Accurate and Fast Retrieval for Complex Non-Metric Data via Neighborhood Graphs[⋆]

Leonid Boytsov[1] and Eric Nyberg[2]

[1] Carnegie Mellon University, Pittsburgh, PA, srchvrs@cs.cmu.edu
[2] Carnegie Mellon University, Pittsburgh, PA, enh@cs.cmu.edu

Abstract. We demonstrate that a graph-based search algorithm—relying on the construction of an approximate neighborhood graph—can directly work with challenging *non-metric* and/or *non-symmetric* distances without resorting to metric-space mapping and/or distance symmetrization, which, in turn, lead to substantial performance degradation. Although the straightforward metrization and symmetrization is usually ineffective, we find that constructing an index using a modified, e.g., symmetrized, distance can improve performance. This observation paves a way to a new line of research of designing index-specific graph-construction distance functions. This is an archival version, the publisher's version is available at Springer.com.

Keywords: k-NN search, non-metric distance, neighborhood graph

1 Introduction and Problem Definition

In this paper we focus on k nearest neighbor (k-NN) search, which is a widely used computer technology with applications in machine learning, data mining, information retrieval, and natural language processing. Formally, we assume to have a possibly *infinite* domain containing objects x, y, z, $\ldots$, which are commonly called data points or simply points. The domain—sometimes called a *space*—is equipped with with a *distance function* $d(x, y)$, which is used to measure dissimilarity of objects x and y. The value of $d(x, y)$ is interpreted as a degree of dissimilarity. The larger is $d(x, y)$, the more dissimilar points x and y are.

Some distances are non-negative and become zero only when x and y have the highest possible degree of similarity. The *metric* distances are additionally symmetric and satisfy the triangle inequality. However, in general, we do not impose any restrictions on the value of the distance function (except that smaller values represent more similar objects). Specifically, the value of the distance function can be negative and negative distance values indicate higher similarity than positive ones.

We further assume that there is a data *sub*set D containing a *finite* number of domain points and a set of queries that belongs to the domain but not to D. We then consider a standard top-k retrieval problem. Given a query q it consists in finding k data set points

[⋆] This work was accomplished while Leonid Boytsov was a PhD student at CMU. Authors gratefully acknowledge the support by the NSF grant #1618159: "Matching and Ranking via Proximity Graphs: Applications to Question Answering and Beyond".

$\{x_i\}$ with smallest values of distances to the query among all data set points (ties are broken arbitrarily). Data points $\{x_i\}$ are called *nearest neighbors*. A search should return $\{x_i\}$ in the order of increasing distance to the query. If the distance is not symmetric, two types of queries can be considered: *left* and *right* queries. In a *left* query, a data point compared to the query is always the first (i.e., the left) argument of $d(x, y)$. Henceforth, for simplicity of exposition we consider only the case of left queries.

Exact methods degenerate to a brute-force search for just a dozen of dimensions [35]. Due to diversity of properties, non-metric spaces lack *common* and *easily identifiable* structural properties such as the triangle inequality. There is, therefore, little hope that *fully* generic exact search methods can be devised. Thus, we focus on the *approximate* version of the problem where the search may miss some of the neighbors, but it may not change the order. The accuracy of retrieval is measured via recall (equal to the average fraction of neighbors found). We cannot realistically devise fast exact methods, but we still hope that our approximate methods are quite *accurate* having a recall close to 100%.

There has been a staggering amount of effort invested in designing new and improving existing k-NN search algorithms (see e.g., [8,29,30,34]). This effort has been placed disproportionately on techniques for *symmetric metric* distances, in particular, on search methods for the Euclidean space. Yet, search methods for challenging non-symmetric and non-metric spaces received very *little* attention. A *filter-and-refine* approach is a common way to deal with an unconventional distance. To this end one would map data to a low-dimensional Euclidean space. The goal is to find a mapping without large distortion of the original similarity measure [17,14]. Jacobs et al. [17] review various projection methods and argue that such a coercion is often against the nature of a similarity measure, which can be, e.g., intrinsically non-symmetric. Yet, they do not provide experimental evidence. We fill this gap and demonstrate that both metric learning and distance symmetrization are, indeed, suboptimal approaches.

Alternatively the metric distance can be learned from scratch [3]. In that, Chechik et al. [9] contended that in the task of distance learning enforcing symmetry and metricity is useful only as a means to prevent overfitting to a small training set. However, when training data is abundant, it can be more efficient and more accurate to learn the distance function in an unconstrained bilinear form. Yet, this approach does not necessarily results in a symmetric metric distance [9]. We, in turn, demonstrate that a graph-based retrieval algorithm—relying on the construction of approximate neighborhood/proximity graphs—can deal with challenging non-metric distances directly without resorting to a low-dimensional mapping or full symmetrization. In that, unlike prior work [24,25], as we show in § 3, several of our distances are substantially non-symmetric.

Whereas the filter-and-refine symmetrization approach is detrimental, we find that constructing an index using the symmetrized distance can improve results. Furthermore, we show that the index construction algorithm can be quite sensitive to the order of distance function arguments. In most cases, changing the argument order is detrimental. However, this is not a universal truth: Quite surprisingly, we observe small improvements in some cases by building the graph using the argument-reversed distance function. We believe this observations motivates the line of research to design indexing distance functions—different from original distance functions—that result in better performance.

The remaining paper contains the description of employed retrieval algorithms and related experimental results.

