Nearest-Neighbor Search in NLP Applications using the Non-Metric Space Library (NMSLIB)

Leo (Leonid) Boytsov

https://github.com/searchivarius/NonMetricSpaceLib
Nearest-Neighbor Search in NLP Applications using the Non-Metric Space Library (NMSLIB)

Leo (Leonid) Boytsov

https://github.com/searchivarius/NonMetricSpaceLib
Acknowledgements and Some History

- Supported by CMU’s OAQA project\(^1\) and the European iAD\(^2\) center.

- Code was written mostly by Bileg(saikhan) Naidan (NTNU) and Leo(nid) Boytsov (CMU)

- Catalyzed by an 11-701 course project

- Includes contributions from several people: Lawrence Cayton, Wei Dong, Avrelin Nikita, Alexander Ponomarenko, Yury Malkov, Daniel Lemire

\(^1\)https://github.com/oaqa
\(^2\)http://www.iad-center.com/
Talk Outline

• Definition of the nearest-neighbor (NN) search
  • Importance of non-metric search
Talk Outline

- Definition of the nearest-neighbor (NN) search
  - Importance of non-metric search
- Survey of applications in ML/NLP
  - Is NN search useful?
Talk Outline

- Definition of the nearest-neighbor (NN) search
  - Importance of non-metric search
- Survey of applications in ML/NLP
  - Is NN search useful?
- Targeted mini-survey of the state of the art
  - How can ML improve NN search?
  - Some state-of-the-art comparisons
Talk Outline

- Definition of the nearest-neighbor (NN) search
  - Importance of non-metric search

- Survey of applications in ML/NLP
  - Is NN search useful?

- Targeted mini-survey of the state of the art
  - How can ML improve NN search?
  - Some state-of-the-art comparisons

- Overview of the Non-Metric Space Library (NMSLIB)
  - Technical details
  - More state-of-the-art comparisons
  - Future work
Problem Statement

Importance of Non-Metric Access Methods
Nearest-neighbor search ($k$-NN search)

- **Input:** A set of $n$ data points (objects) and a distance function $d(x, y)$
- **Query:** New object $q$ and $k$
- **Task:** Quickly find $k$ most similar objects in the data set to $q$
Nearest-neighbor search ($k$-NN search)

- **Input:** A set of $n$ data points (objects) and a distance function $d(x, y)$
- **Query:** New object $q$ and $k$
- **Task:** Quickly find $k$ most similar objects in the data set to $q$
## Distance functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Distance function</th>
<th>Symmetry</th>
<th>Triangle ineq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean ((L_2))</td>
<td>[d(x, y) = \sqrt{\sum(x_i - y_i)^2}]</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Cosine distance</td>
<td>[1 - \frac{x \cdot y}{</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>KL-diverg.</td>
<td>[\sum x_i \log \frac{x_i}{y_i}]</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>JS-diverg.</td>
<td>symmetrized &amp; smoothed KL-diverg.</td>
<td>✔️</td>
<td>✗</td>
</tr>
</tbody>
</table>

Distance functions can be **metric** or **non-metric**.
Why Non-Metric Distances?

Source: Jacobs et al. (2000)
Why Non-Metric Distances?

Source: Jacobs et al. (2000)
More Non-Metric Examples

- Kullback-Leibler divergence:

\[
KL\text{-}\text{div}(p, q) = \sum p_i \log \frac{p_i}{q_i}
\]

- Many statistically learned similarity functions

- More examples: Jacobs et al. (2000)
Applications

Is Nearest-Neighbor Searching Useful?
Some Applications in ML and NLP

- Answering analogy questions — Mikolov et al. (2013b)
- Classification — Wan and Peng (2005); Kusner et al. (2015)
- Entity detection — Liu et al. (2011); Wang et al. (2009)
- Collaborative filtering — slideshare.net/erikbern/music-recommendations-mlconf-2014
- First story detection — Petrović et al. (2010)
- Data Imputation — Troyanskaya et al. (2001)
Analogy Questions

*man* is to *king* as *woman* is to ?

