Re-ranking via Local Embeddings: A Use Case with Permutation-based Indexing and the nSimplex Projection

Lucia Vadicamo[a,*], Claudio Gennaro[a], Fabrizio Falchi[a], Edgar Chávez[b], Richard Connor[c], Giuseppe Amato[a]

[a]*Institute of Information Science and Technologies (ISTI), CNR, Pisa, Italy*
[b]*Department of Computer Science, CICESE, Ensenada, Mexico*
[c]*Division of Mathematics and Computing Science, University of Stirling, Scotland*

Abstract

Approximate Nearest Neighbor (ANN) search is a prevalent paradigm for searching intrinsically high dimensional objects in large-scale data sets. Recently, the permutation-based approach for ANN has attracted a lot of interest due to its versatility in being used in the more general class of metric spaces. In this approach, the entire database is ranked by permutation distance to the query and revised in that order, and even if some clever tricks can be used to avoid a sequential scan of the database for this ranking, a sizeable percentage of the database should be reviewed, using the original metric, to achieving high recall.

To reduce the number of metric computations and the number of database elements accessed, we propose, in this paper, a re-ranking based on a local embedding using the nSimplex projection. The nSimplex projection produces euclidean vectors from objects in metric spaces obeying the n-point property. The mapping is obtained from the distances to a set of references, and the original metric can be lower bounded and upper bounded by the euclidean distance of objects sharing the same set of references.

Our approach is particularly advantageous for extensive databases or expensive metric function. We reuse the distances computed in the permutations in the first stage, and hence the memory footprint of the index is not increased.

An extensive experimental evaluation of our approach is presented, demonstrating excellent results even on a set of hundreds of millions of objects.

Keywords: metric search, permutation-based indexing, n-point property, nSimplex projection, metric local embeddings, distance bounds

*Corresponding author

Email addresses: lucia.vadicamo@isti.cnr.it (Lucia Vadicamo), claudio.gennaro@isti.cnr.it (Claudio Gennaro), fabrizio.falchi@isti.cnr.it (Fabrizio Falchi), elchavez@cicese.mx (Edgar Chávez), richard.connor@stir.ac.uk (Richard Connor), giuseppe.amato@isti.cnr.it (Giuseppe Amato)

1. Introduction

Proximity search is successfully used to retrieve data objects that are close to a given query object under some metric function. It has a vast number of applications in many branches of computer science, including pattern recognition, computational biology, and multimedia information retrieval, to name but a few. This search paradigm, referred to as *metric search*, is based on the assumption that data objects are represented as elements of a metric space (D, d) where the *metric*[1] function $d : D \times D \rightarrow \mathbb{R}^+$ provides a measure of the closeness of the data objects.

In metric search, the main concern is processing and structuring a finite set of data $X \subset D$ so that *proximity queries* can be answered quickly and with a low computational cost. A proximity query is defined by a query object $q \in D$ and a proximity condition, such as "find all the objects within a threshold distance of q" (*range query*) or "finding the k closest objects to q" (*k-nearest neighbour query*). The response to a query is the set of all the objects $o \in X$ that satisfy the considered proximity condition. In this work, we focus on the k-nearest neighbour (k-NN) search since, as also highlighted in [2, 3], (i) it allows us to control the size of the results set, and (ii) it is simpler to use in high-dimensional space where it is not obvious to define a meaningful distance value to be used with other search paradigms, like the range query. However, providing an exact response to a k-NN query is not feasible if the search space is very large or it has a high intrinsic dimensionality since it would be necessary to inspect a large fraction of the data to process the query. In such cases, the exact search rarely outperforms a sequential scan [4, 5]. To overcome this phenomenon, known as *curse of dimensionality* [6], researchers proposed several *approximate search* methods that are less (but still) affected by it. The main idea of approximate methods is to efficiently find a set of results that is likely to contain most of the objects that satisfy the query proximity condition. However, the efficiency of these methods comes at the expense of a certain reduction of the accuracy (e.g. false hits or missing results). A limited imprecision in the response to the query, however, is totally tolerated in many applications, such as multimedia retrieval where the concept of "(di)similarity" may differ on the user' expectations, and close approximations may be good enough for human perception [7].

Many Approximate Nearest Neighbor (ANN) methods are based on the idea of mapping the data objects into a more tractable space in which we can efficiently perform the search. Successful examples are the *Permutation-Based Indexing* (PBI) approaches that represent data objects as a sequence of identifiers (*permutation*). Typically, the permutation for an object o is computed as a ranking list of some preselected reference points (*pivots*) according to their distance to o. The main rationale behind this approach is that if two objects are very close one to the other they will sort the set of pivots in a very similar way,

[1]Throughout this paper, we use the terms "metric" and "distance" interchangeably to indicate a function satisfying the metric postulates of non-negativity, identity, symmetry, and triangle inequality [1].

and thus the corresponding permutation representations will be close as well. The search in the permutation space is used to select a candidate result set that is then typically refined by comparing each candidate object to the query one (according to the actual metric governing the data space). This refinement step, therefore, requires access to the original data, which is likely to be too large to fit into the main memory. However, some kind of refinement step is likely to be necessary as the search in the permutation space usually has relatively low precision.

In this paper, we focus on permutation-based k-NN search and we investigate several approaches to perform the refining step without access to the original data. Our techniques approximate the actual distance between a query and the candidate objects by exploiting the distances between the objects and the pivots (calculated at indexing time and stored within the permutations) and the distances between the query and the pivots (evaluated when computing the query permutation). In particular, for a large class of metric spaces that meet the so-called "*n-point property*" [8, 9] we propose the use of the *nSimplex projection* [10] that allows mapping metric objects into a finite-dimensional Euclidean space where upper- and lower- bounds for the original distances can be calculated. We show how these distance bounds can be used to improve the permutation-based results without accessing to the original data set.

A preliminary version of this work appeared in [11]. The present contribution gives, also, a more detailed description of the proposed approaches and an extensive experimental evaluation. In particular, it includes new results on large scale and investigates the use of compressed versions of the inverted files to index the data. The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 provides basic concepts of the metric space transformations used in our work (namely, permutation-based representations, Pivoted embedding, and nSimplex projection). In Section 4 we describe several pivot-based approaches to refine a permutation-based candidate set. A detailed experimental evaluation and analysis of those approaches is presented in Section 5. Finally, the conclusion is drawn in the last section.

Table 1 summarises the notation used in this paper.

2. Related Work

The idea of approximating the distance between any two metric objects by comparing their permutation-based representations was originally proposed in [12, 13]. Several techniques for indexing and searching permutations were considered in literature, including indexes based on inverted files, like the *Metric Inverted File* (MI-File) [14] and its variants [15], or using prefix trees, like the *Permutation Prefix Index* (PP-Index) [2] and the *Pivot Permutation Prefix Index* (PPP-Index) [16]. In [17], the metric objects in the inverted index are represented by a signature built from the l nearest references to them. However, in all above approaches, the candidate result set identified by performing the search in the permutation space should be refined to achieve high effectiveness. Typically the results are refining by directly comparing the query object

Table 1: Notation used throughout this paper

Symbol	**Definition**
(D, d)	metric space
X	finite search space, $X \subseteq D$
$\mathcal{P}_n = \{p_1, \dots, p_n\}$	set of pivots, $p_i \in D$
n	number of pivots
o, s	data objects, $o, s \in X$
q	query, $q \in D$
k, k'	number of results of a nearest neighbour search
amp	amplification factor
Π_o	pivot permutation
Π_o^{-1}	inverted permutation
l	permutation prefix length (location parameter)
$\Pi_{o,l}$	permutation prefix of length l (truncated permutation)
$\Pi_{o,l}^{-1}$	inverted truncated permutation
$PivotSet(\Pi_{o,l})$	the pivots whose identifiers appear in $\Pi_{o,l}$
$\Gamma_{o,q}$	pivots in the intersection $PivotSet(\Pi_{q,l}) \cap PivotSet(\Pi_{o,l})$
$S_{\rho,l}$	Spearman's rho with location parameter l
ℓ_2	Euclidean distance
ℓ_∞	Chebyshev distance
$f_{\mathcal{P}_n} : (D, d) \to (\mathbb{R}^n, \ell_\infty)$	Pivoted embedding
$\phi_{\mathcal{P}_n} : (D, d) \to (\mathbb{R}^n, \ell_2)$	nSimplex projection
$\lvert \cdot \rvert$	size of a set

with the obtained candidate results according to the original distance and data representation.

The common approach to generate a permutation-based representation of a data object is ordering the identifiers of a set of pivots according to their distances to the object [18]. However, the computation of these distances is just one, yet effective, approach to associate a permutation to each data object. For example, the *Deep Permutations* [19] have been recently proposed as an efficient and effective alternative for generating permutations of emerging deep features. However, this approach is suitable only for specific data domains while the traditional approach is generally applicable since it requires only the existence of a distance function to compare data objects.

In [20], Figueroa et al. have tried different distances between permutations instead of the canonical Spearman Footrule or Spearman Rho metrics. The aim of this work, however, was to reduce the number of distance computations and the size of the index.

The distances between the data objects and a set of pivots can be used also to embed the data into another metric space where it is possible to deduce upper- and lower- bounds on the actual distance of any pair of objects. In this context, one of the very first embeddings proposed in a metric search scenario was the one representing each data object with a vector of its distances to the pivots. The LAESA [21] is a notable example of indexing technique using this approach. Connor et al. [10, 9, 22] observed that for a large class of metric spaces it is possible to use the distances to a set of n pivots to project the data objects into a n-dimensional Euclidean space such that in the projected space 1) the distances object-pivots are preserved, 2) the Euclidean distance between any two points is a lower-bound of the actual distance, 3) also an upper-bound can be easily computed. They called this approach *nSimplex projection* and they proved that it can be used in all the metric spaces meeting the *n-point property* [23]. As also pointed out in [8], many common metric spaces meet the desired property, like Cartesian spaces of any dimension with the Euclidean, cosine or quadratic form distances, probability spaces with the Jensen-Shannon or the Triangular distance, and more generally any Hilbert-embeddable space [23, 24].

Recently, The nSimplex projection has been exploited to generate a novel permutation-based representation for metric objects, called SPLX-Perm [**?**]. It is based on the idea of mapping the data object to Euclidean vectors, which are in turn transformed into permutations using an approach similar to that used in the Deep Permutations [19].

3. Background

This section summarises key concepts of some metric space transformations based on the use of distances between data objects and a set of pivots. The rationale behind these approaches is to project the original data into a space that has better indexing properties than the original one, or where the function used to compare the objects is less expensive than the original distance. In particular, we review data transformations into permutation spaces (where objects can be efficiently indexed using PBI methods) and two pivot-based embeddings that allow computing upper- and lower- bounds of the actual distance.

3.1. Permutation-Based Representations

Let $\mathcal{D}$ a data domain and $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}^+$ a *metric* function on it[2]. A permutation-based representation Π_o (briefly *permutation*) of an object $o \in \mathcal{D}$ with respect to a fixed set of *pivots*, $\{p_1, \ldots, p_n\} \subset \mathcal{D}$, is the sequence of pivots identifiers ordered by their distance to o.

