On Efficient Retrieval of Top Similarity Vectors

Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, Ping Li
Cognitive Computing Lab
Baidu Research USA
1195 Bordeaux Dr, Sunnyvale, CA 94089, USA
10900 NE 8th St, Bellevue, WA 98004, USA
{shulongtan,zhixinzhou,v_xuzhaozhuo,liping11}@baidu.com

Abstract

Retrieval of relevant vectors produced by representation learning critically influences the efficiency in natural language processing (NLP) tasks. In this paper we demonstrate an efficient method for searching vectors via a typical non-metric matching function: inner product. Our method, which constructs an approximate Inner Product Delaunay Graph (**IPDG**) for top-1 Maximum Inner Product Search (MIPS), transforms retrieving the most suitable latent vectors into a graph search problem with great benefits of efficiency. Experiments on data representations learned for different machine learning tasks verify the outperforming effectiveness and efficiency of the proposed IPDG.

1 Introduction

With the popularity of representation learning methods, such as Word2vec (Mikolov et al., 2013a), words are represented as real-valued embedding vectors in the semantic space. Therefore, retrieval of similar word embeddings is one of the most basic operations in natural language processing with wide applicability in synonym extraction (Yoon et al., 2017), sentence alignment (Levy et al., 2017), polysemous word learning (Sun et al., 2017) and semantic search for documents related to a query.

In this work, we address on efficient retrieval of similar word embeddings via inner product (dot product) similarity. Inner product is a general semantic matching function with applications in neural probabilistic language models (Bengio et al., 2003), machine translation (Gao et al., 2014), question answering (Lee et al., 2015), and attention mechanisms (Vaswani et al., 2017). For normalized vectors, inner product is equivalent to cosine similarity, which is a common semantic textual similarity utilized in semantic classification and search (Sahami and Heilman, 2006; Ramage et al., 2009; Agirre et al., 2012; Huang et al., 2013; Liu et al., 2015; Faruqui et al., 2016; Sultan et al., 2016; Köper and im Walde, 2018; Gong et al., 2018), Relation Extraction (RE) (Plank and Moschitti, 2013) and text coherence evaluation (Putra and Tokunaga, 2017). For un-normalized vectors, although cosine similarity is still widely applied, the final matching scores of word embeddings are usually weighted (Acree et al., 2016; Srinivas et al., 2010) by ranking-based coefficients (e.g., the side information), which transforms the problem back to search via inner product (see Eq. (2)).

Formally, retrieving the most similar word with the inner product ranking function is a *Maximum Inner Product Search* (**MIPS**) problem. MIPS is a continuously addressed topic (Bachrach et al., 2014; Shrivastava and Li, 2014; Kalantidis and Avrithis, 2014; Shrivastava and Li, 2015; Guo et al., 2016; Wu et al., 2017), and it has non-trivial differences with traditional *Approximate Nearest Neighbor Search* (ANNS) (Friedman et al., 1975, 1977; Indyk and Motwani, 1998) problems. ANNS is an optimization problem of finding the close points to the query point in a given set. Usually, the "close" means smaller in metric distances such as cosine or Euclidean distance, which have obvious geometrical implications. However, inner product is a typical non-metric measure, which distinguishes MIPS from traditional ANNS problems. Thus, methods designed for ANNS may have performance limitations in MIPS. For NLP tasks, such as retrieving relevant word embeddings by cosine and Euclidean distances, different ANNS methods have been studied (Sugawara et al., 2016). To our best knowledge, there is little literature on MIPS for retrieving word or language representations.

Currently, search on graph methods, such as *Hierarchical Navigable Small World graphs* (HNSW), is regarded as the state-of-the-art ANNS method (Malkov and Yashunin, 2018). Performance evaluation has demonstrated that HNSW is able to strongly outperform other methods ANNS benchmarks for metric distances. Meanwhile, the graph structure also has the flexibility of defining measures on edges, making HNSW feasible for MIPS. Morozov et al (Morozov and Babenko, 2018) conduct HNSW for MIPS and achieve positive results and also they introduce concepts of Delaunay Graph to explain similarity graph based methods for MIPS. Nevertheless, the link between HNSW and Delaunay Graph is still tenuous. Although global optima of MIPS will be retrieved by Delaunay Graph, there are little evidence showing that HNSW approximates proper Delauny Graph for inner product. How to provide a solid graph-based MIPS method is still an open question.

In this paper, we propose a new search on graph method, namely Inner Product Delaunay Graph (**IPDG**), for MIPS. Our key contributions can be summarized as follows:

- Design an edge selection algorithm specifically for inner product that reduces useless edges on graph and thus improves the searching efficiency.
- Propose a two rounds graph construction algorithm for effectively approximating Delaunay Graph under inner product.
- Empirically evaluate the effectiveness and efficiency. Provide a state-of-the-art MIPS method for similarity search in word embedding datasets.

The organization of this paper is as below: in the next section, we will introduce the research background. In Section 3, the approximate Inner Product Delaunay Graph (IPDG) will be introduced. For Section 4, we explore the effectiveness and efficiency of IPDG in maximum inner product word retrieval and compare it with state-of-the-art MIPS methods. Section 5 concludes the whole paper.