- Substantial prior work: Turney (2012)
- **Human-level** performance achieved *10 years* ago (or earlier): Turney (2004)
Analogy Questions

**man** is to **king** as **woman** is to **queen**

- Substantial prior work: Turney (2012)
- **Human-level** performance achieved **10 years** ago (or earlier): Turney (2004)
Analogy Questions: word2vec

man \ − \ king \ \approx \ woman \ − \ queen

king \ − \ man \ \approx \ queen \ − \ woman

queen = \arg\max_w \ \text{similarity}(w, \text{king} − \text{man} + \text{woman})

— Cosine similarity is the best similarity function — Mikolov et al. (2013b)
Analogy Questions: word2vec

man – king ≈ woman – queen

king – man ≈ queen – woman

Is everything Ok here?

queen = \text{argmax}_w \text{ similarity}(w, \text{king} – \text{man} + \text{woman})

— Cosine similarity is the best similarity function — Mikolov et al. (2013b)
Analogy Questions: You Need a Hack!

- man to king as woman to ?
- acorn to woods as apple to ?
- pleasure to smile as pain to ?
- France to Paris as Japan to ?

<table>
<thead>
<tr>
<th>queen</th>
<th>orchard</th>
<th>grimace</th>
<th>Tokyo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analogy Questions: You Need a Hack!

- man to king as woman to ?
- acorn to woods as apple to ?
- pleasure to smile as pain to ?
- France to Paris as Japan to ?

<table>
<thead>
<tr>
<th>queen</th>
<th>orchard</th>
<th>grimace</th>
<th>Tokyo</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.80</td>
<td>king</td>
<td>0.66</td>
<td>woods</td>
</tr>
<tr>
<td>0.63</td>
<td>smile</td>
<td>0.81</td>
<td>Tokyo</td>
</tr>
</tbody>
</table>
Analogy Questions: You Need a Hack!

- **man** to **king** as **woman** to ?
- **acorn** to **woods** as **apple** to ?
- **pleasure** to **smile** as **pain** to ?
- **France** to **Paris** as **Japan** to ?

<table>
<thead>
<tr>
<th></th>
<th>queen</th>
<th>orchard</th>
<th>grimace</th>
<th>Tokyo</th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td>king</td>
<td>0.66</td>
<td>smile</td>
<td>0.81</td>
</tr>
<tr>
<td>woman</td>
<td>king</td>
<td>0.66</td>
<td>smile</td>
<td>0.81</td>
</tr>
<tr>
<td>acorn</td>
<td>queen</td>
<td>0.51</td>
<td>pain</td>
<td>0.68</td>
</tr>
<tr>
<td>woods</td>
<td>queen</td>
<td>0.51</td>
<td>pain</td>
<td>0.68</td>
</tr>
<tr>
<td>smile</td>
<td>king</td>
<td>0.66</td>
<td>smile</td>
<td>0.81</td>
</tr>
<tr>
<td>pain</td>
<td>queen</td>
<td>0.61</td>
<td>pain</td>
<td>0.68</td>
</tr>
<tr>
<td>Japan</td>
<td>king</td>
<td>0.66</td>
<td>smile</td>
<td>0.81</td>
</tr>
<tr>
<td>Paris</td>
<td>queen</td>
<td>0.51</td>
<td>pain</td>
<td>0.68</td>
</tr>
</tbody>
</table>

*HEHACKWASONLYBRIEFLYMentionedinanotherpaperbyIKOLOVETAL.(2013A)*
Analogy Questions: You Need a Hack!

- **man to king** as **woman** to ?
- **acorn** to **woods** as **apple** to ?
- **pleasure** to **smile** as **pain** to ?
- **France** to **Paris** as **Japan** to ?