Formally, the permutation $\Pi_o = [\Pi_o(1), \Pi_o(2), ..., \Pi_o(n)]$ lists the pivot identifiers $\{1, \ldots, n\}$ in an order such that $\forall\, i \in \{1, \ldots, n-1\}$,

$$d(o, p_{\Pi_o(i)}) < d(o, p_{\Pi_o(i+1)}) \tag{1}$$

or

$$\left[d(o, p_{\Pi_o(i)}) = d(o, p_{\Pi_o(i+1)})\right] \wedge \left[\Pi_o(i) < \Pi_o(i+1)\right]. \tag{2}$$

An equivalent permutation-based representation is the *inverted permutation*, defined as $\Pi_o^{-1} = [\Pi_o^{-1}(1), \Pi_o^{-1}(2), \ldots, \Pi_o^{-1}(n)]$, where $\Pi_o^{-1}(i)$ denotes the position of a pivot p_i in the permutation Π_o. The inverted permutation is such that $\Pi_o(\Pi_o^{-1}(i)) = i$. Note that the value at the coordinate i in the permutation Π_o is the identifier of the pivot at i-th position in the ranked list of the nearest pivots to o; the value at the coordinate i in the inverted representation Π_o^{-1} is the rank of the pivot p_i in the list of the nearest pivots to o.

The inverted permutation representation is often used in practice since it allows us to represent permutations in a Cartesian coordinate system and easily compute most of the commonly-used distances between permutations as distances between Cartesian points. Several metric functions have been used in literature to compare permutations, notably including Kendall's tau, Spearman's Rho and the Spearman's Footrule distances. In this paper, we use the Spearman's Rho. It is defined as $S_\rho(\Pi_o, \Pi_s) = \ell_2(\Pi_o^{-1}, \Pi_s^{-1})$ for any two permutations Π_o, Π_s.

Most of the PBI methods, e.g. [14, 2, 16], use only a fixed-length prefix of the permutations to represent and compare objects. This choice is based on the intuition that the most relevant information in the permutation is present in its very first elements, i.e. the identifiers of the closest pivots. Moreover, using the positions of the nearest l out of n pivots often leads to obtaining better or similar effectiveness then using the full-length permutation [14, 18], resulting also in a more compact data encoding. The permutation prefixes are compared using *top-l distances* [25]. An example is given by the Spearman's Rho with location parameter l, which is defined as $S_{\rho,l},(\Pi_o, \Pi_s) = \ell_2(\Pi_{o,l}^{-1}, \Pi_{s,l}^{-1})$, where

[2]In this work, we focus on metric search. The requirement that the function d satisfies the metric postulates is sufficient, but not necessary, to produce a permutation-based representation. For example, d may be a dissimilarity function.

$\Pi_{o,l}^{-1}$ is the *inverted truncated permutation*:

$$\Pi_{o,l}^{-1}(i) = \begin{cases} \Pi_o^{-1}(i) & \text{if} \quad \Pi_o^{-1}(i) \leq l \\ l+1 & \text{otherwise} \end{cases} \tag{3}$$

3.2. Pivoted embedding

The distances between the metric objects and a set of pivots $\mathcal{P}_n = \{p_1, \ldots, p_n\}$ can be used to embed a metric space into $(\mathbb{R}^n, \ell_\infty)$ by using the following transformation:

$$\begin{aligned} f_{\mathcal{P}_n} : (D, d) &\to (\mathbb{R}^n, \ell_\infty) \\ o &\to [d(o, p_1), \ldots, d(o, p_n)] \end{aligned}$$

The triangle inequality of the metric governing the space guarantees that

$$\max_{i=1,\ldots,n} |d(o,p_i) - d(s,p_i)| \leq d(o,s) \leq \min_{i=1,\ldots,n} |d(o,p_i) + d(s,p_i)| \tag{4}$$

which it means that $\ell_\infty(f_{\mathcal{P}_n}(o), f_{\mathcal{P}_n}(s))$ is a lower-bound of $d(o,s)$ and that also an upper-bound can be defined using the projected objects $f_{\mathcal{P}_n}(o)$, $f_{\mathcal{P}_n}(s)$ (see [1, pp.28]). Please note that if we use just a subset $\mathcal{P}_l$ of size l of the pivots $\{p_1, \ldots, p_n\}$, the corresponding mapping $f_{\mathcal{P}_l}$ provides upper- and lower- bounds that are less tight than that obtained using $f_{\mathcal{P}_n}$.

This family of embeddings, referred to as *Pivoted embedding* in the following, are typically used in indexing tables like LAESA [21] or for space pruning [1]. However, as further described in Section 4, in this work we used them not for indexing purpose, but rather as techniques to approximate the distances between a query and data objects already indexed using a permutation-based approach.

3.3. nSimplex projection

The *nSimplex projection* [10] is a space transformation of the form

$$\phi_{\mathcal{P}_n} : (D, d) \to (\mathbb{R}^n, \ell_2)$$

that uses the distances to a set of pivots $\mathcal{P}_n = \{p_1, \ldots, p_n\}$ for embedding metric objects into a finite-dimensional Euclidean space. It can be applied to any metric space that meets the so called *n-point property* [23], which provides geometric guarantees stronger than triangle inequality. In particular, a metric space has the n-point property if, and only if, any set of n points of the space can be *isometrically* embedded into a $(n-1)$-dimensional Euclidean space, i.e. there exist a mapping of those points to n Euclidean vectors that preserves all the $\binom{n}{2}$ inter-points distances. In other words, the n points can be isometrically mapped to the vertices of a $(n-1)$-dimensional simplex[3].

[3]A simplex is a generalisation of a triangle (2-dimensional simplex) or a tetrahedron (3-dimensional simplex) in arbitrary dimensions. Specifically, the $(n-1)$-dimensional simplex generated by the vertices $v_1, \ldots, v_n$ equals the union of all the line segments joining v_n to the points of the $(n-2)$-dimensional simplex of vertices $v_1, \ldots, v_{n-1}$.

The n-point property guarantees that given the set of pivots $\mathcal{P}_n$, we can determine the vectors $v_{p_1}, \dots, v_{p_n}$ such that

$$\forall\, i, j \in \{1, \dots, n\}: \quad \ell_2(v_{p_i}, v_{p_j}) = d(p_i, p_j).$$

We refer the $(n-1)$-dimensional simplex generated by those vectors to as the *simplex base*. Then, for any further object $o \in D$ the $(n+1)$-point property guarantees that there exists a vertex $v_o \in \mathbb{R}^n$ such that

$$\forall\, i \in \{1, \dots, n\}: \quad \ell_2(v_{p_i}, v_o) = d(p_i, o),$$

i.e., the vector v_o is the apex of a n-dimensional simplex built upon the simplex base where the length of the i-th edge connecting v_o to the simplex base equals the actual distance $d(p_i, o)$. The nSimplex projection $\phi_{\mathcal{P}_n}$ is the transformation that maps an object $o \in D$ to the apex $v_o \in \mathbb{R}^n$ built upon the simplex base. Connor et al. [10] provided an iterative algorithm to compute the coordinates of the vertices v_{p_i} of the simplex base as well as the coordinates of the apex v_o associated to a metric object o. Remark that this algorithm determines those coordinates by only exploiting the distances $d(p_i, p_j)$ and $d(p_i, o)$, for $i, j \in \{1, \dots, n\}$. Moreover, the simplex base is computed once and is reused for projecting every data object. Given the distances $d(p_i, o)$, the cost for computing v_o is $O(n)$ Euclidean distance between vectors having less than n dimensions.

One of the main outcomes of this embedding is that it allows us to determine upper- and lower-bounds of the actual distance by computing the Euclidean distance between two vectors. In facts, given the apexes

$$\begin{aligned} \phi_{\mathcal{P}_n}(o) &= [x_1, x_2, \dots, x_{n-1}, x_n] \\ \phi_{\mathcal{P}_n}(s) &= [y_1, y_2, \dots, y_{n-1}, y_n] \end{aligned}$$

it holds

$$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad \leq \quad d(o, s) \quad \leq \quad \sqrt{\sum_{i=1}^{n-1} (x_i - y_i)^2 + (x_n + y_n)^2} \tag{5}$$

for any two objects $o, s \in D$. Therefore given the vector

$$\phi^-_{\mathcal{P}_n}(s) = [y_1, y_2, \dots, y_{n-1}, -y_n]$$

we have that $\ell_2(\phi_{\mathcal{P}_n}(o), \phi_{\mathcal{P}_n}(s))$ and $\ell_2(\phi_{\mathcal{P}_n}(o), \phi^-_{\mathcal{P}_n}(s))$ are respectively a lower- and and upper-bound for $d(o, s)$. Moreover, these bounds become tighter with increasing number of pivots n [26, 10]. Recently,

Note that, as observed in [8, 10], there exist a large class of metric spaces that satisfy the n-point property and therefore can be transformed by the nSimplex projection. Examples are given by the Euclidean spaces of any dimension, spaces with the Triangular or Jensen-Shannon distances, and, more generally, any Hilbert-embeddable spaces. Moreover, if a metric space does not meet the n-point property, (e.g. Hamming or Chebyshev metrics), there always exists a proximity preserving mapping of this space into a metric space with this property [27].

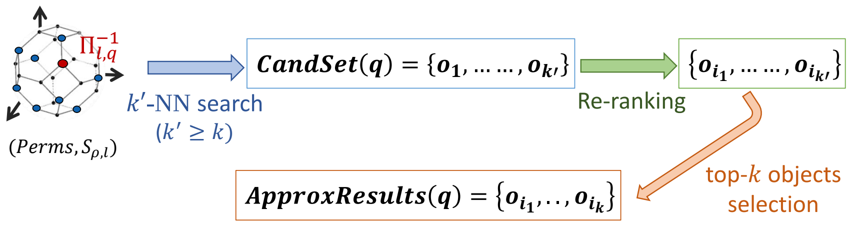

$\Pi^{-1}_{l,q}$
$(Perms, S_{\rho,l})$
k'-NN search
$(k' \geq k)$
$CandSet(q) = \{o_1, \ldots\ldots, o_{k'}\}$
Re-ranking
$\{o_{i_1}, \ldots\ldots, o_{i_{k'}}\}$
top-k objects selection
$ApproxResults(q) = \{o_{i_1}, .., o_{i_k}\}$

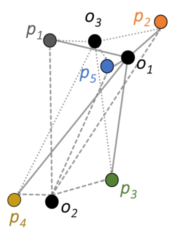
Figure 1: Illustration of the pipeline adopted in this work to compute the approximate k-NN results to a query by re-ranking a permutation-based candidate result set.

4. Re-ranking Permutation-Based Candidate Set

The permutation-based methods for approximate search are filter-and-refine approaches that map original data (D, d) into a permutation space. The permutation representations are used to identify a set of candidate results for a given query $q \in D$. The candidate results are then refined, typically by comparing the candidate objects with the query one according to the actual distance d. This refining approach based on the actual distance, however, requires to store the original data set and access to it at query time. In the following, we investigate the use of other refining approaches. The aim is improving the permutation-based results while getting rid of the original data set.