2 Background

In this section, we will first introduce the definition of *Maximum Inner Product Search* (MIPS) problem and review state-of-the-art methods for MIPS. Later a theoretical solution for MIPS by searching on the Delaunay Graphs will be summarized.

2.1 Problem Statement

In machine learning tasks, embedding methods such as Word2vec (Mikolov et al., 2013a,b), Glove (Pennington et al., 2014) or deep collaborative filtering (Xu et al., 2018) learn representations of data as dense distributed real-value vectors. Formally, for latent space $X \subset \mathbb{R}^d$, given an arbitrary query vector $q \in X$ and a set of vectors $S = \{x_1, \ldots, x_n\} \subset X$, vector similarity is defined as a continuous symmetric matching function, $f : X \times X \to \mathbb{R}$. The goal of similar vector retrieval is to find:

$$\arg\max_{x \in S} f(x, q). \tag{1}$$

In our paper, we specially discuss the non-metric similarity measure, inner product:

$$f(x, q) = x^\top q, \quad x, q \in X = \mathbb{R}^d \backslash \{0\}.$$

Without loss of the generality, we can always assume $\|q\| = 1$. We are not interested in the zero vector since its inner product with any vector is always zero. The problem in Eq. (1) with respect to the inner product is often referred to as *Maximum Inner Product Search* (MIPS) in literature.

The weighted cosine ANNS problem can also be viewed as the MIPS problem. We consider a data set $S = \{(z_i, w_i) : i \in [n]\}$ where w_i is an real scalar and z_i is a vector.

$$w \cos(z, q) = w \frac{z^\top q}{\|z\|\|q\|} = \frac{wz}{\|z\|}^\top \frac{q}{\|q\|}, \tag{2}$$

where $\|q\| = 1$. As can be seen, weighted ANNS w.r.t. cosine similarity is equivalent to MIPS by letting $x_i = w_i z_i / \|z_i\|$.

2.2 Related Works

Previous approaches for *Maximum Inner Product Search* (**MIPS**) can be mainly categorized into: (1) reducing MIPS to ANNS; (2) non-reduction methods. Reduction methods add wrappers on indexed data and queries asymmetrically and reduce the MIPS problem to ANNS in metric spaces (Shrivastava and Li, 2015; Bachrach et al., 2014). For example, given the query q, the indexed data $S = \{x_1, ..., x_n\}$ and $\Phi = \max_i ||x_i||$, the wrapper can be defined as:

$$P(x) = [x/\Phi; \sqrt{1 - ||x||^2/\Phi^2}], \tag{3}$$

$$Q(q) = [q; 0]. \tag{4}$$

It is not difficult to prove that searching on the new data by cosine or ℓ_2-distance is equal to search on the original data by inner product. Recently, researchers found that methods above can be improved further, based on the observation of the long tail distribution in data norms (Huang et al., 2018;

Yan et al., 2018). New approaches are proposed by adding wrappers for each norm range, such as **Range-LSH** (Yan et al., 2018). With reductions like the above one, any ANNS methods can be applied for MIPS. However, it was shown that there are performance limitations for the reduction MIPS methods (Morozov and Babenko, 2018).

Recently, more and more non-reduction methods are proposed, specifically for MIPS. Guo et al. proposed an MIPS method based on **Product Quantization (PQ)** (Guo et al., 2016). Yu et al. used an upper bound of inner product as the approximation of MIPS and designed a greedy search algorithm to find this approximation, called **Greedy-MIPS** (Yu et al., 2017). Graph-based non-reduction MIPS method, **ip-NSW**, was firstly introduced in Morozov and Babenko (2018) and the theoretical basis for conducting MIPS by similarity graph was also provided. Continuing of the advantages of similarity graph based methods for ANNS, ip-NSW showed superior performance for MIPS.

2.3 Delaunay Graph

Delaunay Graph plays an important role in the similarity search. The properties and construction of ℓ^2-Delaunay Graph have been considered in literature (Aurenhammer, 1991; Cignoni et al., 1998). Indeed, one can generalize the definition to any real binary function, including inner product.

Definition 2.1. The *Voronoi cell* R_i with respect to f and x_i is the set

$$R_i := \{q \in X : f(x_i, q) \geq f(x, q) \text{ for all } x \in S\}.$$

Moreover, $x \in S$ is an *extreme* point if it is associated with a nonempty Voronoi cell.

Definition 2.2. The *Delaunay Graph* G with respect to f and S is an undirected graph with vertices S satisfying $\{x_i, x_j\} \in G$ if and only if $R_i \cap R_j \neq \emptyset$.