2 Methods and Materials

2.1 Retrieval Algorithms

We consider two types of retrieval approaches: the filter-and-refine method using brute-force search and indexing using the graph-based retrieval method Small World Graph (SW-graph) [22]. In the filter-and-refine approach, we use a proxy distance to generate a list of k_c candidate entries (closest to the query with respect to the proxy distance) via the brute-force, i.e., exhaustive, search. For k_c candidate entries x_i we compute the true distance values $d(x_i, q)$—or $d(q, x_i)$ for right queries—and select k closest entries.

The filter-and-refine approach can be slow even if the proxy distance is quite cheap [24], whereas indexing can dramatically speed up retrieval. In particular, state-of-the-art performance can be achieved by using graph-based retrieval methods, which rely on the construction of an exact or approximate *neighborhood* graph (see, e.g., [2,24]). The neighborhood graph is a data structure in which data points are associated with graph nodes and sufficiently close nodes are connected by edges. A search algorithm is a graph-traversal routine exploiting a property "the closest neighbor of my closest neighbor is my neighbor as well." The neighborhood graph is often defined as a directed graph [12,11], where the edges go from a vertex to its neighbors (or vice versa), but undirected edges have been used too [22,20] (undirected nodes were also quietly introduced in kgraph[3]). In a recent study, the use of undirected neighborhood graphs lead to a better performance [20].

Constructing an *exact* neighborhood graph is hardly feasible for a large high-dimensional data set, because, in the worst case, the number of distance computations is $O(n^2)$, where n in the number of data points. An *approximate* neighborhood graph can be constructed substantially more efficiently [22,11]. To improve performance, one can use various graph pruning methods [20,23,13]: In particular, it is not useful to keep neighbors that are close to each other [20,13].

Neighborhood graphs have a long history. Toussaint published a pioneering paper where he introduced neighborhood graphs on the plane in 1980 [33]. Arya and Mount were first to apply neighborhood graphs to the problem of k-NN search in a high-dimensional space [1]. Houle and Sakuma proposed the first hierarchical, i.e., multi-layer, variant of the neighborhood graph called SASH, where data points at layer i are connected only to the nodes at layer $i + 1$ [15]. Malkov and Yashunin proposed an efficient multi-layer neighborhood-graph method called a Hierarchical Navigable Small World (HNSW) [23]. It is a generalization and improvement of the previously proposed method navigable Small World (SW-graph) [22], which has been shown to be quite efficient in the past [22,24]

Although there are different approaches to construct a neighborhood graphs, all retrieval strategies known to us rely on a simple semi-greedy graph-traversal algorithm with (possibly) multiple restarts. Such an algorithm keeps a priority queue of elements,

[3] https://github.com/aaalgo/kgraph

Name	max. # of rec.	Dimensionality	Source
RandHist-d	0.5×10^6	$d \in \{8, 32\}$	Histograms sampled uniformly from a simplex
RCV-d	0.5×10^6	$d \in \{8, 128\}$	d-topic LDA [4] RCV1 [19] histograms
Wiki-d	2×10^6	$d \in \{8, 128\}$	d-topic LDA [4] Wikipedia histograms
Manner	1.46×10^5	1.23×10^5	Question and answers from L5 collection in Yahoo WebScope

Table 1: Data sets

which ranks candidates in the order of increasing distance to the query. At each step, the search retrieves one or more elements from the queue that are closest to the query and explores their neighborhoods. Previously unseen elements may be added to the queue. For a recent experimental comparison of several retrieval approaches see [32].

Although, HNSW is possibly the best retrieval method for generic distances [23,20], in our work we use a modified variant of SW-graph, where retrieval starts from a single point (which is considerably more efficient compared to multiple starting points). The main advantage of HNSW over the older version of SW-graph is due to (1) introduction of pruning heuristics, (2) using a single starting point during retrieval. We want to emphasize that comparison of HNSW against SW-graph in [23] is not completely fair, because it basically uses an undertuned SW-graph. Furthermore, gains from using a hierarchy of layers are quite small: see Fig. 3-5 from [23]. At the same time pruning heuristics introduce another confounding factor in measuring the effect of distance symmetrization (and proxying), because symmetrization method used in the pruning approach can be different from the symmetrization method used by k-NN search employed at index time. Thus—as we care primarily about demonstrating usefulness (or lack thereof) of different distance modifications during construction of the graph rather than merely achieving maximum retrieval efficiency—we experiment with a simpler retrieval algorithm SW-graph. The employed algorithm has three main parameters. Parameter `NN` influences (but does not define directly) the number of neighbors in the graph. Parameters `efConstruction` and `efSearch` define the depth of the priority queue used during index and retrieval stages, respectively.