We focus on the k-NN search and we assume that the data objects are represented and indexed using permutation prefixes instead of the full-length permutations (as done in many PBI approach [2, 14, 16]). Let $CandSet(q)$, with $|CandSet(q)| = k' > k$, the set of candidate results selected using the permutation-based encoding. The candidate result set can be built, for example, by performing a k'-NN search in the permutation space (e.g. using the MI-File [14]) or by finding objects with a common permutation prefix (e.g. using the PP-codes [2]). The candidate result set is then refined by selecting the top-k candidate objects ranked according to a dissimilarity function (Figure 1). To use a dissimilarity function that does not require to access to the original data, we propose to re-rank the objects based of their distances to a set of pivots. In facts, the distances between the objects and the pivots are calculated when computing the permutation-based representation and therefore can be easily reused at query time.

Let $PivotSet(\Pi_{o,l})$ the set of the l closest pivots to the object o, i.e. the pivots whose identifiers appear in the prefix permutation $\Pi_{o,l}$. We assume that the distances between each object and its l closest pivots are stored and indexed within the object permutation prefix. This can be done with a slight modification of the used permutation-based index. Figure 2 shows a naive example for integrating the object-pivot distances into the posting lists, such as the ones used in the MI-file [14]. In the following, we assume that the objects are indexed



Permutation representations (permutation prefix length $l = 3$)		
$\Pi_{o_1} = [5,2,1,3,4]$	$\Pi_{o_2} = [4,3,5,1,2]$	$\Pi_{o_3} = [5,1,2,3,4]$
$\Pi^{-1}_{o_1} = \overset{p_1,p_2,p_3,p_4,p_5}{[3,2,4,5,1]}$	$\Pi^{-1}_{o_2} = \overset{p_1,p_2,p_3,p_4,p_5}{[4,5,2,1,3]}$	$\Pi^{-1}_{o_3} = \overset{p_1,p_2,p_3,p_4,p_5}{[2,3,4,5,1]}$
$\Pi^{-1}_{o_1,l} = \overset{p_1,p_2,p_3,p_4,p_5}{[3,2,4,4,1]}$	$\Pi^{-1}_{o_2,l} = \overset{p_1,p_2,p_3,p_4,p_5}{[4,4,2,1,3]}$	$\Pi^{-1}_{o_3,l} = \overset{p_1,p_2,p_3,p_4,p_5}{[2,3,4,4,1]}$

Posting Lists			
p_1	$(o_3, 2)$	$(o_1, 3)$	
p_2	$(o_1, 2)$	$(o_3, 3)$	
p_3	$(o_2, 2)$	$(o_3, 3)$	
p_4	$(o_2, 1)$		
p_5	$(o_1, 1)$	$(o_3, 1)$	$(o_2, 3)$

Posting Lists with distances			
p_1	$(o_3, 2, d(p_1, o_3))$	$(o_1, 3, d(p_1, o_1))$	
p_2	$(o_1, 2, d(p_2, o_1))$	$(o_3, 3, d(p_2, o_3))$	
p_3	$(o_2, 2, d(p_3, o_2))$	$(o_3, 3, d(p_3, o_3))$	
p_4	$(o_2, 1, d(p_4, o_2))$		
p_5	$(o_1, 1, d(p_5, o_1))$	$(o_3, 1, d(p_5, o_3))$	$(o_2, 3, d(p_5, o_2))$

Figure 2: Example of traditional posting lists and posting lists with distances generated to index three objects using five pivots and a prefix lenght $l = 3$

using this modified version of inverted files, however, the approaches presented in this paper can be extended to cope with different permutation-based indexes.

We propose to refine the candidate result set according to a dissimilarity function derived from the distance bounds provided either by the *Pivoted embedding* (Sec. 3.2) and the *nSimplex projection* (Sec. 3.3), since these metric mappings rely only on the object-pivot distances. Specifically, at query time, for each object $o \in CandSet(q)$ we approximate the actual distance $d(o, q)$ on the basis of the distances $d(q, p_j)$, $d(o, p_j)$ for $p_j \in \Gamma_{o,q} = PivotSet(\Pi_{q,l}) \cap PivotSet(\Pi_{o,l})$ as follows:

Pivoted embedding - As a consequence of Equation 4 we have

$$\max_{p_j \in \Gamma_{o,q}} |d(o, p_j) - d(q, p_j)| \leq d(o, q) \leq \min_{p_j \in \Gamma_{o,q}} |d(o, p_i) + d(q, p_i)| \qquad (6)$$

so we consider three possible re-rankings of the candidate objects, based on the following dissimilarity measures

$$P_{lwb}(o, q) = \max_{p_j \in \Gamma_{o,q}} |d(o, p_j) - d(q, p_j)| \qquad \text{lower-bound}$$

$$P_{upb}(o, q) = \min_{p_j \in \Gamma_{o,q}} (d(o, p_j) + d(q, p_j)) \qquad \text{upper-bound}$$

$$P_{mean}(o, q) = (P_{upb}(o, q) + P_{lwb}(o, q))/2 \qquad \text{mean}$$

Simplex projection - For each candidate object o, the pivots in $\Gamma_{o,q}$ are used to build the simplex base. The simplex base and the distances $d(o, p_j)$,

$d(q, p_j)$ with $p_j \in \Gamma_{o,q}$ are used to compute the apexes $\phi_{\Gamma_{o,q}}(o), \phi_{\Gamma_{o,q}}(q), \phi^-_{\Gamma_{o,q}}(q) \in \mathbb{R}^h$, where $h = |\Gamma_{o,q}| \leq l$. We consider the re-rankings of the candidate objects based on the following dissimilarity measures:

$$\begin{aligned} S_{lwb}(o,q) &= \ell_2(\phi_h(o), \phi_h(q)) && \text{lower-bound} \\ S_{upb}(o,q) &= \ell_2(\phi_h(o), \phi^-_h(q)) && \text{upper-bound} \\ S_{mean}(o,q) &= (S_{upb}(o,q) + S_{lwb}(o,q))/2 && \text{mean} \end{aligned}$$

Other dissimilarity functions over the apex vectors may be considered as well, in particular, any function that is always between the lower-bound and the upper-bound could be a good option since both the Simplex bounds asymptotically approach the true distance when increasing the number of pivots. In this work, we also consider the Zenith function, which was recently proposed in [28], that equals the quadratic mean of the lower- and upper-bounds

$$S_{zenith}(o,q) = \sqrt{\left(S_{upb}(o,q)^2 + S_{lwb}(o,q)^2\right)/2} \qquad \text{zenith}$$

The main difference between the mean and zenith distance is that the latter has a geometrical interpretation as the Euclidean distance between two vertex in $\mathbb{R}^{h+1}$ (see [28] for futher details).

Note that the Simplex bounds are highly affected by the number h of pivots used to build the simplex base (the higher h, the tighter the bounds), moreover the number h and the used simplex base change when changing the candidate object o. This means that the quality of the simplex-based approximation of the distance $d(o, q)$ may vary significantly when changing the considered candidate object. To overcome this issue, we also considered the re-ranking according to

$$\begin{aligned} S_{norm.mean}(o,q) &= S_{mean}(o,q)/g(h) && \text{normalized mean} \\ S_{norm.zenith}(o,q) &= S_{zenith}(o,q)/g(h) && \text{normalized zenith} \end{aligned}$$

where $g(h)$ is a normalization factor, further discussed in Section 5.3.

The lower-bounds S_{lwb} and P_{lwb} are metrics, while the other considered measures are dissimilarity functions. Finally, we remark that for all these approaches no new object-pivot distance is evaluated at either indexing or query time, since the used distances are already computed for building the permutation-based representations of the objects/query. Moreover, the distances $d(o, p_j)$ with $p_j \in \Gamma_{o,q}$ are retrieved while scanning the posting list to build the candidate result set, therefore the considered re-ranking approaches do not require further disk accesses in addition to the index accesses already made to find the candidate results.

5. Experiments

In this section, we experimentally evaluate the quality of the re-ranking approaches discussed above. We first describe the employed data sets (Section 5.1) and other experimental settings (Section 5.2). Then, we report results and their analysis for several case studies (Sections 5.3-5.5)

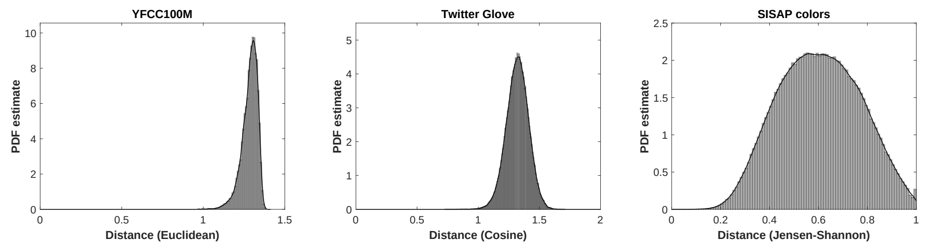
5.1. Test Data

The experiments were conducted on three publicly available data sets, namely YFCC100M [29], Twitter-Glove [30], and SISAP Colors [31]. To test our techniques on a variety of metric spaces, we selected a different type of data descriptor and metric for each data set:

YFCC100M is a collection that contains about 96M images, all uploaded to Flickr between 2004 and 2014 and published under a CC commercial or non-commercial license. As image descriptors we used deep Convolutional Neural Network features obtained from the activations of the *fc6* layer of the HybridNet [32] after the ReLu and the ℓ_2 normalization stages. The resulting features are 4,096-dimensional vectors. These feature were originally extracted by Amato et al. [33] and are available at `http://www.deepfeatures.org/`. We followed the common choice of using the *Euclidean distance* for comparing this kind of features.

Twitter-GloVe is a collection of 1.2M GloVe [30] features (word embeddings) trained on tweets. We used the 100-dimensional pre-trained word vectors available at `https://nlp.stanford.edu/projects/glove/`. These word vectors are often used as vocabulary terms to embed a document into a vector representation, for example by averaging the vectors of the terms contained in the text. In such cases, the space of the vocabulary terms is representative of the space of the document embeddings. The Euclidean distance or the Cosine similarity are typically used to compare two GloVe vectors since they provide an effective method for measuring the linguistic or semantic similarity of the corresponding words. In our experiments we used the *Cosine distance*, defined as $d_{\text{Cos}}(x,y) = \sqrt{1 - \frac{x \cdot y}{\|x\|_2 \|y\|_2}}$, which is equivalent to the Cosine similarity (i.e., the closest object to a query according to d_{Cos} are the most similar objects to the query according to the Cosine similarity).

SISAP Colors is a commonly used benchmark for metric indexing approaches. It contains about 113K feature vectors of dimensions 112, representing color histograms of medical images. We compare the feature vectors using the *Jensen-Shannon distance*, which is defined as the square root of the Jensen-Shannon divergence (JSDiv), i.e. $d_{\text{JS}}(x,y) = \sqrt{JSDiv(x,y)}$. The term Jensen-Shannon divergence is used variously with slightly different meanings in literature; to avoid ambiguity, we follow the definition used



(a) *YFCC100M (*`IDim`*=* 276*)* (b) *Twitter-GloVe (*`IDim`*=* 106*)* (c) *SISAP Colors (*`IDim`*=* 8*)*

Figure 3: Probability density function estimated on a sample of 500,000 distances.

in [8]:

$$JSDiv(x, y) = 1 - \frac{1}{2} \sum_i h(x_i) + h(y_i) - h(x_i + y_i),$$

where $h(z) = -z \log_2 z$.