An example of Voronoi cells and corresponding Delaunay Graph in inner product space is shown in Figure 1. Regions in different colors correspond to Voronoi cells for extreme points (red ones). Delaunay Graph connects extreme points. Different from metric similarities (e.g. ℓ^2-norm), the Voronoi cells of some data points with respect to inner product are possibly empty. By Definition 2.2, a data point is isolated (i.e., have no incident edges) if its Voronoi cell is empty. As we can see in Figure 1, there are many isolated points (blue ones). The

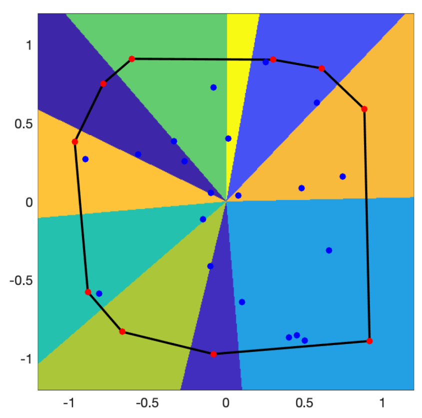


Figure 1: This shows the relation between Delaunay Graph and Voronoi cells in inner product space. The red dots are $extreme$ points of each Voronoi cell. Delaunay Graph connects extreme points with black edges. If we search on this dataset, every query has a maximum inner product with one of these extreme points (i.e., red ones).

proportion of extreme points is relatively small in general. And Theorem 2.1 will show that only extreme points can achieve a maximum inner product score for any nonzero query.

The definition of an extreme point is equivalent to the one in (Barber et al., 1996), i.e., $x \in S$ is extreme if and only if x is on the boundary of the convex hull of S. In the two dimensional cases, the edges form the boundary of the convex hull, which is also shown in Figure 1.

2.4 Search on Delaunay Graph

Searching on the Delaunay Graph is demonstrated effective for similarity search (Morozov and Babenko, 2018). In the inner product case, given any query vector $q \in X$, we start from an extreme point, then move to its neighbor that has a larger inner product with q. We repeat this step until getting an extreme point has a larger inner product with q than all its neighbors and then we return it. It can be demonstrated this returned local optimum is actually the global optimum.

Generally, for any searching measure f, if the corresponding Voronoi cells are connected, then the local optimum returned by the greedy search is also the global optimum. Formally the statement can be summarized as below. The proof can be found in Morozov and Babenko (2018).

Theorem 2.1. Suppose f satisfies that the Voronoi cells R_i with respect to any subsets of S (including S itself) are connected on X, and G is the Delaunay Graph with respect to f and some S, then for $q \in X$, a local maximum in the greedy search starting from an extreme point, that is, $x_i \in S$ satisfies

$$f(x_i, q) \geq \max_{x \in N(x_i)} f(x, q) \quad (5)$$
$$\text{where} \quad N(x_i) = \{x \in S : \{x_i, x\} \in G\}$$

is a global maximum.

Suppose the assumptions (i.e., connected Voronoi cells) in Theorem 2.1 hold, we say searching on Delaunay Graph can find the global maximum. It is easy to check that the assumptions hold for the inner product case since the Voronoi cells w.r.t. the inner product are either empty or a convex cone, so they are connected. Then we can claim that searching on Delaunay Graph in inner product, the vector in S that has the maximum inner product with the query vector will be retrieved.

3 Inner Product Delaunay Graph

Although the Delaunay Graph has demonstrated its potentials in similarity search, the direct construction of the Delaunay Graph in large scale and high dimensional datasets is unfeasible due to the exponentially growing number of edges in high dimension. To remedy this issue, practical algorithms usually approximate Delaunay Graphs. In this section, we will present the new proposed algorithm for constructing approximate Delaunay Graph in inner product space, namely Inner Product Delaunay Graph (IPDG). Two key features of our algorithm will be introduced first: i) edge selection specifically for inner product; and ii) the two rounds graph construction. And then we will conduct a case study on the toy dataset to show the effectiveness of IPDG in constructing better approximate Delaunay Graphs for inner product.

3.1 Edge Selection for Inner Product

To balance the effectiveness (retrieval of the nearest neighbor) and the efficiency (complete the process within limit time) of the retrieval, some empirical tricks are usually applied in previous search on graph methods: a) use directed edges instead of undirected edges; b) restrict the degree of outgoing edges for each node; and c) select more diverse outgoing edges (Malkov and Yashunin, 2018; Morozov and Babenko, 2018).

Specifically, for the inner product case, ip-NSW proposed in (Morozov and Babenko, 2018) applies all tricks listed above (although the authors did not mention it in the paper, the implementation did inherit all features from HNSW). We found that the edge selection method is vital for the trade-off of effectiveness and efficiency in searching. However, the existing edge selection techniques used in HNSW and ip-NSW are actually designed for metric distances, which are inapplicable for the non-metric measure, e.g., inner product.

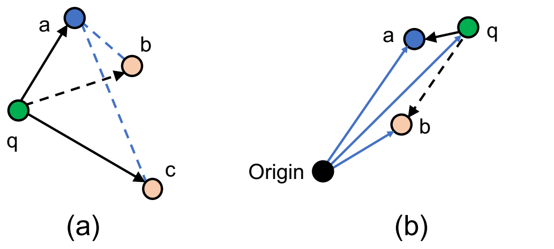


Figure 2: The example of edge selection used in constructing approximate Delaunay Graph. (a) the selection method for metric spaces used in HNSW and ip-NSW. c is selected while b is abandoned since it is not diverse from a. (b) the edge selection in IPDG. b will be ignored because a is a "super" point of it, which has been selected.