2.2 Data sets and Distances

In our experiments, we use the following distances (see Table 2): KL-divergence, the Itakura-Saito distance, the Rényi divergence, and BM25 similarity [28]. The first three distances are statistical distances defined over probability distributions. Statistical distances in general and, KL divergence in particular, play an important role in ML [7,31]. Both the KL-divergence and the Itakura-Saito distances were used in prior work [7]. BM25 similarity is a popular and effective similarity metric commonly used in information retrieval. It is a variant of a TF×IDF similarity computed as

$$\sum_{x_i = y_i} \mathrm{TF}_q(x_i) \cdot \mathrm{TF}_d(y_i) \cdot \mathrm{IDF}(y_i), \tag{1}$$

Denotation/Name	d(x,y)	Notes
Kullback-Leibler diverg. (KL-div.) [18]	$\sum\limits_{i=1}^{m} x_i \log \frac{x_i}{y_i}$	
Itakura-Saito distance [16]	$\sum\limits_{i=1}^{m} \left[\frac{x_i}{y_i} - \log \frac{x_i}{y_i} - 1\right]$	
Rényi diverg. [27]	$\frac{1}{\alpha-1} \log \left[\sum\limits_{i=1}^{m} x_i^{\alpha} y_i^{1-\alpha}\right], 0 < \alpha < \infty$	We use $\alpha \in 0.25, 0.75, 2$
BM25 similarity [28]	$-\sum_{x_i=y_i} \mathrm{TF}_q(x_i) \cdot \mathrm{TF}_d(y_i) \cdot \mathrm{IDF}(y_i)$	$\mathrm{TF}_q(x)$ and $\mathrm{TF}_d(y)$ are (possibly scaled) term frequencies in a query and document.

Table 2: Distance Functions

where $\mathrm{TF}_q(x)$ and $\mathrm{TF}_d(y)$ are term frequencies of terms x and y in a query and a document, respectively. IDF is an inverse document frequency (see [28] for more details). When we use BM25 as a distance, we take the negative value of this similarity function. Although BM25 is expressed as an inner product between query and document TF$\times$IDF vectors, this distance is *not* symmetric. Term frequencies are computed differently for queries and documents and the value of the similarity normally changes when we swap function arguments.

The Rényi divergence is a single-parameter family of distances, which are not symmetric when the parameter $\alpha \neq 0.5$. By changing the parameter we can vary the degree of symmetry. In particular, large values of α as well as close-to-zero values result in highly non-symmetric distances. This flexibility allows us to stress-test retrieval methods by applying them to challenging non-symmetric distances.

The data sets are listed in Table 1. Wiki-d and RCV-d data sets consists of dense vectors of topic histograms with d topics. RCV-d set are created by Cayton [7] from the RCV1 newswire collection [19] using the latent Dirichlet allocation (LDA) method [4]. These data sets have only 500K entries. Thus, we created larger sets from Wikipedia following a similar methodology. RandHist-d is a synthetic set of topics sampled uniformly from a d-dimensional simplex.

The Manner data set is a collection of TF$\times$IDF vectors generated from data set L5 in Yahoo WebScope[4]. L5 is a set of manner, e.g., how-to, questions posted on the Yahoo answers webite together with respective answers. Note that we keep only a single best answer—as selected by a community member—for each question.

3 Experiments

We carry out two experimental series. In the first series, we test the efficacy of the filter-and-refine approach (using collection subsets) where the distance function is obtained via metrization or symmetrization of the original distance. One of the important

[4] https://webscope.sandbox.yahoo.com

objectives of this experimental series is to demonstrate that unlike some prior work [24,25] we deal with *substantially non*-symmetric data. In the second series, we carry out a fully-fledged retrieval experiment using SW-graph [22] with different index- and query-time symmetrization approaches. Overall, we have 31 combination of data sets and distance functions (see § 2.2). However, due to space limitations, we had to omit some experimental results and minor setup details. A fuller description is available in §2.3.2 of the unpublished tech report [5].

Proxying Distance via Metrization and Symmetrization In this section, we use a proxy distance function to generate a list of k_c candidates, which are compared directly to the query. The candidate generation step employs an *exact* brute-force k-NN search with the proxy distance. On one hand, the larger is k_c, the more likely we find all true nearest neighbors. On the other hand, increasing k_c entails a higher computational cost. We consider two types of proxy distances: a learned distance (which is a metric in four out of five cases), and a symmetrized version of the original non-symmetric distance.

Distance learning We considered five approaches to learn a distance and a pseudo-learning approach where we simply use the Euclidean L_2 distance as a proxy. Computing L_2 between data points is a strong baseline, which sometimes outperforms true distance learning methods, especially for high-dimensional data. Four of the distance-learning methods [36,10,26,21] learn a global linear transformation of the data, which is commonly referred to as the Mahalanobis metric learning. The value of the L_2 distance between transformed vectors is used as a proxy distance function. The learned distance, is clearly a metric. We also use a non-linear Random Forest Distance (RFD) method that employs a random-forest classifier [37] and produces generally non-metric, but symmetric, distance. Note that we do not learn a distance function for the Manner data set that contains extremely high dimensional sparse TF×IDF vectors.

In all cases, the distance is trained as a classifier that learns to distinguish between close and distant data points. More specifically, we create sets of positive and negative examples. A positive example set contains pairs of points that should be treated as similar, i.e., near points, while the negative example set contains pairs of points that should be treated as dissimilar ones. The underlying idea is to learn a distance that (1) pulls together points from the positive example set and (2) pushes points from the negative example set apart. More details are given in [5].

