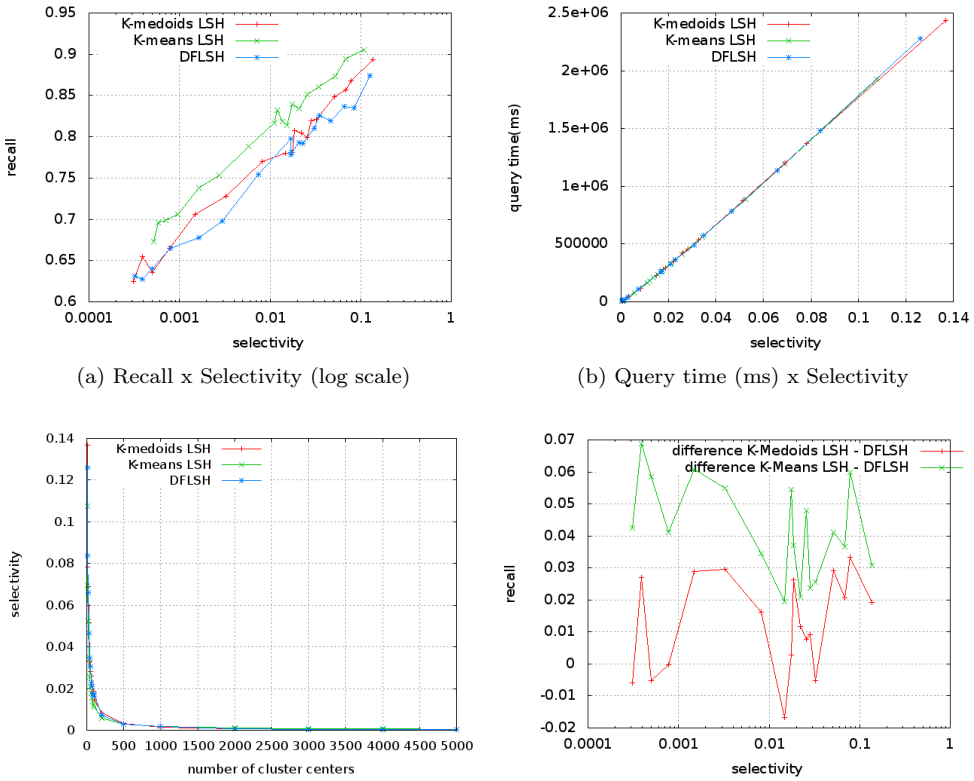
Figure 1b depicts a strong linear correlation between query time and selectivity. Other noticeable strong correlation appears between selectivity and number of cluster centers (Figure 1c). Theoretically it is possible to show that, given an approximate uniform population of points in the hash buckets, the selectivity for a t number of cluster centers is $O(t^{-1})$. The plot shows that the experimental data follows a power law curve.

---

[4]the term selectivity has distinct use in the database and image retrieval research communities. Here we adopt the latter, following *Paulève et al.*

[5]*Commons Math: The Apache Commons Mathematics Library*, accessed in 22/09/2013. `http://commons.apache.org/proper/commons-math/`

[6]Including more than one hash function would have approximately the same effect in the recall and querying time metric for all methods, nevertheless the preprocessing time for K-means LSH and K-medoids LSH (envolving extra rounds of the optimization procedure) would increase much more then for DFLSH (sampling more points)



(a) Recall x Selectivity (log scale)

(b) Query time (ms) x Selectivity

(c) Selectivity x Number of cluster centers

(d) Difference of Recall x Selectivity (log scale)

Fig. 1: Comparison of K-means LSH, K-medoids LSH and DFLSH for 10-NN

Using DFLSH as a baseline, we plot the difference on the recall from K-means LSH and K-medoids LSH to DFLSH (Figure 1d. The recall difference can be up to 0.07 positive (11%) for K-means LSH and 0.03 (5%) for K-medoids LSH. The trend on the differences is sustained along the curves for different values of selectivity. There is a clear indication that K-medoids LSH is a viable approach with equivalent performance to K-means LSH and DFLSH.

Initialization effect:. First of all it is important to notice that the initial segment of the curves in Figure 2 are not much informative – those points corresponds to number of clusters up to 20, implying in an almost brute-force search over the dataset (notice the explosion on the selectivity in Figure 1c). Figure 2 depicts the difference on the recall metric for the two initialization algorithm, using the random selection as a baseline. The K-means++ initialization affects positively the results with average gain of 3%. In the other hand the Park and Jun initialization does not presents great gain over the random baseline (in fact, the average gain is negative). Further experiments with more datasets are needed in order to achieve statistical confidence on these results, but the initial analysis indicates that the K-means++ initialization can contribute to better results in the recall metric, while the Park and Jun method presents a null effect (or negative).