<table>
<thead>
<tr>
<th></th>
<th>queen</th>
<th>orchard</th>
<th>grimace</th>
<th>Tokyo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0.80</strong></td>
<td>king</td>
<td>0.66</td>
<td>smile</td>
<td>0.81</td>
</tr>
<tr>
<td><strong>0.71</strong></td>
<td>queen</td>
<td>0.51</td>
<td>pain</td>
<td>0.68</td>
</tr>
<tr>
<td><strong>0.62</strong></td>
<td>monarch</td>
<td>0.45</td>
<td>grin</td>
<td>0.66</td>
</tr>
</tbody>
</table>

*The table entries represent the similarity scores between the given words and their correlates.*
Analogy Questions: You Need a Hack!

- man to king as woman to ?
- acorn to woods as apple to ?
- pleasure to smile as pain to ?
- France to Paris as Japan to ?

<table>
<thead>
<tr>
<th>queen</th>
<th>orchard</th>
<th>grimace</th>
<th>Tokyo</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.80</td>
<td>king</td>
<td>0.66</td>
<td>smile</td>
</tr>
<tr>
<td>0.71</td>
<td>queen</td>
<td>0.51</td>
<td>pain</td>
</tr>
<tr>
<td>0.62</td>
<td>monarch</td>
<td>0.45</td>
<td>grin</td>
</tr>
<tr>
<td>0.59</td>
<td>princess</td>
<td>0.44</td>
<td>grimace</td>
</tr>
</tbody>
</table>

— The hack was only briefly mentioned in another paper by Mikolov et al. (2013a)
\textit{k-NN Classification}

An example of 3-NN binary classification:
**$k$-NN Classification**

An example of 3-NN binary classification:
An example of 3-NN binary classification:
"Probably the main insight was that KNN is capable of making very good meta-features. Never underestimate nearest neighbours algorithm."

— Alexander Guschin, 2d place, Kaggle Otto Product Classification
One possible application to document classification:

- Compute pairwise similarities between document words using the “semantic” distance based either on WordNet: Wan and Peng (2005), or on word embeddings: Kusner et al. (2015)

- Aggregate pairwise similarities using either the Word Mover’s Distance: Kusner et al. (2015), or the signature quadratic form distance: Beecks et al. (2010)
$k$-NN Classification in NLP

\[\text{average error w.r.t. BOW}\]

\begin{table}[h]
\begin{tabular}{c|c|c|c|c|c|c}
method & Okapi BM25 & TF-IDF & BOW & CCG & mSDA & LDA & LSI & WMD \\
\hline
average error & 1.29 & 1.15 & 1.00 & 0.72 & 0.60 & 0.55 & 0.49 & 0.42 \\
\end{tabular}
\end{table}

\textit{Figure 4.} The $k$NN test errors of various document metrics averaged over all eight datasets, relative to $k$NN with BOW.

Source: Kusner et al. (2015)
Part III

State of the Art
Can ML Improve It?
How to find similar objects?

Two main options available:

- Brute-force (always **exact**)
- Indexing (can be exact or approximate)
How to find similar objects?

Are exact indexing methods sufficiently efficient in practice?
CURSE OF DIMENSIONALITY

Source: http://liledekahlan.eklablog.com/tulips-paysages-mystiques-c176638
Curse of Dimensionality: Summary

- Exact indexing can be fast in low dimensions, but is mostly slow in a high-dimensional space.
- **Approximate** search can be fast.
- Approximate search may be the only efficient option in a non-metric space.
State-of-the-art approximate search methods (not exhaustive)

- Locality Sensitive Hashing (LSH)
- Proximity graphs (kNN-graphs)
- Permutation methods
- Hierarchical space partitioning (trees)
- Inverted files (usually used as an auxiliary data structure)
Locality Sensitive Hashing (LSH)
Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ “sensitive” to locality
  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)
Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ “sensitive” to locality
  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)

- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$
Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ “sensitive” to locality
  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)
- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$
- For reliable retrieval, use many hash functions
Locality Sensitive Hashing (LSH)