Symmetrization Given a non-symmetric distance, there are two folklore approaches to make it symmetric, which use the value of the original distance $d(x, y)$ as well as the value of the distance function obtained by reversing arguments: $d_{\text{reverse}}(x, y) = d(y, x)$. Informally, we call the latter an *argument-reversed* distance. In the case of an average-based symmetrization, we compute the symmetrized distance as an average of the original and argument-reversed distances:

$$d_{\text{sym}} = \frac{d(x, y) + d_{\text{reverse}}(x, y)}{2} = \frac{d(x, y) + d(y, x)}{2} \quad (2)$$

In the case of a min-based symmetrization, we use their minimum:

$$d_{\text{sym}} = \min\left(d(x, y), d_{\text{reverse}}(x, y)\right) = \min\left(d(x, y), d(y, x)\right) \quad (3)$$

Data set	Distance	Symmetrization		Distance learning	
		k_c (cand. k)	Recall reached	k_c (cand. k)	Recall reached
Wiki-8	Itakura-Saito	20	99	2560	99
Wiki-8	KL-div.	40	99	640	99
Wiki-8	Rényi div. $\alpha = 0.25$	20	100	640	100
Wiki-8	Rényi div. $\alpha = 2$	20	99	640	99
RCV-128	Itakura-Saito	80	99	20480	58
RCV-128	KL-div.	40	100	20480	94
RCV-128	Rényi div. $\alpha = 0.25$	80	100	5120	99
RCV-128	Rényi div. $\alpha = 2$	80	99	20480	66
Wiki-128	Itakura-Saito	20	99	20480	80
Wiki-128	KL-div.	40	99	20480	99
Wiki-128	Rényi div. $\alpha = 0.25$	160	99	5120	99
Wiki-128	Rényi div. $\alpha = 2$	80	99	20480	87
RandHist-32	Itakura-Saito	5120	96	20480	99
RandHist-32	KL-div.	160	100	2560	99
RandHist-32	Rényi div. $\alpha = 0.25$	20	100	1280	100
RandHist-32	Rényi div. $\alpha = 2$	2560	99	20480	100
Manner	BM25	1280	100	N/A	N/A

Table 3: Loss of effectiveness due to symmetrization and distance learning for 10-NN search (using at most 200K points for distance learning and at most 500K points for symmetrization)

Symmetrization techniques given by Eq. (2) and Eq. (3) are suboptimal in the sense that a *single* computation of the symmetrized distance entails *two* computations of the original distance. We can be more efficient when a distance function permits a more *natural* symmetrization, in particular, in the case of BM25 (see Eq. 1) we can compute the query term frequency using the same formula as the document term frequency. Furthermore, we can "share" a value of IDF_i between the query and the document vectors by "assigning" each vector the value $\sqrt{\mathrm{IDF}_i}$. Although the resulting function is symmetric, it is not equivalent to the original BM25. More formally, in this "shared" setting a query vector is represented by the values $\mathrm{TF}(x_i) \cdot \sqrt{\mathrm{IDF}(x_i)}$, whereas a document vector is represented by the values $\mathrm{TF}(y_i) \cdot \sqrt{\mathrm{IDF}(y_i)}$. The pseudo-BM25 similarity is computed as the inner product between query and document vectors in the following way:

$$d(x, y) = -\sum_{x_i = y_i} \left(\mathrm{TF}(x_i)\sqrt{\mathrm{IDF}(x_i)}\right) \cdot \left(\mathrm{TF}(y_i)\sqrt{\mathrm{IDF}(y_i)}\right) \quad (4)$$

Discussion of Results All the code in this section is implemented in Python. Thus, for efficiency reason, we limit the number of data points to 200K in the symmetrization experiment and to 500K in the distance learning experiment. Experimental results for $k = 10$ are presented in Table 3, where we measure how many candidates k_c we need to achieve a nearly perfect recall with respect to the original distance (we test all $k_c = k \cdot 2^i$, $i \leq 7$). We employ several symmetrization and distance learning methods: Yet, in the table, we show only the best recall for a given k_c. More specifically, we post the first k_c for which recall reaches 99%. If we cannot reach 99%, we post the maximum recall reached. We omit most low-dimensional results, because they are similar to Wiki-8 results (again, see [5] for a more detailed report).

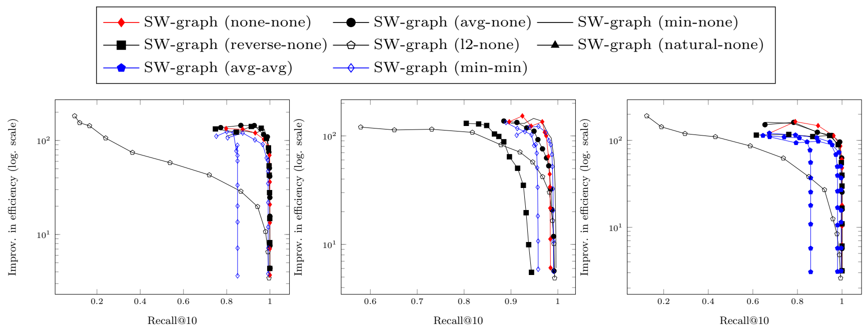
From Table 3 we can immediately see that distance learning results in a much worse approximation of the original distance than symmetrization. For high-dimensional data, it is not always possible to achieve the recall of 99% for 10-NN search. When it is possible we need to retrieve from one thousand to 20 thousand candidate entries! Even for the low-dimensional Wiki-8 data set, achieving such high recall requires at least 640 candidate entries. We conclude that using distance learning is not a promising direction, because retrieving that many candidate entries accurately is hardly possible without resorting to the brute force search with the proxy distance (which is, in turn, not efficient).

In contrast, in the case of symmetrization, the number of required candidate entries is reasonably small except for Manner and RandHist-32 data sets. We, therefore, explore various symmetrization approaches in more details in the following section. Also note that KL-divergence can be symmetrized with little loss in accuracy, i.e., on the histogram-like data KL-divergence is only mildly non-symmetric. There is prior work on non-metric k-NN search that demonstrated good results specifically for KL-divergence [24,25] for Wiki-d and RCV-d data sets. However, as our experiments clearly show, this work does not use a substantially non-symmetric distance.