As shown in Figure 2 (a), the edge selection for metric spaces works as below: for each new inserting node (or edge updating node) q and its nearest neighbor set (candidates) from Algorithm 2, a directed edge from q to the nearest neighbor a is constructed first. For other candidates, say b, the edge selection algorithm will check whether:

$$dis(q, b) < dis(a, b), \quad (6)$$

where $dis(\cdot, \cdot)$ is the distance of two vectors, such as ℓ_2-distance or angular distance. If it is true, there will be an edge from q to b, otherwise, b will be abandoned in the selection. By this way, in a restricted degree, the new inserting node will have diverse outgoing neighbors. As shown in Figure 2 (a), b is not selected while c is selected.

It is obvious that the edge selection method for metric spaces is not suitable for inner product. As presented in Figure 2 (b), although $q^\top b > a^\top b$ (corresponding to $dis(q, b) < dis(a, b)$), b should not be selected, since $a^\top b > b^\top b$ and for any query vector q' with all positive elements, we have $q'^\top a > q'^\top b$. This means that b is dispensable in the top-1 MIPS task and the edge from q to b should not be constructed. To solve this issue, we propose a new edge selection method by checking whether:

$$b^\top b > a^\top b. \quad (7)$$

If it is true, we will select b. Otherwise, we will skip b since a is a "super" point of b and b is dispensable. In this way, each inserting node will trend to connect with extreme points but not other short norm vectors. The detailed algorithm is summarized in Algorithm 1 Lines $17-28$.

Algorithm 1 IPDG Construction

```
1: **Input:** dataset $S$, the size of candidate size $N$, maximum outgoing degree of graph $M$.
2: Initialize graph $G = \emptyset$. $round = 0$
3: **while** $round < 2$ **do**
4:     $round = round + 1$
5:     **for** each $x$ in $S$ **do**
6:         $A \leftarrow$ GREEDY_SEARCH$(x, G, N)$.
7:         $B \leftarrow$ EDGE_SELECTION$(A, M)$.
8:         Add edges $\overrightarrow{xy}$ to $G$ for every $y \in B$.
9:         **for** each $y$ in B **do**    ▷ Edge Updating
10:            $C \leftarrow \{z \in S : \overrightarrow{yz} \in G\} \cup \{x\}$.
11:            $D \leftarrow$ EDGE_SELECTION$(C, M)$.
12:            Remove original outgoing edges of $y$, and add edges $\overrightarrow{yz}$ to $G$ for $z \in D$.
13:        **end for**
14:    **end for**
15: **end while**
16: **Output:** graph $G$.
17: **function** EDGE_SELECTION$(A, M)$
18:     $B = \emptyset$.
19:     **for** $y \in A$ **do**
20:         **if** $y^\top y \geq \max_{z \in B} y^\top z$ **then**
21:             $B = B \cup \{y\}$.
22:         **end if**
23:         **if** $|B| \geq M$ **then**
24:             Break.
25:         **end if**
26:     **end for**
27:     **Output:** $B$.
28: **end function**
```

3.2 Two-Round Construction

Based on the new edge selection method introduced above (and the reverse edge updating, see Algorithm 1 Lines $9-13$), nodes with larger norms will have higher probabilities to be selected as outgoing neighbors. So extreme points of the dataset will have more incoming edges and non-extremes points will more likely have no incoming edges in general. This is consistent with the true Delaunay Graphs in inner product space as previously shown in Figure 1.

However, at beginning of the graph construction, relatively "super" points are not true extreme points. Vectors coming in later may be better candidates (i.e., true extreme points). This issue will damage the overall graph quality and affect the final searching performance. A straightforward method may probably help: inserting data points with larger norms first. We tried this trick but it did not work well. The reason is that high norm points are not necessarily extreme points. Norms of extreme points for some Voronoi cells may be relatively small. The top large norm points may be just from one or a few Voronoi cells. In high dimensional data, it is difficult to find true extreme points. Alternatively, we design a two rounds construction algorithm to solve this issue and exploit the additional round construction to update edges, especially for nodes inserted in the beginning. In this way, the graph construction algorithm can detect extreme points automatically. We tried to conduct this two rounds construction method for ip-NSW too, but there are no significant improvements.

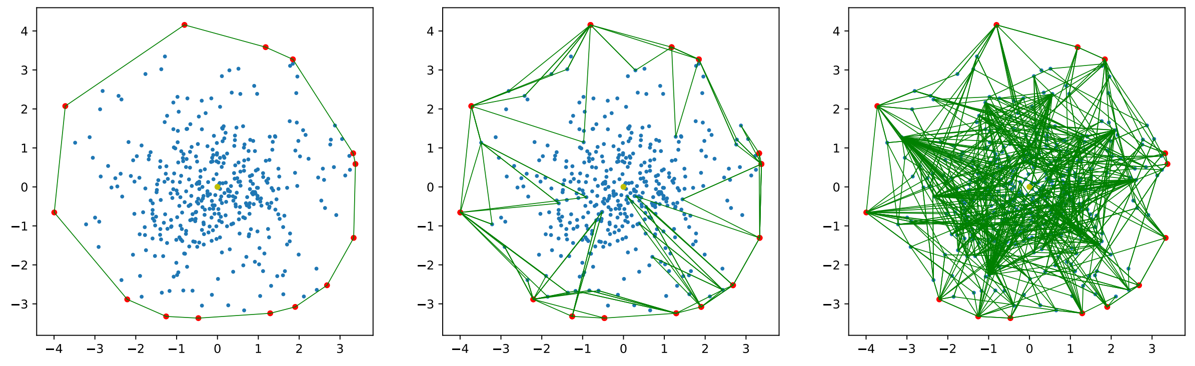
We share the graph construction algorithm for IPDG, including the edge selection function in Algorithm 1. After the graph being constructed, we perform MIPS via a greedy search algorithm presented in Algorithm 2. The greedy search algorithm is also used in the graph construction for candidates collecting.