The distance distributions of the three considered data sets are depicted in Figure 3. In the figure captions we also report the Intrinsic Dimentionality (`IDim`) of the data that was estimated as in [34], i.e. `IDim`$= \dfrac{\mu^2}{\sigma^2}$, where μ is the mean and σ is the standard deviation of the distances between the data objects.

5.2. Experimental setup

For each data set we build a ground-truth for the exact k-NN search related to 1,000 randomly-selected queries.[4] The ground-truths are used to evaluate the quality of the approximate results obtained by re-ranking a permutation-based result set of size $k' \geq k$. Specifically, for each query object we select a candidate result set by performing a k'-NN search in the permutation space. Then we re-rank the candidate results and we select the top-k objects as the approximate answer to the k-NN query.

The quality of the approximate results was evaluated using the *recall@k*, defined as $|\mathcal{R} \cap \mathcal{R}^A|/k$, where $\mathcal{R}$ is the result set of the exact k-NN search in the original metric space and $\mathcal{R}^A$ is the approximate result set. We set $k = 10$ and $k' = 100$, thus the candidate result set is computed by performing a 100-NN search in the permutation space.

The permutation-based representations of the data objects were generated using a total of $n = 4,000$ pivots for YFCC100M and Twitter-GloVe data, and

[4]The query objects were removed from the ground-truths of Twitter GloVe and SISAP Color data sets. For the YFCC100M we keep the query objects in the ground-truth to have results comparable with other research papers that used the same ground-truth (e.g. [19, 11]). Please also note that many re-ranking measures tested in this paper are not metrics, thus there are no guarantees that the less similar object to a query will be the query itself.

$n = 1,000$ pivots for the smaller SISAP Colors data set. In our tests, we used fixed-length permutation prefixes to represent the data objects. The permutation prefixes were compared using *Spearman's rho with location parameter* metric, where the location parameter is the length l of the permutation prefixes. Note that the case $l = n$ simply corresponds to use of the full-length permutations with the traditional Spearman's rho metric. In the experiments, we evaluate the performance for various permutation prefix lengths.

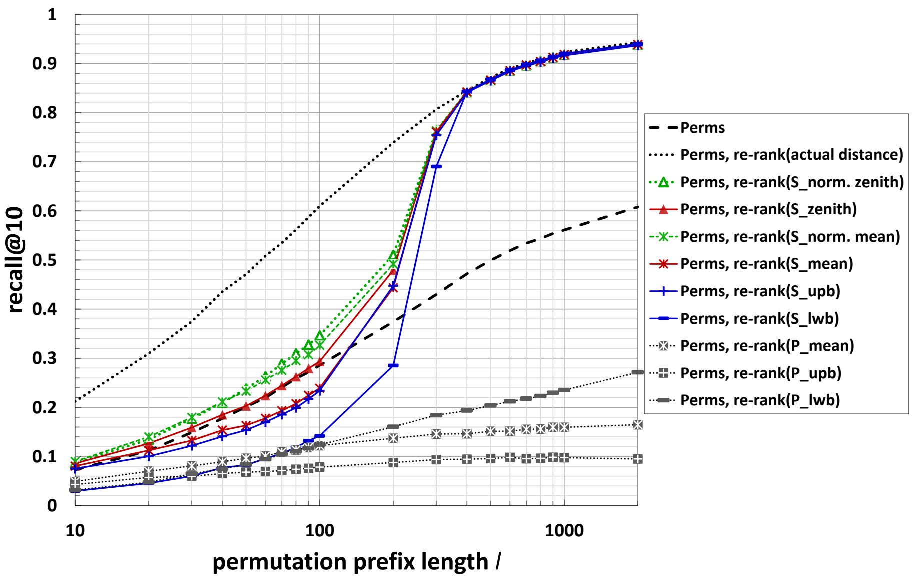
5.3. Results

This section presents comparative results for all the approaches described in Sec. 4 to re-rank a permutation-based candidate results set. We remind the reader that the considered techniques are based on various `Pivoted embedding` and `Simplex projection` dissimilarity measures (namely, lower-bound, upper-bound, and mean), as well as the `Simplex` zenith function. We compared these approaches also with two baselines: 1) the permutation-based results before any re-ranking, 2) the re-ranking based on the actual distance. The permutation-based results before any re-ranking are simply the first k candidate objects ordered according to their permutation-based distance to the query (i.e. the k-NN results in the permutation space). A good re-ranking technique in terms of effectiveness should at least improve the recall of the permutation-based results, and ideally achieves a performance close to that obtained using the re-ranking based on the actual distance. In fact, the latter one is the approach that provides the maximum possible recall for the given candidate result set, but it requires to access the original metric object o to compute the distance $d(q, o)$ between the query q and every candidate object o.

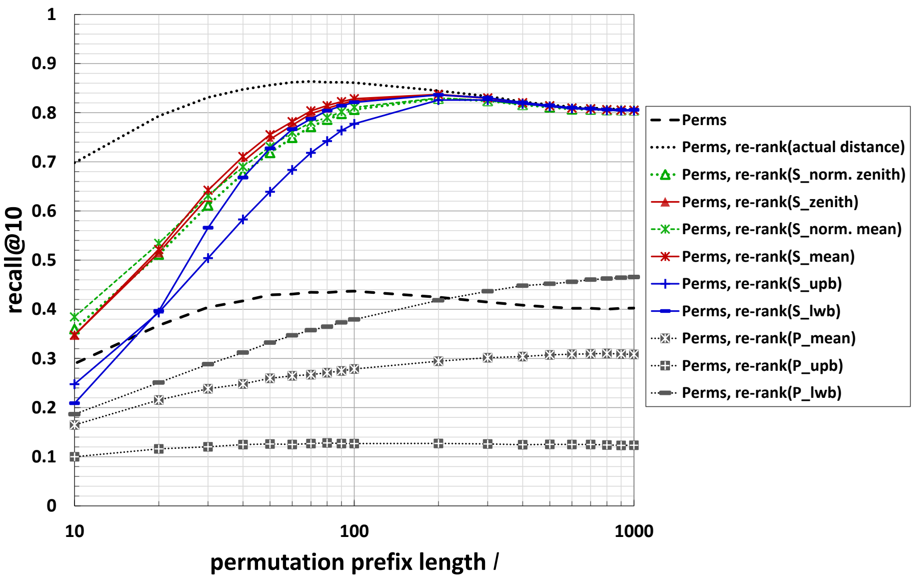
Figure 4 illustrates the results on Twitter-GloVe and SISAP Colors data sets. Figures 5a and 5b show the results on two subsets of YFCC100M that contain 1M and 10M images, respectively. We used the term "`Perms`"to indicate the permutation-based results before any re-ranking, and "`Perms, re-rank(f)`" for the re-ranking based on the measure f, where f may be either the

- actual distance d,
- `Pivoted embedding` measures (P_{lwb}, P_{upb}, P_{mean}),
- `Simplex` measures (S_{lwb}, S_{upb}, S_{mean}, S_{zenith}, $S_{norm.\,mean}$, $S_{norm.\,zenith}$).

In each graph, we report the *recall*@10 varying the length l of the permutation prefixes used to represent the data objects (the number n of pivots is fixed). Please note that the prefix length l influences the quality of the candidate set to be re-ranked, as well as the quality of the `Pivoted embedding` and `Simplex projection` distance approximations. In fact, for a fixed value l and for a candidate object o, the number h of pivots used to compute the distance approximations is less than l; moreover, it varies and depends on the candidate object o as it is equal to the cardinality of $\Gamma_{o,q}$ (i.e. the intersection between the query permutation prefix and the object permutation prefix). Typically h is greater for objects in top positions in the permutation-based result list and decrease for

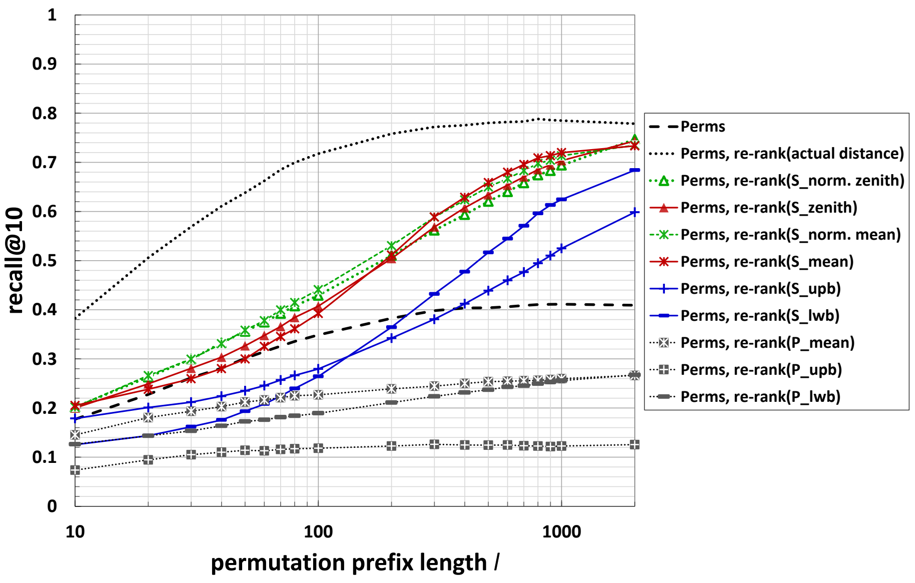


(a) `Twitter-GloVe, Cosine distance,` $n = 4,000$

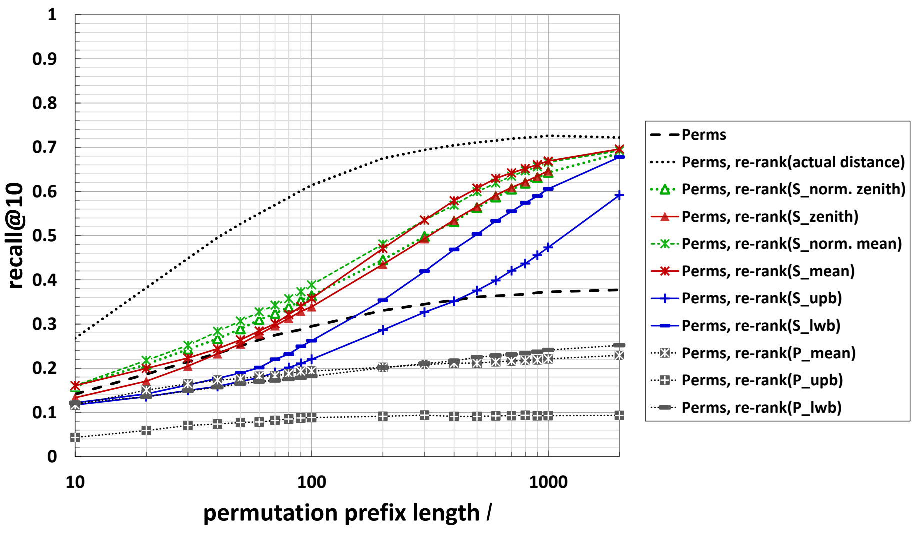


(b) `SISAP Colors, Jensen-Shannon distance,` $n = 1,000$

Figure 4: *Recall*@10 of several re-ranking approaches varying the permutation prefix length l. The candidate set to be reordered is selected with a 100-NN search in the permutation space using the Spearman's rho with location parameter l.