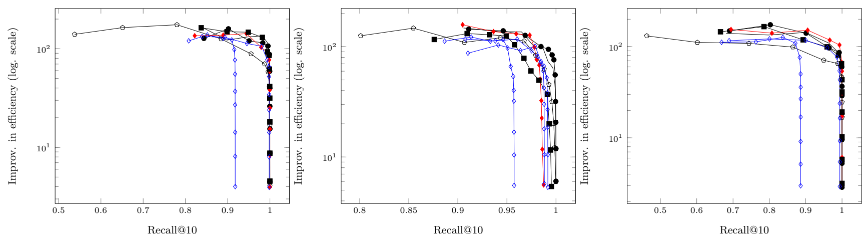
Experiments with Index- and Query-Time Symmetrization for SW-graph In this section, we evaluate the effect of the distance symmetrization in two scenarios (for 10-NN search):

- A symmetrized distance is used for *both* indexing and retrieval. We call this a full symmetrization scenario. The search procedure is carried out using an SW-graph index [22] (see § 2.1). This search generates a list of k_c candidates. Then, candidates are compared exhaustively with the query. This filter-and-refine experiment is analogous to the previous-subsection experiments except here we use approximate instead of the exact brute-force search.
- The second scenario relies on a partial, i.e., index-time only, symmetrization. Specifically, the symmetrized distance is used only to construct a proximity/neighborhood graph via SW-graph. Then, the search procedure uses the *original*, *non-symmetrized* distance to "guide" the search through the proximity graph.

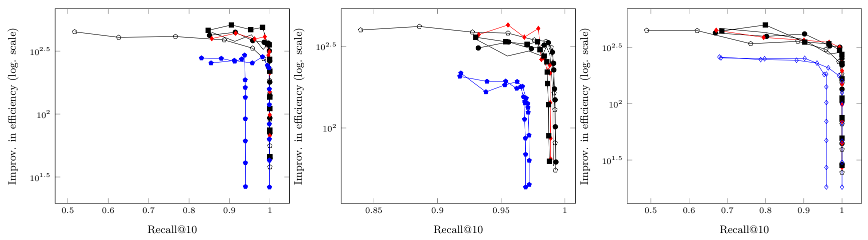
Overall, we have 31 combinations of data sets and distances, but in this paper we present the results for most interesting cases (again see [5] for a complete set of plots). We randomly split data three times into queries and indexable data set points. For all distances except Rényi divergence we use 1K queries for each split, i.e., the total number of queries is 3K. Because Rényi divergence is slow to compute, we use only 200 queries per split (i.e., the overall number of queries is 600).



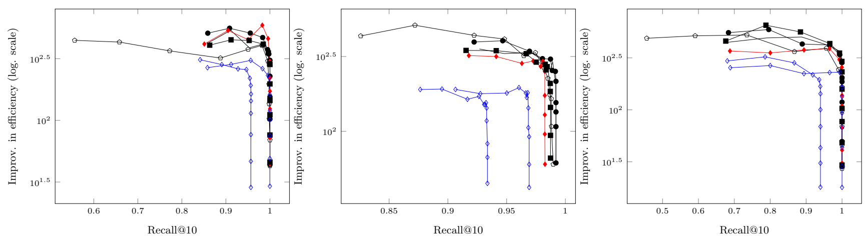
(a) RCV-8 (Itakura-Saito) (b) Wiki-8 (Itakura-Saito) (c) RandHist-8 (Itakura-Saito)



(d) RCV-8 (KL-div.) (e) Wiki-8 (KL-div.) (f) RandHist-8 (KL-div.)