• Design a hash function $h(x)$ “sensitive” to locality
  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)

• The smaller is $d(x, y)$ the more likely $h(x) = h(y)$

• For reliable retrieval, use many hash functions

• Works well for some $L_p$ spaces & cosine similarity
Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ “sensitive” to locality
  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)
- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$
- For reliable retrieval, use many hash functions
- Works well for some $L_p$ spaces & cosine similarity
- Also, perhaps, for kernelized similarity functions
  — Kulis and Grauman (2009)
Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ “sensitive” to locality
  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)
- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$
- For reliable retrieval, use many hash functions
- Works well for some $L_p$ spaces & cosine similarity
- Also, perhaps, for kernelized similarity functions
  — Kulis and Grauman (2009)
- Much less is known about performance in a more general case
  — However, Athitsos et al. (2008)
Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)
Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

Each point is connected with two nearest neighbors:
Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

Each point is connected with **two** nearest neighbors:

![Graph diagram](attachment:graph.png)
Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

Each point is connected with two nearest neighbors:
Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

Each point is connected with two nearest neighbors:
Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

Each point is connected with **two** nearest neighbors:
Permutation Methods

Filter-and-refine using **pivot-based projection** to the Euclidean space ($L_2$):

- Select pivots randomly
- Rank pivots by their distances to data points
- Filter by comparing **pivot rankings**
- Refine by comparing remaining points to the query
Hierarchical space partitioning (VP-tree aka Ball-tree)

- A binary space-partitioning tree
  — Proposed independently by Uhlmann (1991) and Yianilos (1993)
- A **metric-space** generalization of KD-tree
- Uses the **triangle inequality** to prune unpromising partitions
VP-tree: One Tree Node

Creating one index tree node:

- A (random) pivot $\pi$ is selected
- The space is divided by a sphere into two halves
- The radius of the sphere is a median distance to $\pi$. 
VP-tree: Three Types of Query Balls

The triangle inequality makes pruning possible:

- Red query ball: prune the **outer** partition
- Blue query ball: prune the **inner** partition
- Gray query ball: **cannot** prune, visit both
VP-tree: Pruning Rule
VP-tree: Pruning Rule Learned By Sampling

The pruning function obtained by **sampling**. The red dashed line denotes a median distance $R$ from data set points to the pivot $\pi$. 

Colors, $L_2$  
RCV-8, KL-div  
RCV-16, KL-div
VP-tree: Pruning Rule Learned By Sampling

The pruning function obtained by sampling. The red dashed line denotes a median distance \( R \) from data set points to the pivot \( \pi \).

What if we learn a parametric piecewise-linear function instead?
VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall
VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

Efficiency vs. recall (10-NN search): **higher and to the right** is better (VLDB’15 results):
VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

Efficiency vs. recall (10-NN search): **higher and to the right** is better (VLDB’15 results):

Can often match or outperform the multiprobe LSH (MPLSH)
— Boytsov and Naidan (2013), Naidan et al. (2015)
VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

Efficiency vs. recall (10-NN search): **higher and to the right** is better (VLDB’15 results):
VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

Efficiency vs. recall (10-NN search): VP-tree is better than the proximity graph (VLDB’15 results):
More Examples of ML for Nearest-Neighbor Search

• Data-optimality: tune parameters to your data set
  — Dong et al. (2008); Cayton and Dasgupta (2007)

• Learn a distance function
  — Xing et al. (2002); Prekopcsák and Lemire (2012)

• Learn a monotonic transformation of the distance function
  — Skopal and Bartoš (2012)
Part IV

Non-Metric Space Library (NMSLIB)