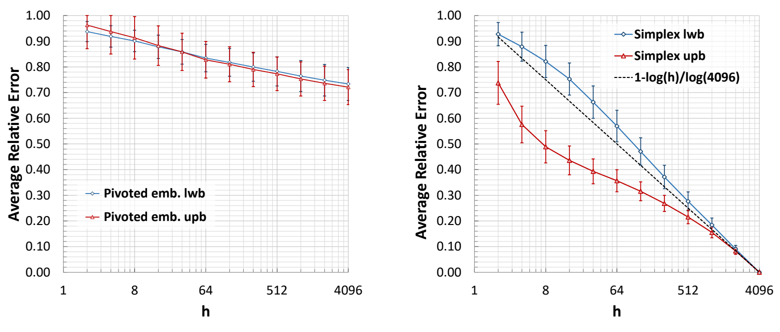


(a) Results on 1 million images



(b) Results on 10 million images.

Figure 5: `YFCC100M, Euclidean distance`: *Recall*@10 varying the permutation prefix length l on subsets of 1M images (5a) and 10M images (5b). The number of pivots is fixed to $n = 4,000$. The candidate set to be reordered is selected with a 100-NN search in the permutation space using the Spearman's rho with location parameter l.



(a) `Pivoted embedding` (b) `Simplex projection`

Figure 6: YFCC100M, Euclidean Distance - Average relative error of the `Pivoted embedding` and `Simplex embedding` bounds with respect to the actual distance varying the number of h of pivots used to compute the bounds. Similar trends are obtained on Twitter-GloVe and SISAP colors data sets.

objects that are far according to the permutation-based distance. Moreover, the greater the l, the greater the h and so the better the approximation bounds.

Surprisingly, we observed that in almost all the tested cases the `Pivoted embedding` approach greatly degrades the quality of the permutation-based results. Moreover, on YFCC100M and Twitter-GloVe it never reaches a *recall*@10 greater than 0.3. Hence, the `Pivoted` distance approximations resulted to be not adequate for the considered re-ranking purpose. One of the reasons for its poor performance is that the `Pivoted` lower-bound approximates well the actual distance $d(o, q)$ only if o and q are very close to each other in the original metric space, or if $\Gamma_{o,q}$ contains at least one pivot that is very close to q and far to o (or vice versa). However, for randomly selected pivots in high dimensional space this is unlikely to happen: for a random pivot p and for an object o not so close to q, we often have that the distances $d(o, p)$ and $d(q, p)$ are both close to the mean value in the distribution of the data distances, and so the lower-bound results to be close to zero. This means that when we use the `Pivoted` lower-bound for re-ranking purpose, it may happen that many objects are incorrectly swapped and far objects can be assigned in top-positions. In addiction, we observed that the `Pivoted` distance bounds have high relative errors with respect to the actual distance and that these errors slightly decrease when increasing the number h of pivots used to compute the bounds (Figure 6a).

The `Simplex` distance bounds showed similar drawbacks when using relatively small prefix lengths. In particular, they are mostly influenced by the fact that the `Simplex` bounds asymptotically approach the true distances when increasing the number h of pivots used to build the simplex base and that the tightness of the bounds highly depends on h. In fact, in all the tested cases, we observed that there exists a value $\tilde{h}$ for which the full convergence is achieved. This value is 4,096 for YFCC100M, and about 100 for Twitter-GloVe/SISAP Colors. The effect of the convergence of the `Simplex` bounds is evident in both the Twitter-GloVe (Figure 4a) and the SISAP Colors (Figure 4b) data: for $l > 300$ we observed that the number of pivots in the intersection $\Gamma_{o,q}$ starts to exceed $\tilde{h} = 100$ for most of the candidate objects o, and so all the `Simplex` bounds provide an exact or almost exact approximation of the actual distances. As a consequence, for $l > 300$ all the recall curves of the `Simplex`-based re-ranking approaches coincide with the recall curve obtained using the re-ranking based on the actual distance. For the YFC100M data set, instead, the `Simplex` bounds recall curves do not reach the values obtained by using the actual distance because we are considering prefix lengths smaller than the number $\tilde{h}$ of pivots needed to have the convergence.

We remark that the performance of the `Simplex` bounds are poor for *small prefix lengths* mainly because

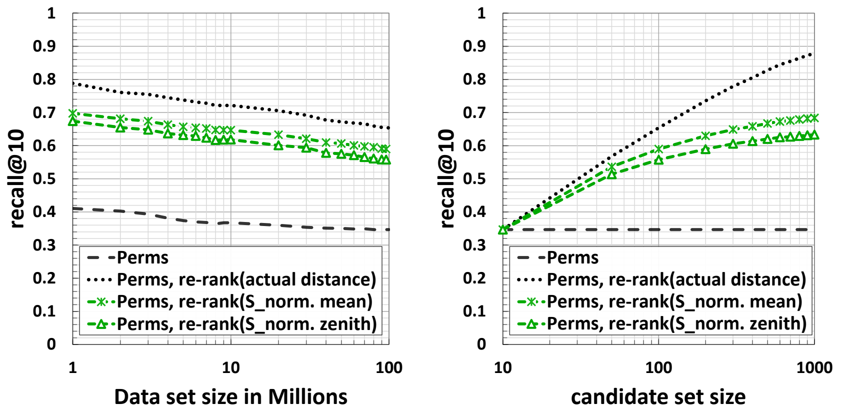
- for two objects $o, s \in CandSet(q)$ such that $|\Gamma_{o,q}| < |\Gamma_{s,q}| << \tilde{h}$ we may have $S_{lwb}(o,q) < S_{lwb}(s,q)$ even if $d(o,q) > d(s,q)$;
- the upper-bound, which is not a metric, particularly fails in approximate small distances and $S_{upb}(o,o)$ may be much greater than 0.

This behaviour of the S_{upb} is somehow observable in the recall values obtained on the YFFCC100 data: for small l the results of the re-ranking based on the S_{upb} are better than that of S_{lwb} on the 1M images subset (Fig. 5a), but on 10M images (Fig. 5b) the curve of S_{upb} never exceeds that of S_{lwb}. The reason is that the actual distances between the query and the candidate objects are likely to be smaller when performing the nearest neighbour search on a lager subset of data. Thus, given that for small l the S_{upb} does not approximate well tiny distances, the performance of the re-ranking based on the upper-bound drops when the candidate objects are selected by searching 10M images.

It is worth noting that, accordingly to our experimental observations, if we use the same simplex base (e.g. the one formed by the pivots in the query permutation prefix) to project all the candidate objects, we achieve re-ranking scores better than that showed in the Figures above, especially for relatively small prefix lengths. However, this approach is not directly applicable in the analysed scenario. In fact, we used inverted files to index the permutations and store the distances object-pivots. This implies that at query time, for each candidate object o we had access only to the distances $d(o,p)$ with p appearing in both the object and query permutation prefixes. Therefore, the set of pivots employed to build the simplex base changes when considering different candidate objects. This means that the "quality" (tightness) of the `Simplex`-based approximations of the query-object distances is not uniform within the set of the candidate objects. To overcome this issue, we tested normalised versions of all the `Simplex` distance bounds by taking into account the number h of pivots used for projecting the data. In the graphs reported in this paper, we show only the normalised version of the mean and zenith distance since they were the ones obtaining the best results. As normalisation factor we used $g(h) = \log(h)$ since we experimentally observed that the relative errors of the Simplex bounds decrease logarithmically with h (e.g. Figure 6b).

The re-rankings based on the normalised versions of the `Simplex` mean and zenith have practically the same performance on Twitter Glove and SISAP Colors data sets, while on YFCC100M data the $S_{norm.\,mean}$ shows slightly better recall values. Moreover, in all the tested cases, the re-rankings using those `Simplex` measures always improved the permutation-based results (i.e. the `Perms` baseline). For example, for $n = 4,000$ pivots and $l = 800$, the $recall$@10 is improved from 0.37 to 0.64 on YFCC100M (10M subset). For $n = 4,000$ and $l = 300$, the recall increases form 0.43 to 0.76 on Twitter-GloVe, while for $l = 80$ and $n = 1,000$ it raises from 0.43 to 0.80 on SISAP Colors. We provided examples with $l < n$ instead of considering the full-length permutation since when using inverted files the number of index blocks accessed is proportional to l^2/n and does not depend on the number of retrieved objects. Moreover, it is worth to note that the quality of permutation-based results it is not always improved by considering large prefix lengths. In fact, it often happens that there exists an optimal prefix length for which we achieve a recall that is better or very similar to that obtained using the full-length permutations. This phenomenon is observable in the YFCC100M and the SISAP Colors data set (see Fig. 5 and 4b), where the `Perms` recall line has a plateau or decreases after achieving a maximum value. Other examples of this phenomenon can be found e.g. in [18] where it was observed the existence of an optimal prefix length $l < n$ for some synthetic and real-word data sets. The intuition is that in those cases the intrinsic complexity of the data set is already well described when permutation prefixes with length equal to the optimal value are used, therefore increasing the length of the prefixes may add noisy information instead of improving the data representation. This phenomenon is not yet completely investigated in literature, however, we mentioned to clarify why in the SISAP Colors case the performance is affected by using a large l parameter.

Finally, we observe that the cost of the considered re-ranking approaches depends on the query object since it changes according to the numbers of pivots in the intersections of permutation prefixes of the query and the candidate objects. If using the algorithm proposed in [10], the cost for building a simplex base using h pivots is $O(h^3)$ floating point operations (flops), while the cost for projecting an object is $O(h^2)$ flops. Thus, for k' candidate objects whose permutation prefixes have on average h_l pivots in common with the query permutation we have a cost of $O(k'(h_l^3 + h_l^2) + h_l^2)$ flops to compute the `Simplex` bounds. However, the k' simplex bases can be computed in parallel since they referred to different sets of pivots. Just to provide an example, for $l = 300$ the time cost for computing all the simplex bases and projecting both the query and the candidate objects is about 300ms on an Intel i7 3.5 GHz. We also observe that this cost may be greatly reduced if some of these simplex bases are pre-computed or partially computed. In facts, if we have a simplex base built upon the pivot set $\{p_{i_1}, \ldots, p_{i_h}\}$ and we extend it by adding a further pivot the cost is $O(h^2)$ flops instead of $O(h^3)$ flops needed to build it from scratch. Thus implementations that exploit hashmaps or prefix trees to dynamically cash the computed simplex bases would accelerate the response at query time.