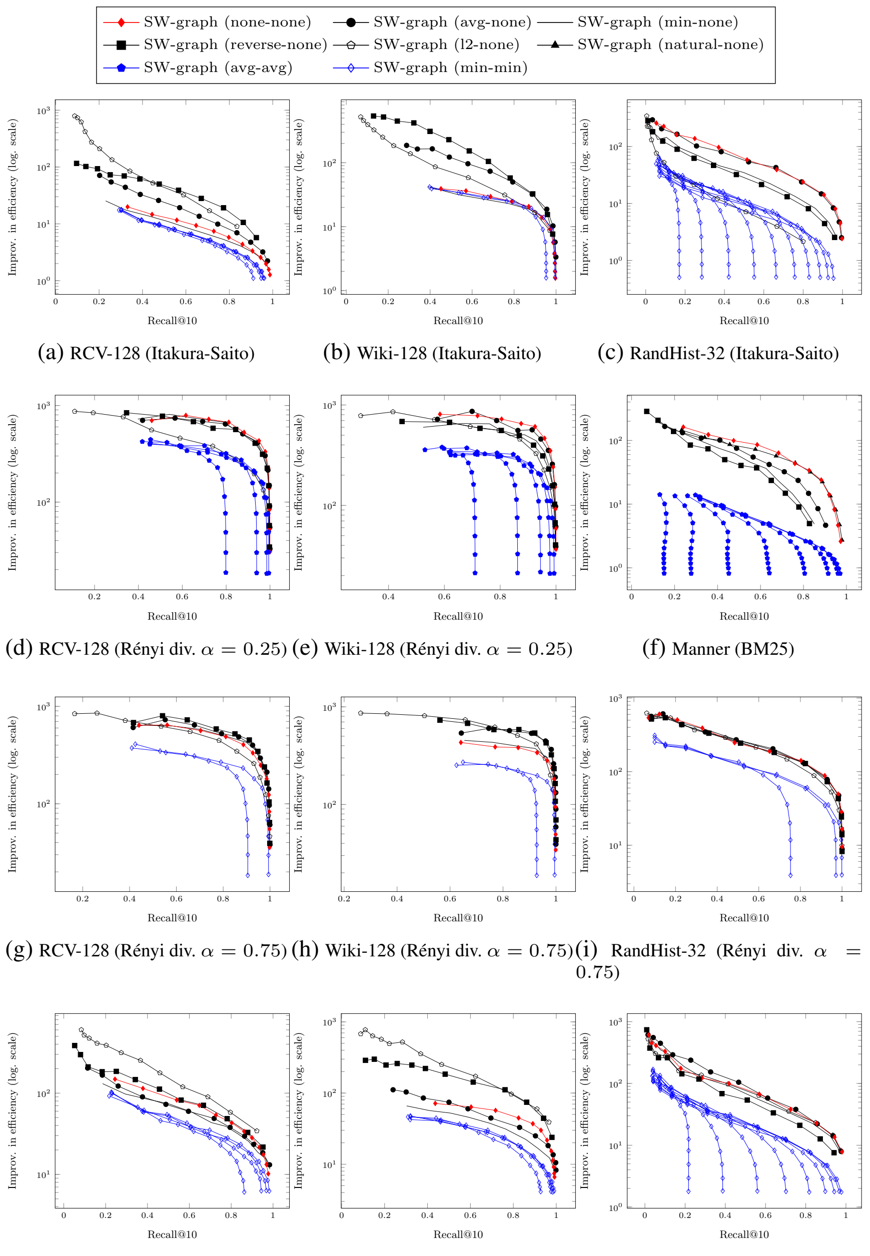


(g) RCV-8 (Rényi div. $\alpha = 0.25$) (h) Wiki-8 (Rényi div. $\alpha = 0.25$) (i) RandHist-8 (Rényi div. $\alpha = 0.25$)



(j) RCV-8 (Rényi div. $\alpha = 0.75$) (k) Wiki-8 (Rényi div. $\alpha = 0.75$) (l) RandHist-8 (Rényi div. $\alpha = 0.75$)

Fig. 1: Efficiency/effectiveness trade-offs of symmetrization in 10-NN search (part I). The number of data points is at most 500K. Best viewed in color.



(a) RCV-128 (Itakura-Saito) (b) Wiki-128 (Itakura-Saito) (c) RandHist-32 (Itakura-Saito)

(d) RCV-128 (Rényi div. $\alpha = 0.25$) (e) Wiki-128 (Rényi div. $\alpha = 0.25$) (f) Manner (BM25)

(g) RCV-128 (Rényi div. $\alpha = 0.75$) (h) Wiki-128 (Rényi div. $\alpha = 0.75$) (i) RandHist-32 (Rényi div. $\alpha = 0.75$)

(j) RCV-128 (Rényi div. $\alpha = 2$) (k) Wiki-128 (Rényi div. $\alpha = 2$) (l) RandHist-32 (Rényi div. $\alpha = 2$)

Fig. 2: Efficiency/effectiveness trade-offs of symmetrization in 10-NN search (part II). The number of data points is at most 500K. Best viewed in color.

Experiments are carried out using a `nmslib4a_bigger_reruns` branch[5] of NMSLIB [6]. We did not modify the standard NMSLIB code for SW-graph: Instead, we created a new implementation (file `small_world_rand_symm.cc`).

In the second scenario, we experiment with index- and query-time symmetrization in an actual indexing algorithm SW-graph rather than relying on the brute-force search. This approach generates a *final* list of k nearest neighbors rather than k_c candidates. *No* further re-ranking is necessary. We use two actual symmetrization strategies (the minimum- and the average-based symmetrization) as well as two types of *quasi*-symmetrization. For the first quasi-symmetrization type, we build the proximity graph using the Euclidean distance between vectors. The second quasi-symmetrization consists in building the proximity graph using the argument-reversed distance (see p. 6).

We verified that *none* of these quasi-symmetrization approaches would produce a better list of candidates in the filter-and-refine scenario (where the brute-force search is used to produce a candidate list). For example, for Wiki-128 and KL-divergence, it takes $k_c = 40$ candidates to exceed a 99% recall in a 10-NN search for the minimum-based symmetrization. For the L_2-based symmetrization, it takes as many as $k_c = 320$ candidates. The results are even worse for the filtering based on the argument-reversed distance: By using as many as $k_c = 1280$ candidates we obtain a recall of only 95.6%. It clearly does not make sense to evaluate these quasi-symmetrization methods in the complete filter-and-refine scenario. Yet, we need to check if it is beneficial to build the graph using a distance *different* from the original one.

Discussion of Results Experiments were run on a laptop (i7-4700MQ @ 2.40GHz with 16GB of memory). Results are presented in Fig. 1 (low-dimensional data) and Fig. 2 (high-dimensional data). These are efficiency-effectiveness plots: Recall@10 is shown on the x-axis, improvement in efficiency—i.e., the speed up over the brute-force search—is shown on the y-axis. Higher and to the right is better. We test several modifications of SW-graph each of which has an additional marker in the form: `a-b`, where `a` denotes a type of index-time symmetrization and `b` denotes a type of query-time symmetrization. Red plots represent the original SW-graph, which is labeled as SW-graph (none-none).