Algorithm 2 GREEDY_SEARCH$(q, G, N))$

```
1: **Input:** The query $q$, the index graph $G$, the size of candidate set $N$.
2: Randomly choose a node with outgoing edges, say $y$. $A \leftarrow \{y\}$. Mark $y$ as checked and the rest as unchecked.    ▷
   In the practical implementation, A is a priority queue for efficiency. We note A as a set here to simplify the expression.
3: **while** not all nodes in $G$ are checked **do**
4:     $A \leftarrow A \cup \{z \in S : \overrightarrow{yz} \in G, y \in A, z$ unchecked$\}$
5:     Mark nodes in $A$ as checked.
6:     $A \leftarrow$ top $N$ candidates in $A \cup Z$ in descending order of inner product with $q$.
7:     **if** $A$ does not update **then**
8:         Break.
9:     **end if**
10: **end while**
11: **Output:** $A$.
```

3.3 A Toy Example

To further explain the differences between the proposed method and previous state-of-the-art, ip-NSW, we conduct a case study on a toy example



(a) (b) (c)

Figure 3: This is a toy example for approximate inner product Delaunay Graph construction (green lines are edges here) and red dots are $extreme$ points too. (a) is the true Delaunay Graph. (b) is an approximation by IPDG. (c) is built by ip-NSW. Note that IPDG and ip-NSW construct directed edges instead of undirected ones for the efficiency consideration. Only edges for nodes with incoming edges are shown in (b) and (c).

data which is shown in Figure 3. We randomly generate 400 two dimensional vectors following distribution Normal$(0, I_2)$. Figure 3 (a) shows the true Delaunay Graph for inner product. Red nodes correspond to extreme points of this dataset. Figure 3 (b) and (c) are graphs built by the proposed IPDG and ip-NSW, respectively. The parameter N is set to 10 and M is set to 2 for both algorithms in this study. Note that graphs built by IPDG and ip-NSW are directed graphs. To give better showing out, we only keep edges corresponding to nodes with incoming edges and other edges are ignored. Nodes without incoming edges will not be visited and do not affect the searching process, thus can be removed after the graph construction. As can be seen, the graph built by IPDG is more like the true Delaunay Graph and is more efficient for MIPS, while the graph built by ip-NSW have too many useless edges as shown in Figure 3 (c).

4 Experiments

In this section, we evaluate the proposed IPDG by comparing it with state-of-the-art MIPS methods.

4.1 Datasets

We used the following three pre-trained embeddings to investigate the performance of IPDG in MIPS for similar word searching. For each word embedding datasets, we random select 10000 vectors as queries and others as the base data.

fastTextEn and **fastTextFr** are 300 dimensional English and French word embeddings trained on Wikipedia using fastText (Joulin et al., 2016).

GloVe50 are 50 dimensional word embeddings trained on Wikipedia2014 and Gigaword5 using GloVe (Pennington et al., 2014).

As most state-of-the-art MIPS algorithms evaluate their performance on recommendation datasets, we also benchmark IPDG on three recommendation datasets: Amazon Movie (**Amovie**), **Yelp** and **Netflix**. We use the Matrix Factorization (MF) method in (Hu et al., 2008) to obtain latent vectors of user and item. Then, in the retrieval process, user vectors are regarded as queries and the item vector that has the highest inner product score with each query should be returned by the MIPS algorithm.

Datasets	Dimension	# Base Data
fastTextEn	300	989873
fastTextFr	300	1142501
GloVe	50	1183514
Amovie	64	104708
Yelp	64	25815
Netflix	50	17770

Table 1: Statistics of the datasets.

Statistics of the six datasets are listed in Table 1. They vary in dimension (300, 64 and 50), sources (recommendation ratings, word documents) and extraction methods (fastText, GloVe and MF), which is sufficient for fair comparison. The ground truth is the top-1 nearest neighbor by inner product.

4.2 Baselines

In this paper, we compare IPDG with state-of-the-art MIPS methods. Firstly, reduction methods can be baselines. Some popular ANNS open source platforms utilize the reduction trick to solve MIPS, such as Annoy[1]. As introduced in Section 2.2, with reductions, any ANNS methods can be applied for