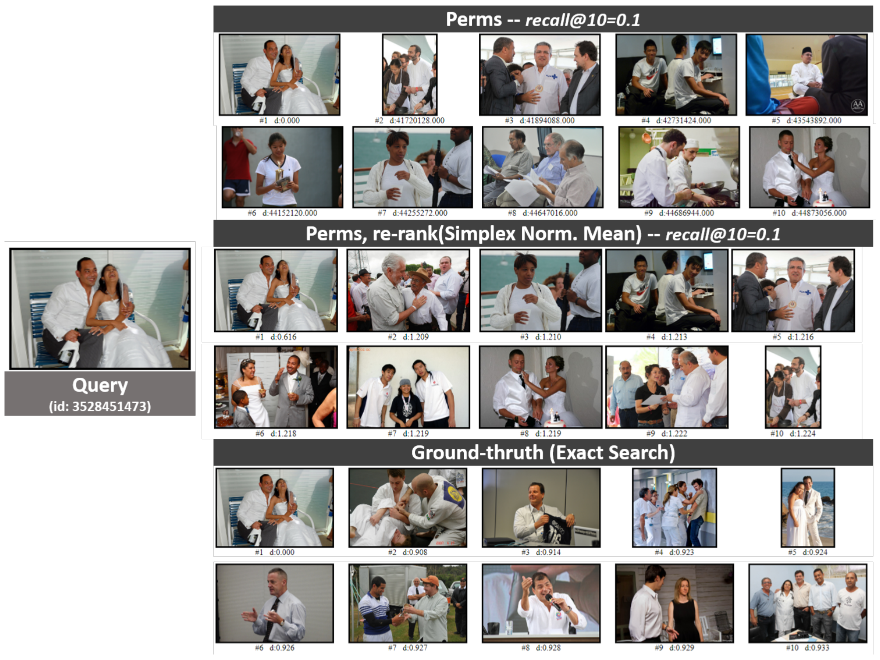
(a) Recall@10 varying the data set size (b) Recall@10 varying the candidate set size (full YFCC100M dataset)

Figure 7: `YFCC100M, Euclidean distance,` $n = 4000$, $l = 800$

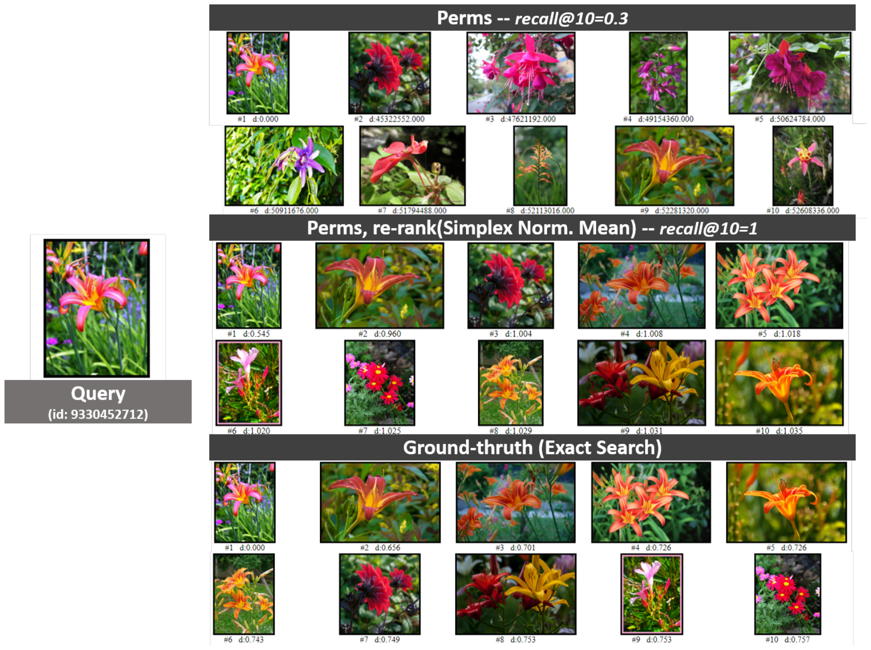
5.4. Results on large scale

To evaluate our techniques on large scale, we perform the k-NN search on various subsets of YFCC100M. Figure 7a shows the *recall*@10 varying the data set size from 1 to 96 million images. The candidate results to be re-ranked were selected by using permutation prefixes of length $l = 800$. We observe that the performance of our techniques with respect to the `Perms` and `re-rank(actual distance)` baselines is stable when increasing the size of the data set. In particular, the *relative improvement* in the recall obtained by the `Simplex` mean and zenith re-rankings with respect to `Perms` results ranges between 70% (at 96M) and 78% (at 8M). Moreover, for large sizes of the data set the relative gap between our techniques and the re-ranking based on the actual distance slightly decreases.

We also investigate the performance varying the size k' of the candidate set to be re-ranked (Figure 7b). In this case, the candidate set was selected by performing a k'-NN search in the permutation space using the Spearman's rho with location parameter $l = 800$. As expected, the gap between our approaches and the re-ranking based on the actual distance increases for large sizes of the candidate set due to the errors in approximating the actual distance by the `Simplex` measures. In facts, for the considered parameter $l = 800$, on average we have about 550 pivots in the intersection between the query and the object permutation prefixes, but for the YFCC100M data set the convergence of the Simplex bounds is achieved using 4,096 pivots. Thus, the effects of distance approximation errors becomes more evident when we re-rank larger set of data. Nevertheless, even when considering $k' = 1,000$ as candidate set size, the improvements in the recall of our approach with respect to the `Perms` baseline is considerable (from 0.35 to 0.59)


Perms -- recall@10=0.1
Perms, re-rank(Simplex Norm. Mean) -- recall@10=0.1
Ground-thruth (Exact Search)
Query
(id: 3528451473)

(a)


Perms -- recall@10=0.3
Perms, re-rank(Simplex Norm. Mean) -- recall@10=1
Ground-thruth (Exact Search)
Query
(id: 9330452712)

(b)

Figure 8: Examples of 10-NN search results obtained using the permutations without any re-ranking, our Simplex re-ranking technique, and the sequential scan (ground-truth).

Table 2: Examples of disk space in Gigabytes needed by various approaches to index and search the YFCC100M data set using inverted files. The re-ranking based on the Simplex measures requires to store the object-pivot distances within the permutation prefixes. The re-ranking based on the actual distance needs to store the permutation prefixes as well as the original data.

Approach	**Disk space (GB)**			
	1M deep features		100M deep features	
	$l = 300$	$l = 800$	$l = 300$	$l = 800$
`Perms`	0.7	1.9	94.3	251.5
`Perms,re-rank(actual dist.)`	16.0	17.1	1620.5	1777.7
`Perms,re-rank(Simplex)`	1.8	4.8	206.1	549.5
`Perms,re-rank(Simplex)` with distances quantized to 8 bits	1.0	2.6	122.2	326.0

Finally, in Figure 8, we report some examples of 10-NN results obtained on 10M images of YFCC100M using 1) the permutations without any re-ranking, 2) our re-ranking approach 3) brute-force (exact search via sequential scan). We considered the case $l = 800$ and we selected examples for which our technique achieved the worse and the best recall@10 over all the $1,000$ tested queries, which are 0.1 (Fig. 8a) and 1 (Fig. 8b) respectively. In the first example, it is interesting to note that even if the recall of our technique is very low, from a visual similarity point of view our results are not so worse than the ground-truth images. The second example shows a case in which we achieved the maximum recall (1.0), allowing us to highly improve the recall obtained by the permutation-based search (0.3). Note that in this case, even though the set of our results coincides with the ground-truth set, the ordering of the results is different because for $l = 800$ the `Simplex` bounds are not converged yet to the actual distance.

A demo of the K-NN search results on YFCC100M for various data set sizes (from 1 to 96M) and various re-ranking approaches is available at the link `http://cloudone.isti.cnr.it/rerankingPerms/`.

5.5. Results using quantized distances

In the experiments analyzed so far, the object-pivot distances used to perform the re-rankings were indexed within the object permutation prefixes by using inverted files (as discussed in Section 4. The disk space needed by the inverted index can be estimated in general assuming to encode each entry of the posting lists with $\lceil log_2|X|\rceil + 32$ bits, where $|X|$ is the size of the data set. This space is largely sufficient to encode both the ID of the object and its distance from the pivot corresponding to the list to which the entry belongs to. As observed in [14], the positions of the objects can be neglected by ordering the entries of the posting list according to the position of the objects. Therefore, for a fixed l, the size of the inverted index used by our approaches is $l|X|(\lceil log_2|X|\rceil + 32)$ bits. For reference we also observe that 1) the size of the inverted index of the `Perms` approach (i.e. the one that does not store the distances) is $l|X|(\lceil log_2|X|\rceil)$ bits; 2) the search approach relied on the re-ranking of the permutation-based results according to the actual distance requires to store both the `Perms` index and the original data set, thus it needs $l|X|(\lceil log_2|X|\rceil) + |X|(\lceil log_2|X|\rceil + D * 32)$ bits, if the data objects are D-dimensional real-valued vectors.

For example, the disk space required to index and search 1M deep fearures ($D = 4,096$) of YFCC100M using permutation prefixes of length $l = 300$ are about 1.8 GB for our techniques, 0.7 GB for the `Perms` approach, and 16 GB for the `Perms, re-rank(actual distance)` technique (see also Table 2).

To reduce the size of our inverted index, we investigated the idea of compressing the posting lists by storing quantized distances. Since the quantized distances are then used to compute the Simplex projection of the data objects, the performance of our re-ranking techniques may degrade due to the quantization errors. We investigated this aspect by testing several floating-point quantization approaches in conjunction with our Simplex-based re-ranking technique. Specifically, for a value x in a finite range (x_{min}, x_{max}) we tested the following scalar quantizers.[5]

Uniform quantizer is probably the simplest type of quantizer. It divides the interval (x_{min}, x_{max}) into L interval of the same length $Q = (x_{max} - x_{min})/L$. Each value x is then mapped to the middle value of the interval it belong to, i.e.,

$$q_{unif}(x) = x_{min} + Q/2 + Q\lfloor (x - x_{min})/Q \rfloor \tag{7}$$

Nonuniform quantizer are typically modeled as a cascade of a non-linear mapping (*compressor*) followed by a uniform quantizer followed by an inverse non-linear mapping (*expander*). The non-linear mapping before the uniform quantization allows us to keep the number of quantization interval constant while differentiating the size of those intervals to approximate the input better in certain regions (e.g. regions that have more probability mass). We considered the following nonuniform quantizers:

μ-law quantizer uses the μ-law mapping as compressor function, which is defined as

$$F_\mu(x) = V \frac{\log\left(1 + \mu \frac{|x|}{V}\right)}{\log(1 + \mu)} \mathrm{sign}(x) \tag{8}$$

where $V = \max\{|x_{min}|, |x_{max}|\}$, and μ is a compression parameter (e.g. $\mu = 255$ is used in the North American and Japanese standards for digital telecommunication signals). The transformed values are

[5] Note that in our case $x_{min} = 0$ since we are considering distance values.

then quantized using a uniform quantizer for the trasformed interval. Thus, the final quantized value associated to x is $q_{unif}(F_\mu(x))$.

To transform a quantized value y back, we use the inverse μ-law:

$$F_\mu^{-1}(y) = \frac{V}{\mu}\left((1+\mu)^{|y|/V} - 1\right)\operatorname{sign}(y). \tag{9}$$

Since the values close to zero are less compressed than values with greater absolute values (the quantization intervals increase logarithmically), usually the values are mean-centered before the quantization.