4. CONCLUSION

Efficient large-scale similarity search is a crucial operation for Content-based Multimedia Information Retrieval (CMIR) systems. But because those systems employ high-dimensional feature vectors, or other complex representations in metric spaces, providing fast similarity search for them has been a persistent research challenge. LSH, a very successful family of methods, has been advanced as a

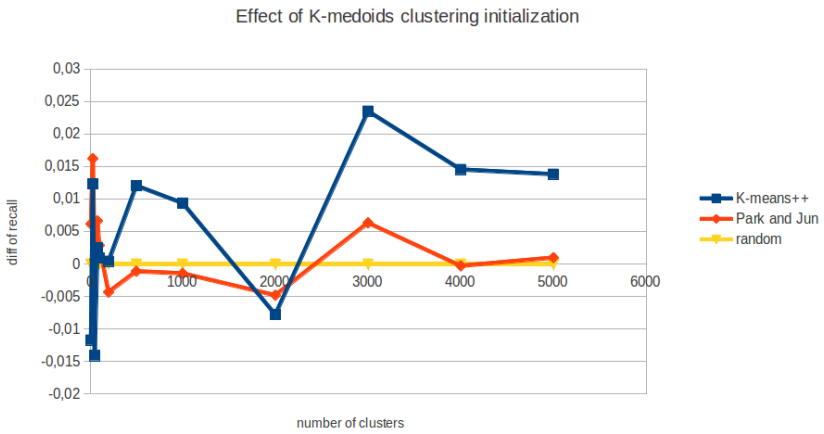


Fig. 2: Effect of the initialization procedure for the K-medoids clustering in the quality of nearest-neighbors search

solution to the problem, but it is available only for a few distance functions. In this article we propose to address that limitation, by extending LSH to general metric spaces, using K-medoids clustering as basis for a LSH family of functions. We show in our experiments that the K-medoids LSH improves the results over the random choice of sample of DFLSH, while keeping the advantage of relying only on distance information. As expected, K-medoids LSH performance is slightly worse than K-means LSH, but it is important to note that K-means relies heavily on the vector-space structure, and many data types of interest do not offer such structure.

REFERENCES

Akune, F., Valle, E., and Torres, R. MONORAIL: A Disk-Friendly Index for Huge Descriptor Databases. In *2010 20th International Conference on Pattern Recognition*. IEEE, pp. 4145–4148, 2010.

Arthur, D. and Vassilvitskii, S. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. SODA '07. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 1027–1035, 2007.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry - SCG '04*. ACM Press, New York, New York, USA, pp. 253, 2004.

Gionis, A., Indyk, P., and Motwani, R. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*. VLDB '99. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 518–529, 1999.

Indyk, P. and Motwani, R. Approximate nearest neighbors. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*. ACM Press, New York, New York, USA, pp. 604–613, 1998.

Kang, B. and Jung, K. Robust and Efficient Locality Sensitive Hashing for Nearest Neighbor Search in Large Data Sets. In *NIPS Workshop on Big Learning (BigLearn)*. Lake Tahoe, Nevada, pp. 1–8, 2012.

Kaufman, L. and Rousseeuw, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.

Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60 (2): 91–110, Nov., 2004.

Novak, D., Kyselak, M., and Zezula, P. On locality-sensitive indexing in generic metric spaces. *Proceedings of the Third International Conference on SImilarity Search and APplications - SISAP '10*, 2010.

Ostrovsky, R., Rabani, Y., Schulman, L., and Swamy, C. The Effectiveness of Lloyd-Type Methods for the k-Means Problem. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. Vol. 59. IEEE, pp. 165–176, 2006.

Park, H.-S. and Jun, C.-H. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications* 36 (2): 3336–3341, 2009.

Paulevé, L., Jégou, H., and Amsaleg, L. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters* 31 (11): 1348–1358, Aug., 2010.

Tellez, E. S. and Chavez, E. On locality sensitive hashing in metric spaces. In *Proceedings of the Third International Conference on SImilarity Search and APplications*. SISAP '10. ACM, New York, NY, USA, pp. 67–74, 2010.