[1]https://github.com/spotify/annoy

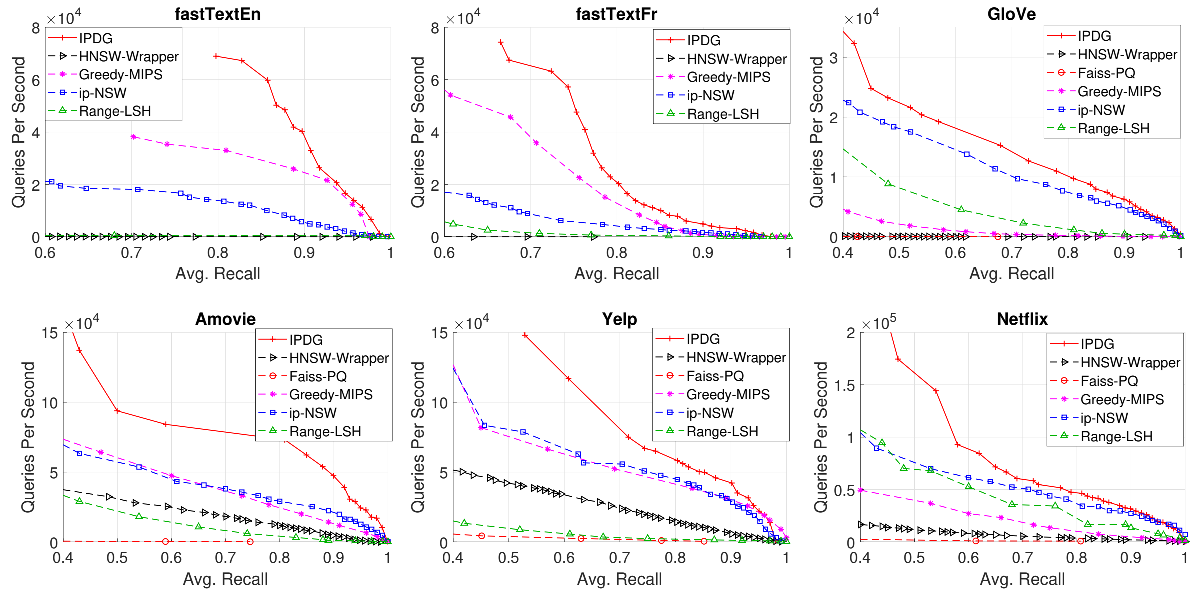


Figure 4: Recall vs. Time curves of for all methods in top-1 MIPS. Results for Faiss-PQ on fastTextEn and fastTextFr are not shown since they cannot produce recalls greater than 0.6. Best results are in upper right corners.

MIPS. In this line, we choose HNSW (Malkov and Yashunin, 2018) (referred to as **HNSW-Wrapper**) as the baseline and neglect other alternatives since HNSW is usually regarded as the most promising method for ANNS in metric spaces. We exploit the original implementation of HNSW[2] and add the wrapper introduced in Section 2.2.

Range-LSH (Yan et al., 2018) is also an reduction MIPS method and considers norm distribution of the data. The original implementation[3] is used.

Faiss-PQ[4] is a popular open source ANNS platform from Facebook, which is mainly implemented by Product Quantization (PQ) techniques. It contains MIPS as one component.

Greedy-MIPS is an MIPS algorithm from Yu et al. (2017). We use the original implementation[5].

ip-NSW is a state-of-the-art MIPS algorithm proposed in (Morozov and Babenko, 2018).[6].

4.3 Experimental Settings

There are two popular ways to evaluate ANNS/MIPS algorithms: i) **Recall vs. Time**; ii) **Recall vs. Computations**. Recall vs. Time reports the number of queries an algorithm can process per second at each recall level. Recall vs. Computations reports the amount/percentage of pairwise distance/similarity computations that the ANNS/MIPS algorithm costs at each recall level. Both evaluation indicators have their own pros and cons. Recall vs. Time is straightforward but it may introduce bias in implementation. Recall vs. Computations is beyond implementation but it does not consider the cost of different index structures. We will show both of these perspectives in the following experiments for the comprehensive evaluation.

All comparing methods have tunable parameters. In order to present a fair comparison, we vary all parameters over a fine grid for all methods. For each algorithm in each experiment, we will have multiple points scattered on the plane. To plot curves, we first find out the best result, $\max_x$, along with the x-axis (i.e., Recall). Then 100 buckets are produced by splitting the range from 0 to max_x evenly. For each bucket, the best result along the y-axis (e.g., the biggest amount of queries per second or the lowest percentage of computations) is chosen. If there are no data points in the bucket, the bucket will be ignored. In this way, we shall have multiple pairs of data for drawing curves. All time-related experiments were performed on a 2X 3.00 GHz 8-core i7-5960X CPU server with 32GB memory.

4.4 Experimental Results

We first show experimental results for all comparison algorithms from the view of Recall vs. Time, which are shown in Figure 4. Overall, the proposed method IPDG performs consistently and significantly better than baselines on all six datasets. As

[2]https://github.com/nmslib
[3]https://github.com/xinyandai/similarity-search
[4]https://github.com/facebookresearch/faiss
[5]https://github.com/rofuyu/exp-gmips-nips17
[6]https://github.com/stanis-morozov/ip-nsw