Black plots represent modifications, where symmetrization is used only during indexing: SW-graph (avg-none), SW-graph (min-none), SW-graph (l2-none), SW-graph (reverse-none), and SW-graph (natural-none). The first two types of symmetrization are average- and minimum-based. SW-graph (l2-none) is a quasi-symmetrization approach that builds the graph using L_2, but searches using the original distance. SW-graph (reverse-none) builds the graph using the reversed-argument distance, but searches using the original distance. SW-graph (natural-none) is a natural symmetrization of BM25 described by Eq. (4), which is used only for *Manner*.

Blue plots represent the case of full (both query- and index-time) symmetrization. The index is used to carry out a k_c-NN search, which produces a list of k_c candidates for further verification. Depending on which symmetrization approach was more effective in the the first series experiments (with brute-force search), we use either SW-graph (min-min) or SW-graph (avg-avg), which stand for full minimum- or average-based symmetrization. Because we do not know an optimum number of candidate records, we

[5] https://github.com/nmslib/nmslib/tree/nmslib4a_bigger_reruns

experiment with $k_c = k \cdot 2^i$ for successive integer values i. The larger is i, the more accurate is the filtering step and the less efficient is retrieval. However, it does not make sense to increase i beyond the point where the filtering accuracy reaches 99%. For this reason, the minimum value of k_c is k and the largest value of k_c is taken from Table 3.

For the remaining parameters of SW-graph we choose values that are known to perform well in other experiments: `NN`=15, `efConstruction`=100, and `efSearch` = 2^j for $0 \le j \le 12$. Analogous to the first scenario (with brute-force search), we use 31 combination of data sets and distances. In each test, we randomly split data (into queries and indexable data) three times and average results over three splits.

From Figures 1-2, we can see that in some cases there is little difference among best runs with the fully symmetrized distance (a method SW-graph (min-min) or SW-graph (avg-avg)) the runs produced by methods with true index-time symmetrization (SW-graph (min-none), SW-graph (avg-none)), and the original unmodified search algorithm (SW-graph (none-none)). Furthermore, we can see that there is often no difference between SW-graph (min-none), SW-graph (avg-none), and SW-graph (none-none). However, sometimes all fully-symmetrized runs (for all values of k_c) are noticeably less efficient (see, e.g., Panels 1h and 1k). This difference is more pronounced in the case of high-dimensional data. Here, full symmetrization leads to a substantial (up to an order of magnitude) loss in performance in most cases.

Effectiveness of index-time symmetrization varies from case to case and there is no definitive winner. First, we note that in four cases index-time symmetrization is beneficial (Panels 2a, 2b, 2j, 2k). In particular, in Panels 2a, 2b, 2k, there is an up to $10\times$ speedup. Note that it can sometimes be achieved by using an argument-reversed distance (Panels 2a, 2b) or L_2 (2k). This a *surprising* finding given that these quasi-symmetrization approaches do not perform well in the re-ranking–filter-and-refine—experiments. In particular, for L_2 and Wiki-128 reaching a 99% recall requires $k_c = 640$ compared to $k_c = 80$ for min-based symmetrization. For the Itakura-Saito distance and data sets RCV-128 and Wiki-128, it takes $k_c \le 80$ to get a 99% recall. However, using the argument-reversed distance, we do not even reach the recall of 60% despite using a large $k_c = 1280$. It is worth noting, however, that in several cases using argument-reversed distance at index time leads to substantial degradation in performance (see, e.g., Panels 1b and 2f).

To conclude the section, we emphasize that in all cases the best performance is achieved using either the unmodified SW-graph or the SW-graph with an index-time proxy distance. However, there is not a single case where performance is improved by using the fully symmetrized distance (at both indexing and querying steps). Furthermore, in three especially challenging cases: Itakura-Saito distance with RandHist-32, Rényi divergence with RandHist-32, and BM25 with Manner, SW-graph has excellent performance. In all three cases (see Panels 2c, 2l,2f), there is more than a $10\times$ speed up at 90% recall compared to the brute-force search. Note that in these three cases data is substantially non-symmetric: Depending on the case, to accurately retrieve 10 nearest neighbors with respect to the original metric, it requires to obtain 1-5K nearest neighbors using its symmetrized variant (see Table 3). Thus, in these challenging cases, a brute-force filter-and-refine symmetrization solution would be particularly ineffective or inefficient whereas SW-graph has strong performance.

4 Conclusion

We systematically evaluate effects of distance metrization, symmetrization and quasi-symmetrization on performance of brute-force and index-based k-NN search (with a graph-based retrieval method SW-graph). Unlike previous work [24,25] we experiment with substantially non-symmetric distances. Coercion of the non-metric distance to a metric space leads to a substantial performance degradation. Distance symmetrization causes a lesser performance loss. However, in all the cases a full filter-and-refine symmetrization is always inferior to either applying the graph-based retrieval method directly to a non-symmetric distance or to building an index (which is a neighborhood graph) with a modified, e.g. symmetrized, distance. Quite surprisingly, sometimes the best performing index-time distance is neither the original distance nor its symmetrization. This observation motivates a new line of research of designing index-specific graph-construction distance functions.

Acknowledgments This work was done while Leonid Boytsov was a PhD student at CMU. Authors gratefully acknowledge the support by the NSF grant #1618159.