- More state-of-the-art comparisons
- Using NMSLIB in other applications
- Next Steps
NMSLIB Overview: What is Non-Metric Space Library?
NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
- NMSLIB is an evaluation toolkit that simplifies experimentation and processing of results
NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
- NMSLIB is an evaluation toolkit that simplifies experimentation and processing of results
- NMSLIB is extensible (new spaces and methods can be added)
NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
- NMSLIB is an evaluation toolkit that simplifies experimentation and processing of results
- NMSLIB is extensible (new spaces and methods can be added)
- It was designed to be efficient: we provide efficient implementations of major distance functions
NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
- NMSLIB is an evaluation toolkit that simplifies experimentation and processing of results
- NMSLIB is extensible (new spaces and methods can be added)
- It was designed to be efficient: we provide efficient implementations of major distance functions
- NMSLIB was designed as an experimental framework, but we work towards making it useful for a broader user base
NMSLIB Overview: Design Considerations
NMSLIB Overview: Design Considerations

- **Efficiency**
  - Implemented in C++
  - Vectorized (SIMD) distances (major)
  - Memory optimized layouts for trees
NMSLIB Overview: Design Considerations

- **Efficiency**
  - Implemented in C++
  - Vectorized (SIMD) distances (major)
  - Memory optimized layouts for trees

- **Reasonable portability & interoperability**
  - Use C++11, the code works on Linux & Win64
  - We have an experimental version works as a service (client can be Java, C++, Python, ...)
  - We have experimental Python bindings
NMSLIB Overview: Design Considerations

- **Efficiency**
  - Implemented in C++
  - Vectorized (SIMD) distances (major)
  - Memory optimized layouts for trees

- **Reasonable portability & interoperability**
  - Use C++11, the code works on Linux & Win64
  - We have an experimental version works as a service (client can be Java, C++, Python, ...)
  - We have experimental Python bindings

- **Some documentation**
  - Quick start guide
  - Detailed 60-page manual
NMSLIB Overview: Core Methods

- NMSLIB includes four core methods:
  - VP-tree
  - SW-graph (proximity graph)
  - NAPP (Neighborhood APProximation index)
  - Brute-force filtering of permutations
NMSLIB Overview: Core Methods

- NMSLIB includes four core methods:
  - VP-tree
  - SW-graph (proximity graph)
  - NAPP (Neighborhood APProximation index)
  - Brute-force filtering of permutations

- In our evaluations:
  - There was no single best core method
  - Some of the core methods outperformed other approaches
  - All core methods were reasonably effective for the non-symmetric and non-metric distance

— Boytsov and Naidan (2013); Ponomarenko et al. (2014); Naidan et al. (2015)
More State-of-the-Art Comparisons: Public Benchmarks

Evaluation by Erik Bernhardsson

https://github.com/erikbern/ann-benchmarks

One million of SIFT vectors, $L_2$: 
More State-of-the-Art Comparisons: Public Benchmarks

Evaluation by Erik Bernhardsson

https://github.com/erikbern/ann-benchmarks

1.19 million of Glove 100d embeddings, cosine:
Next Steps
Next Steps

- **Practical**
  
  - Index serialization for core methods is still work in progress
  - We have a version that works as a service, but it is not propagated to the main branch yet
  - No automatic parameter tuning for proximity graphs and permutation methods
Next Steps

• **Practical**
  - Index serialization for core methods is still work in progress
  - We have a version that works as a service, but it is not propagated to the main branch yet
  - No automatic parameter tuning for proximity graphs and permutation methods

• **Experimental/Scientific**
  - Implement and test a variety of proximity graphs
  - Compare proximity graphs against recent LSH indices (which we did not adopt yet)
  - Experiment with more challenging spaces
Talk Recap

- Nearest Neighbor search can be useful in ML and NLP
- Non-metric spaces are important
- Our NMSLIB library has decent support for such spaces
- NMSLIB includes SW-graph, which is quite efficient
- That said, NMSLIB is still work in progress
- LSH may not always be the best search method
Thank you for attention!

Our code is on GitHub:

https://github.com/searchivarius/NonMetricSpaceLib
Bibliography I


Bibliography II


Bibliography III