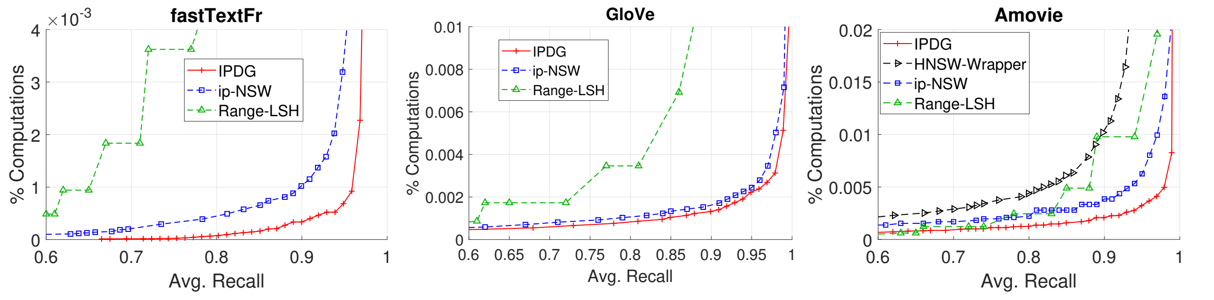


Figure 5: Recall vs. Computations curves in top-1 MIPS. Note that it is unable to show results for HNSW-Wrapper on fastTextFr and Glove in the showing scope. Best results are in lower right corners.

can be seen, some baselines show promising performance on partial datasets but they may work much worse on other datasets. For example, on lower dimensional datasets (i.e., the last four figures of Figure 4), ip-NSW work well but it fails on high dimensional datasets (i.e., fastTextEn and fastTextFr). Greedy-MIPS shows advantages on high dimensional datasets while becomes worse on some lower dimensional datasets, such as Netflix and GloVe. Among all methods, only IPDG works consistently well on all datasets which shows its effectiveness and robustness. Range-LSH performs badly in these experiments. The main reason is that Range-LSH does not have a good "budget" setting, similar to the budget in Greedy-MIPS and the N_{search} parameter in graph-based methods. HNSW-Wrapper does not work comparably with IPDG either, especially on word embedding datasets. On some recall levels, say higher than 0.5, searching by HNSW-Wrapper is extremely slow (see the first three figures). It is clear that HNSW-Wrapper is far from state-of-the-art in challenging MIPS tasks, such as larger or higher dimensional vector datasets. The PQ based method, Faiss-PQ, works badly on all datasets since quantization codes can speedup the retrieval while may largely reduce the search performance, especially for the challenging top-1 MIPS problem. Note that results for Faiss-PQ on fastTextEn and fastTextFr are not shown in Figure 4 since they cannot produce recalls greater than 0.6.

We also show experimental results by Recall vs. Computations in Figure 5. Greedy-MIPS and Faiss-PQ cannot be evaluated from this view and the other four methods are explored here. Due to the limited space, only results on partial datasets are represented. As can be seen, only IPDG and ip-NSW work consistently well on all shown datasets. HNSW-Wrapper and Range-LSH work comparably with the other two methods on the recommendation dataset, Amovie, while performs much worse on the word embedding dataset, fastTextFr and GloVe. It is even unable to show the result for HNSW-Wrapper on fastTextFr and Glove in the showing scope. For IPDG and ip-NSW, they share similar index structures, it is fair to compare their computation amount for each query. To get a similar recall, IPDG requires much less inner product computation. For example, on fastTextFr, to reach the recall at 95%, ip-NSW requires about 0.3% computations while IPDG only needs 0.07% computations. This also demonstrates the efficiency of vector inner product retrieval by IPDG.

4.5 More Comparison with ip-NSW

Datasets	ip-NSW	IPDG
fastTextEn	144339 (14.6%)	100138 (10.1%)
fastTextFr	378875 (33.2%)	250750 (21.9%)
GloVe	622080 (52.6%)	437378 (37.0%)
Amovie	32434 (31.0%)	12985 (12.4%)
Yelp	5224 (20.2%)	1871 (7.2%)
Netflix	17154 (96.5%)	14867 (83.7%)

Table 2: Number and percentage of nodes with incoming edges for graphs built by ip-NSW and IPDG.

In this section, we will conduct a study by comparing the proposed IPDG and its closely related method ip-NSW on the index graph quality. The evaluation measure is the number of nodes with incoming edges. Intuitively, only extreme points of each dataset are useful for top-1 MIPS retrieval. Non-extreme points could be ignored in graph construction (i.e., without incoming edges so will not be visited in searching). Results for $N = 100$ and $M = 16$ are shown in Table 2. As can be seen, the graphs built by IPDG have much fewer nodes with incoming edges, which is consistent with the toy example introduced above. The reason can be explained as below. The finely designed edge selection method in IPDG trends to select extreme points as outgoing neighbors for each newly inserted node or each edge updating node (see Algorithm 1 Lines $9-13$). Meanwhile, extreme points will have more opportunities to keep incoming edges in the edge updating and the second round graph construction. While non-extreme points will lose their incoming edges in these processes.

5 Conclusion and Future Work

Fast similarity search for data representations via inner product is a crucial and challenging task since it is one of the basic operations in machine learning algorithms and recommendation methods. To remedy this issue, we propose a search on graph method, namely Inner Product Delaunay Graph (IPDG), for Maximum Inner Product Search (MIPS) in embedded latent vectors. IPDG provides a better approximation to Delaunay Graphs for inner product than previous methods and is more efficient for the MIPS task. Experiments on extensive benchmarks demonstrate that IPDG outperforms previous state-of-the-art MIPS methods in retrieving latent vectors under inner product.