References

1. Arya, S., Mount, D.M.: Approximate nearest neighbor queries in fixed dimensions. In: SODA. vol. 93, pp. 271–280 (1993)
2. Aumüller, M., Bernhardsson, E., Faithfull, A.: ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Information Systems (2019)
3. Bellet, A., Habrard, A., Sebban, M.: A survey on metric learning for feature vectors and structured data. CoRR **abs/1306.6709** (2013)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research **3**, 993–1022 (2003)
5. Boytsov, L.: Efficient and accurate non-metric k-NN search with applications to text matching. Ph.D. thesis, Carnegie Mellon University (2017)
6. Boytsov, L., Naidan, B.: Engineering efficient and effective non-metric space library. In: In SISAP 2013. vol. 8199, pp. 280–293. Springer (2013)
7. Cayton, L.: Fast nearest neighbor retrieval for bregman divergences. In: Proceedings of the 25th international conference on Machine learning. pp. 112–119. ACM (2008)
8. Chávez, E., Navarro, G., Baeza-Yates, R.A., Marroquín, J.L.: Searching in metric spaces. ACM Comput. Surv. **33**(3), 273–321 (2001)
9. Chechik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. Journal of Machine Learning Research **11**, 1109–1135 (2010)
10. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Proceedings of ICML 2007. pp. 209–216. ACM (2007)
11. Dong, W., Moses, C., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of WWW 2011. pp. 577–586. ACM (2011)
12. Hajebi, K., Abbasi-Yadkori, Y., Shahbazi, H., Zhang, H.: Fast approximate nearest-neighbor search with k-nearest neighbor graph. In: IJCAI/AAAI 2011. pp. 1312–1317 (2011)
13. Harwood, B., Drummond, T.: FANNG: Fast approximate nearest neighbour graphs. In: Proceedings of CVPR. pp. 5713–5722 (2016)
14. Hjaltason, G.R., Samet, H.: Properties of embedding methods for similarity searching in metric spaces. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 530–549 (2003)

15. Houle, M.E., Sakuma, J.: Fast approximate similarity search in extremely high-dimensional data sets. In: ICDE 2005. pp. 619–630 (2005)
16. Itakura, F., Saito, S.: Analysis synthesis telephony based on the maximum likelihood method. In: Proceedings of the 6th International Congress on Acoustics. pp. C17–C20 (1968)
17. Jacobs, D.W., Weinshall, D., Gdalyahu, Y.: Classification with nonmetric distances: Image retrieval and class representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(6), 583–600 (2000)
18. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Statist. **22**(1), 79–86 (03 1951)
19. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: A new benchmark collection for text categorization research. Journal of Machine Learning Research **5**, 361–397 (2004)
20. Li, W., Zhang, Y., Sun, Y., Wang, W., Zhang, W., Lin, X.: Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement (v1.0). CoRR **abs/1610.02455** (2016)
21. Liu, E.Y., Guo, Z., Zhang, X., Jojic, V., Wang, W.: Metric learning from relative comparisons by minimizing squared residual. In: IEEE ICDM 2012. pp. 978–983. IEEE (2012)
22. Malkov, Y., Ponomarenko, A., Logvinov, A., Krylov, V.: Approximate nearest neighbor algorithm based on navigable small world graphs. Inf. Syst. **45**, 61–68 (2014)
23. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. CoRR **abs/1603.09320** (2016)
24. Naidan, B., Boytsov, L., Nyberg, E.: Permutation search methods are efficient, yet faster search is possible. PVLDB **8**(12), 1618–1629 (2015)
25. Ponomarenko, A., Avrelin, N., Naidan, B., Boytsov, L.: Comparative analysis of data structures for approximate nearest neighbor search. In: DATA ANALYTICS 2014, The Third International Conference on Data Analytics. pp. 125–130 (2014)
26. Qi, G., Tang, J., Zha, Z., Chua, T., Zhang, H.: An efficient sparse metric learning in high-dimensional space via l_1-penalized log-determinant regularization. In: In: ICML 2009. pp. 841–848. ACM (2009)
27. Rényi, A.: On measures of entropy and information. In: Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability. vol. 1, pp. 547–561 (1961)
28. Robertson, S.: Understanding inverse document frequency: on theoretical arguments for IDF. Journal of Documentation **60**(5), 503–520 (2004)
29. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
30. Skopal, T., Bustos, B.: On nonmetric similarity search problems in complex domains. ACM Comput. Surv. **43**(4), 34 (2011)
31. Sutherland, D.J.: Scalable, Flexible and Active Learning on Distributions. Ph.D. thesis, Carnegie Mellon University (2016)
32. Tellez, E.S., Ruiz, G., Chávez, E., Graff, M.: Local search methods for fast near neighbor search. CoRR **abs/1705.10351** (2017), http://arxiv.org/abs/1705.10351
33. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. Pattern Recognition **12**(4), 261–268 (1980)
34. Wang, J., Shen, H.T., Song, J., Ji, J.: Hashing for similarity search: A survey. CoRR **abs/1408.2927** (2014)
35. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB. vol. 98, pp. 194–205 (1998)
36. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: NIPS 2005. pp. 1473–1480 (2005)
37. Xiong, C., Johnson, D.M., Xu, R., Corso, J.J.: Random forests for metric learning with implicit pairwise position dependence. In: KDD 2012. pp. 958–966. ACM (2012)