A-law quantizer uses the A-law compressor, which is defined as

$$F_A(x) = V\operatorname{sign}(x)\begin{cases}\dfrac{A|x|/V}{1+\ln(A)} & |x| < V/A\\ \dfrac{1+\ln(A|x|/V)}{1+\ln(A)} & V/A \le |x| \le V\end{cases} \tag{10}$$

where A is a compression parameter and $V = \max\{|x_{min}|, |x_{max}|\}$. The compressed values are then quantized using an uniform quatizer. The mapping used as expander is the inverse A-law:

$$F_A^{-1}(y) = \frac{V\operatorname{sign}(y)}{A}\begin{cases}\dfrac{|y|}{V}(1+\ln(A)) & |y| < \dfrac{V}{1+\ln(A)}\\ \exp(\dfrac{|y|}{V}(1+\ln(A)) - 1) & \dfrac{V}{1+\ln(A)} \le |y| < V\end{cases} \tag{11}$$

Also in this case the data are centered before the quantization.

The number L of interval we used for each quantizer is $L = 2^{\text{nBits}}$, where `nBits` are the number of bits used to store each distance. For each approach, we computed the *Mean Squared Error* (MSE) on a sample set of data. The MSE is a frequently used to evaluate how close are the (reconstructed) quantized values x' to the original values x, and it is defined as $\frac{1}{m}\sum_{i=1}^{m}(x_i - x_i')^2$, where m is the number of samples.

For the μ-law and A-law quantizer, we select the optimal μ and A parameters as the ones providing us the lowest MSE over a sample set of object-pivots distances. We then evaluate the recall obtained by re-ranking the permutation-based candidate set according to the Simplex bounds computed using the quantized distances. Table 3 shows comparative results on 1M subset of YFCC100M using a prefix length of $l = 800$ and `nBits` $= 8$.

As expected the uniform quantizer has really poor results since the distribution of the distances is not uniform. The tested nonuniform quantizers have results similar to each other. We decided to use the μ-law quantizer since it showed slightly better results.

We then thoroughly tested the performance of our technique by varying the data set, the permutation prefix length l, and the number of bits used to

Table 3: Results of various quantization approaches (YFCC100M data set). The quantizers are applied to the distances of the data objects to their l nearest pivots, i.e. the distances store in the posting lists. The reported results were obtained using the `Perms,re-rank(`$S_{norm.mean}$`)` approach, distance quantized to 8 bits, and permutation prefixes of length $l = 800$.

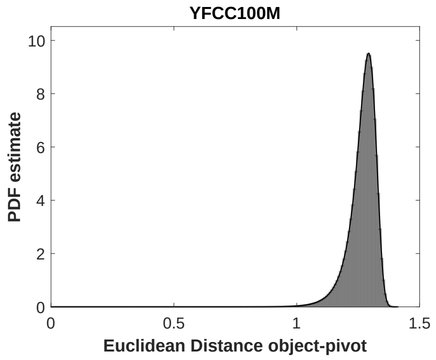
Quantizer	**Compressor parameter**	**MSE**	**recall@10**
Uniform		3.24E-03	0.228
μ-law	$\mu = 48$	2.64E-07	0.698
A-law	$A = 3$	1.04E-06	0.696
No quantization		0	0.698

store each object-pivot distance. Figures 9, 10, and 11 illustrate the results on YFCCC100M, SISAP Colors, and Twitter GloVe, respectively. For the sake of simplicity, we show results obtained using a fixed parameter μ for all tested permutation prefix lengths. This parameter was selected as the one minimising the MSE error in approximating the distances of the objects to all the n pivots. In facts, we observe that results obtained in this way are practically equivalent to that obtained by estimating an optimal parameter μ for each different choice of the parameter l.

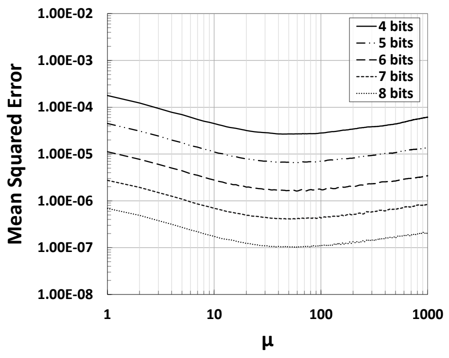
On YFCC100M (Euclidean distance) and SISAP Colors (Jensen-Shannon distance), we were able to satisfactorily preserve the quality of the re-ranked results when using at least 8 bits to store each distance. However, we observed a huge degradation when using fewer bits. For example, the re-ranked results became worse than the permutation-results when we use less than 5 bits. The problem is that the quantized object-pivot distances are then used to compute the Simplex projection of the object, so the quantization errors propagate in the Simplex-based estimation of the query-object distance. The effect of this error propagation is more evident in Twitter Glove data (cosine distance), where the results obtained for $l \geq 300$ is highly degraded using quantized distances. On this data set, we needed about 14 bits to preserve the quality of the re-ranked results even though the MSE errors related to distances quantized using fewer bits were in line with that obtained in the other tested data sets.

6. Conclusions

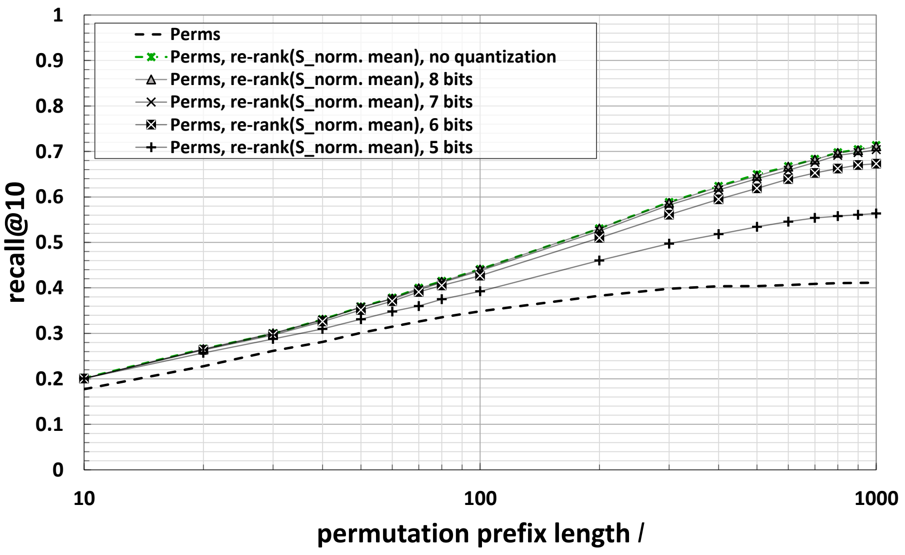
In this article, we presented an approach that exploit a pivot-based local embedding to refine a set of candidate results of a similarity query. The analysed case is refining of a set of approximate nearest neighbour results retrieved using a permutation-based search system. However, our approach can be generalized to other types of approximate search provided that they are based on the use of anchor objects (pivots) from which we pre-calculate the distances for other purposes. For example, some data structures use inverted indices, as the inverted multi-index [35], in which objects belonging to a Voronoi cell are inserted in a



(a) Probability distribution of the distances between the data objects and the n pivots

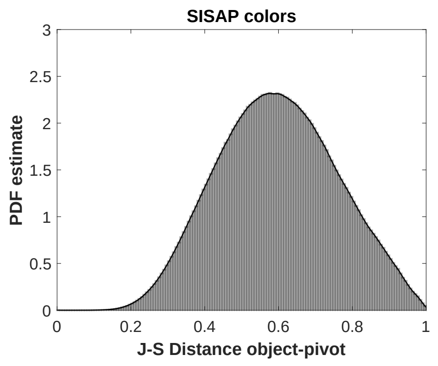


(b) Mean Squared Error for the μ-law quantization varying μ

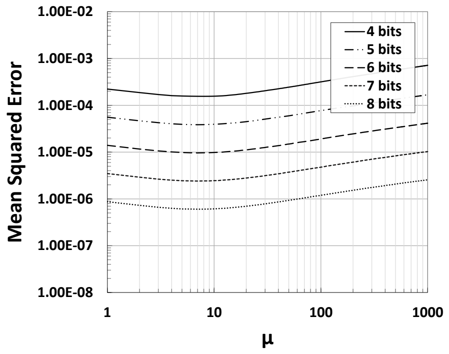


(c) *Recall*@10 varying the permutation prefix length l ($\mu = 57$)

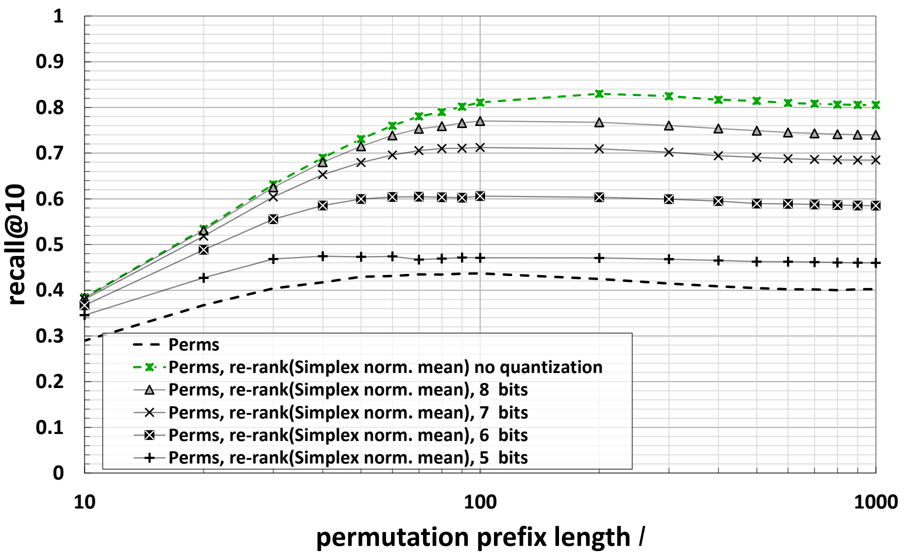
Figure 9: *YFCC100M (1M), Euclidean distance,* $n = 4,000$ *pivots*



(a) Probability distribution of the distances between the data objects and the n pivots

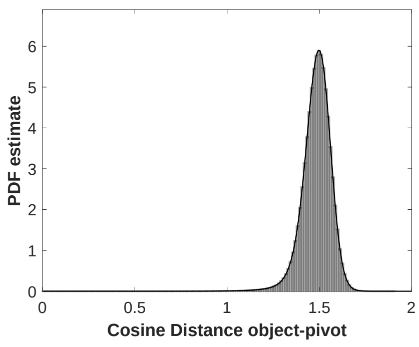


(b) Mean Squared Error for the μ-law quantization varying μ

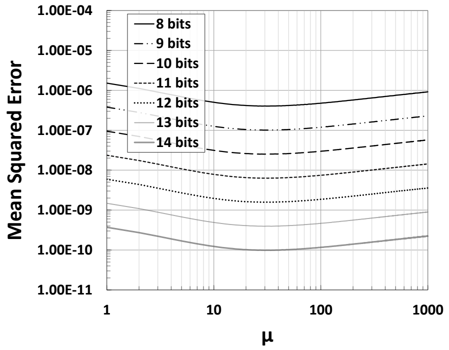


(c) *Recall*@10 varying the permutation prefix length l ($\mu = 3$)