In this paper, we improve the top-1 MIPS performance by graph-based index. In the future, we will try to move the state-of-the-art frontier further, not only for top-1 MIPS but also for top-n, $n > 1$, MIPS results. Besides of metric measures (e.g., ℓ_2-distance and cosine similarity) and inner product, more complicated measures has been studied, for example (Tan et al., 2019). It would be interesting to adopt these measures in NLP tasks. Another promising direction is to adopt a GPU-based system for fast ANNS or MIPS, which has been shown highly effective for generic ANNS tasks (Li et al., 2012; Johnson et al., 2017; Zhao et al., 2019). Developing GPU-based algorithms for MIPS is still a topic which has not been fully explored.

Acknowledgement

The authors would like to sincerely thank the anonymous reviewers of NAACL 2019 and EMNLP 2019, for their helpful comments, which has improved the quality of this paper.

References

Brice Acree, Eric Hansen, Joshua Jansa, and Kelsey Shoub. 2016. Comparing and evaluating cosine similarity scores, weighted cosine similarity scores and substring matching. Technical report.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Franz Aurenhammer. 1991. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405.

Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems (RecSys)*, pages 257–264.

C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

Paolo Cignoni, Claudio Montani, and Roberto Scopigno. 1998. Dewall: A fast divide and conquer delaunay triangulation algorithm in ed. *Computer-Aided Design*, 30(5):333–341.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Jerome H. Friedman, F. Baskett, and L. Shustek. 1975. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24:1000–1006.

Jerome H. Friedman, J. Bentley, and R. Finkel. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 699–709.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and Jinjun Xiong. 2018. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 2341–2351.

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. 2016. Quantization based fast inner product search. In *Artificial Intelligence and Statistics (AISTATS)*, pages 482–490.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM)*, pages 263–272.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information Knowledge Management (CIKM)*, pages 2333–2338.

Qiang Huang, Guihong Ma, Jianlin Feng, Qiong Fang, and Anthony KH Tung. 2018. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1561–1570.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 604–613, Dallas, TX.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Yannis Kalantidis and Yannis Avrithis. 2014. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2321–2328.

Maximilian Köper and Sabine Schulte im Walde. 2018. Analogies in complex verb meaning shifts: the effect of affect in semantic similarity models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 2, pages 150–156.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2015. Reasoning in vector space: An exploratory study of question answering. *arXiv preprint arXiv:1511.06426*.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 1, pages 765–774.

Ping Li, Anshumali Shrivastava, and Christian A. Konig. 2012. Gpu-based minwise hashing: Gpu-based minwise hashing. In *Proceedings of the 21st World Wide Web Conference (WWW)*, pages 565–566, Lyon, France.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 912–921.

Yury A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Stanislav Morozov and Artem Babenko. 2018. Non-metric similarity graphs for maximum inner product search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4722–4731.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1498–1507.

Jan Wira Gotama Putra and Takenobu Tokunaga. 2017. Evaluating text coherence based on semantic similarity graph. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*, pages 76–85.

Daniel Ramage, Anna N Rafferty, and Christopher D Manning. 2009. Random walks for text semantic similarity. In *Proceedings of the 2009 workshop on graph-based methods for natural language processing*, pages 23–31.

Mehran Sahami and Timothy D Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th International Conference on World Wide Web (WWW)*, pages 377–386, Edinburgh, Scotland, UK.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems (NIPS)*, pages 2321–2329, Montreal, Canada.

Anshumali Shrivastava and Ping Li. 2015. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 812–821, Amsterdam, The Netherlands.

Gokavarapu Srinivas, Niket Tandon, and Vasudeva Varma. 2010. A weighted tag similarity measure based on a collaborative weight model. In *Proceedings of the 2nd International Workshop on Search and Mining User-Generated Contents*, pages 79–86.

Kohei Sugawara, Hayato Kobayashi, and Masajiro Iwasaki. 2016. On approximately searching for similar word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 2265–2275.

Md Arafat Sultan, Jordan Boyd-Graber, and Tamara Sumner. 2016. Bayesian supervised domain adaptation for short text similarity. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 927–936.

Yifan Sun, Nikhil Rao, and Weicong Ding. 2017. A simple approach to learn polysemous word embeddings. *arXiv preprint arXiv:1707.01793*.

Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. 2019. Fast Item Ranking under Neural Network based Measures. Technical report, Baidu Research.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Xiang Wu, Ruiqi Guo, Ananda Theertha Suresh, Sanjiv Kumar, Daniel N Holtmann-Rice, David Simcha, and Felix Yu. 2017. Multiscale quantization for fast similarity search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5745–5755.

Jun Xu, Xiangnan He, and Hang Li. 2018. Deep learning for matching in search and recommendation. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1365–1368.

Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. 2018. Norm-ranging lsh for maximum inner product search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2952–2961, Montreal, Canada.

Seunghyun Yoon, Pablo Estrada, and Kyomin Jung. 2017. Synonym discovery with etymology-based word embeddings. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6.

Hsiang-Fu Yu, Cho-Jui Hsieh, Qi Lei, and Inderjit S Dhillon. 2017. A greedy approach for budgeted maximum inner product search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5453–5462, Long Beach, CA.

Weijie Zhao, Shulong Tan, and Ping Li. 2019. SONG: Approximate Nearest Neighbor Search on GPU. Technical report, Baidu Research.