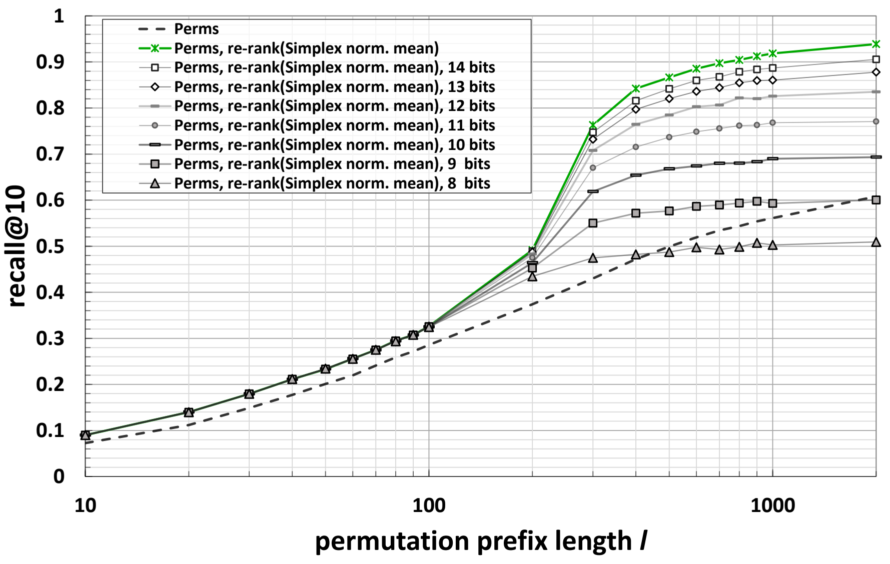
Figure 10: *SISAP colors, Jensen-Shannon distance,* $n = 1,000$



(a) Probability distribution of the distances between the data objects and the n pivots



(b) Mean Squared Error for the μ-law quantization varying μ



(c) *Recall*@10 varying the permutation prefix length l ($\mu = 40$)

Figure 11: *Twitter Glove, Cosine distance,* $n = 4,000$

posting list associated with the centroid of the cell from which we calculated the distance. Other indexes that can benefit from our approach are those based on permutation prefix trees, like PP-Index [2] and PPP-Index [16].

The core idea of the proposed technique is using the distances between an object and a set of pivots (pre-computed at indexing time) to embed the data objects in a metric space where it is possible to compute upper- and lower-bounds for the actual distance. Dissimilarity functions defined upon those bounds are then adopted for re-ranking the candidate objects. The main advantage is that the proposed approach do not need to access the original data as done, instead, by the most commonly used refining technique that relies on computing the actual distances between the query and each candidate object.

We analysed the refining based on two data embeddings, namely the Pivoted embedding and the nSimplex projection, and several dissimilarity functions derived by these space transformations. The refining approaches using the nSimplex projection resulted to be particularly effective for refining permutation-based results. For example, using the refining according to the nSimplex normalised mean function we were able to almost double the precision the permutation-based results even on a data set of about 100 million of objects.

Acknowledgements

The work was supported by Smart News, “Social sensing for breaking news” (CUP CIPE D58C15000270008), VISECH ARCO-CNR (CUP B56J17001330004), and AI4EU project (funded by the EC, H2020 - Contract n. 825619).

References

References

[1] P. Zezula, G. Amato, V. Dohnal, M. Batko, Similarity search: the metric space approach, Vol. 32, Springer Science & Business Media, 2006.

[2] A. Esuli, Use of permutation prefixes for efficient and scalable approximate similarity search, Information Processing & Management 48 (5) (2012) 889–902. doi:10.1016/j.ipm.2010.11.011.

[3] M. Patella, P. Ciaccia, Approximate similarity search: A multi-faceted problem, Journal of Discrete Algorithms 7 (1) (2009) 36–48. doi:10.1016/j.jda.2008.09.014.

[4] R. Weber, H.-J. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, in: Proceedings of 24rd International Conference on Very Large Data Bases (VLDB'98), Vol. 98, Morgan Kaufmann, 1998, pp. 194–205.

[5] B. Naidan, L. Boytsov, E. Nyberg, Permutation search methods are efficient, yet faster search is possible, Proceedings of the 41st International Conference on Very Large Data Bases (VLDB) 8 (12) (2015) 1618–1629. doi:10.14778/2824032.2824059.

[6] V. Pestov, Indexability, concentration, and vc theory, Journal of Discrete Algorithms 13 (2012) 2–18. doi:10.1016/j.jda.2011.10.002.

[7] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, A. El Abbadi, Approximate nearest neighbor searching in multimedia databases, in: Proceedings 17th International Conference on Data Engineering (ICDE 2001), IEEE Computer Society, 2001, pp. 503–511. doi:10.1109/ICDE.2001.914864.

[8] R. Connor, F. A. Cardillo, L. Vadicamo, F. Rabitti, Hilbert Exclusion: Improved metric search through finite isometric embeddings, ACM Transactions on Information Systems (TOIS) 35 (3) (2016) 17:1–17:27. doi:10.1145/3001583.

[9] R. Connor, L. Vadicamo, F. A. Cardillo, F. Rabitti, Supermetric search, Information Systems 80 (2019) 108–123. doi:10.1016/j.is.2018.01.002.

[10] R. Connor, L. Vadicamo, F. Rabitti, High-dimensional simplexes for supermetric search, in: Proceedings of 10th International Conference on Similarity Search and Applications (SISAP 2017), Lecture Notes in Computer Science, Springer International Publishing, 2017, pp. 96–109. doi:10.1007/978-3-319-68474-1_7.

[11] G. Amato, E. Chávez, R. Connor, F. Falchi, C. Gennaro, L. Vadicamo, Reranking permutation-based candidate sets with the n-Simplex projection, in: Proceedings of 10th International Conference on Similarity Search and Applications (SISAP 2018), Lecture Notes in Computer Science, Springer International Publishing, 2018, pp. 3–17. doi:10.1007/978-3-030-02224-2_1.

[12] G. Amato, P. Savino, Approximate similarity search in metric spaces using inverted files, in: Proceedings of 3rd International ICST Conference on Scalable Information Systems (INFOSCALE 2008), ICST / ACM, 2008, pp. 28:1–28:10. doi:10.4108/ICST.INFOSCALE2008.3486.

[13] E. Chávez, K. Figueroa, G. Navarro, Effective proximity retrieval by ordering permutations, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (9) (2008) 1647–1658. doi:10.1109/TPAMI.2007.70815.

[14] G. Amato, C. Gennaro, P. Savino, MI-File: Using inverted files for scalable approximate similarity search, Multimedia Tools and Applications (3) (2014) 1333–1362. doi:10.1007/s11042-012-1271-1.

[15] E. S. Tellez, E. Chávez, G. Navarro, Succinct nearest neighbor search, Information Systems 38 (7) (2013) 1019–1030. doi:10.1016/j.is.2012.06.005.

[16] D. Novak, P. Zezula, PPP-Codes for large-scale similarity searching 24 (2016) 61–87. doi:10.1007/978-3-662-49214-7_2.

[17] E. Chávez, M. Graff, G. Navarro, E. S. Téllez, Near neighbor searching with K nearest references, Information Systems 51 (2015) 43–61. doi:10.1016/j.is.2015.02.001.

[18] G. Amato, F. Falchi, F. Rabitti, L. Vadicamo, Some theoretical and experimental observations on permutation spaces and similarity search, in: Proceedings of 7th International Conference on Similarity Search and Applications (SISAP 2014), Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 37–49. doi:10.1007/978-3-319-11988-5_4.

[19] G. Amato, F. Falchi, C. Gennaro, L. Vadicamo, Deep Permutations: Deep convolutional neural networks and permutation-based indexing, in: Proceedings of 9th International Conference on Similarity Search and Applications (SISAP 2016), Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 93–106. doi:10.1007/978-3-319-46759-7_7.

[20] K. Figueroa, R. Paredes, N. Reyes, New permutation dissimilarity measures for proximity searching, in: Proceedings of 11th International Conference on Similarity Search and Applications (SISAP 2018), Lecture Notes in Computer Science, Springer International Publishing, 2018, pp. 122–133. doi:10.1007/978-3-030-02224-2_10.

[21] M. L. Micó, J. Oncina, E. Vidal, A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements, Pattern Recognition Letters 15 (1) (1994) 9–17. doi:10.1016/0167-8655(94)90095-7.

[22] R. Connor, L. Vadicamo, F. A. Cardillo, F. Rabitti, Supermetric search with the four-point property, in: Proceedings of 9th International Conference on Similarity Search and Applications (SISAP 2016), Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 51–64. doi:10.1007/978-3-319-46759-7_4.

[23] L. M. Blumenthal, Theory and applications of distance geometry, Clarendon Press, 1953.

[24] I. J. Schoenberg, Metric spaces and completely monotone functions, Annals of Mathematics 39 (4) (1938) 811–841.

[25] R. Fagin, R. Kumar, D. Sivakumar, Comparing top k lists, SIAM Journal on discrete mathematics 17 (1) (2003) 134–160.

[26] L. Vadicamo, Enhancing content-based image retrieval using aggregation of binary features, deep learning, and supermetric search, Ph.D. thesis, University of Pisa (2018).
URL `https://etd.adm.unipi.it/theses/available/etd-04242018-161334/`

[27] M. Deza, H. Maehara, Metric transforms and euclidean embeddings, Transactions of the American Mathematical Society 317 (2) (1990) 661–671. doi:10.1090/S0002-9947-1990-0974513-6.

[28] R. Connor, A. Dearle, L. Vadicamo, Modelling string structure in vector spaces, in: Proceedings of the 27th Italian Symposium on Advanced Database Systems (SEBD 2019), CEUR-WS.org, 2019.
URL `http://ceur-ws.org/Vol-2400/paper-45.pdf`

[29] B. Thomee, B. Elizalde, D. A. Shamma, K. Ni, G. Friedland, D. Poland, D. Borth, L.-J. Li, YFCC100M: The new data in multimedia research, Communications of the ACM 59 (2) (2016) 64–73. doi:10.1145/2812802.

[30] J. Pennington, R.Socher, C. D. Manning, GloVe: Global Vectors for Word Representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), ACL, 2014, pp. 1532–1543.
URL `http://www.aclweb.org/anthology/D14-1162`

[31] K. Figueroa, G. Navarro, E. Chávez, Metric spaces library, `www.sisap.org/library/manual.pdf` (2007).

[32] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, A. Oliva, Learning deep features for scene recognition using places database, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Curran Associates, Inc., 2014, pp. 487–495.

[33] G. Amato, F. Falchi, C. Gennaro, F. Rabitti, YFCC100M-HNfc6: a large-scale deep features benchmark for similarity search, in: Proceedings of 9th International Conference on Similarity Search and Applications (SISAP 2016), Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 196–209. doi:10.1007/978-3-319-46759-7_15.

[34] E. Chávez, G. Navarro, R. Baeza-Yates, J. L. Marroquín, Searching in metric spaces, ACM computing surveys (CSUR) 33 (3) (2001) 273–321. doi:10.1145/502807.502808.

[35] A. Babenko, V. S. Lempitsky, The inverted multi-index, IEEE transactions on pattern analysis and machine intelligence 37 (6) (2015) 1247–1260. doi:10.1109/TPAMI.2014.2361319.
URL `https://doi.org/10.1109/TPAMI.2014.2361319`