

Near-Neighbor Search at Scale

Casey Stella

March 6, 2012

Table of Contents

Introduction

Near Neighbor Search

Schemaless Data Stores

Locality Sensitive Hashing

Conclusion

Introduction

- ▶ Hi, I'm Casey

Introduction

- ▶ Hi, I'm Casey
- ▶ I work at Explorys

Introduction

- ▶ Hi, I'm Casey
- ▶ I work at Explorys
- ▶ I am a senior engineer on the high speed indexes team

Introduction

- ▶ Hi, I'm Casey
- ▶ I work at Explorys
- ▶ I am a senior engineer on the high speed indexes team
- ▶ I am also a recovering Mathematician

Introduction

- ▶ Hi, I'm Casey
- ▶ I work at Explorys
- ▶ I am a senior engineer on the high speed indexes team
- ▶ I am also a recovering Mathematician
- ▶ I use Hadoop and HBase (a NoSQL datastore) every day

Introduction

- ▶ Hi, I'm Casey
- ▶ I work at Explorys
- ▶ I am a senior engineer on the high speed indexes team
- ▶ I am also a recovering Mathematician
- ▶ I use Hadoop and HBase (a NoSQL datastore) every day
- ▶ I'm here to talk about Near-Neighbor Searches

Introduction

- ▶ Hi, I'm Casey
- ▶ I work at Explorys
- ▶ I am a senior engineer on the high speed indexes team
- ▶ I am also a recovering Mathematician
- ▶ I use Hadoop and HBase (a NoSQL datastore) every day
- ▶ I'm here to talk about Near-Neighbor Searches
- ▶ In particular, about how they're hard to do efficiently at scale.

Near Neighbor Search

- ▶ Given a point in a vector space, find the nearest points according to some metric

Near Neighbor Search

- ▶ Given a point in a vector space, find the nearest points according to some metric
 - ▶ You probably know a few vector spaces, like \mathbb{R}^2 from Calculus

Near Neighbor Search

- ▶ Given a point in a vector space, find the nearest points according to some metric
 - ▶ You probably know a few vector spaces, like \mathbb{R}^2 from Calculus
 - ▶ You probably know a metric or two like L_2 aka the Euclidean metric

Near Neighbor Search

- ▶ Given a point in a vector space, find the nearest points according to some metric
 - ▶ You probably know a few vector spaces, like \mathbb{R}^2 from Calculus
 - ▶ You probably know a metric or two like L_2 aka the Euclidean metric
 - ▶ Or L_1 a.k.a. Taxicab distance from A.I.

Near Neighbor Search

- ▶ Given a point in a vector space, find the nearest points according to some metric
 - ▶ You probably know a few vector spaces, like \mathbb{R}^2 from Calculus
 - ▶ You probably know a metric or two like L_2 aka the Euclidean metric
 - ▶ Or L_1 a.k.a. Taxicab distance from A.I.
- ▶ Many problems can be rephrased as a near neighbor search (or use it as a primary component)

Near Neighbor Search

- ▶ Given a point in a vector space, find the nearest points according to some metric
 - ▶ You probably know a few vector spaces, like \mathbb{R}^2 from Calculus
 - ▶ You probably know a metric or two like L_2 aka the Euclidean metric
 - ▶ Or L_1 a.k.a. Taxicab distance from A.I.
- ▶ Many problems can be rephrased as a near neighbor search (or use it as a primary component)
 - ▶ Recommendation Systems
 - ▶ Contextual Marketing (i.e. ads)
 - ▶ Clustering data
 - ▶ Lots more

Traditional Approach

- ▶ A naïve approach would be $O(n)$

¹Jacob E. Goodman, Joseph O'Rourke and Piotr Indyk (Ed.) (2004).
“Chapter 39 : Nearest neighbours in high-dimensional spaces”. Handbook of
Discrete and Computational Geometry (2nd ed.). CRC Press

Traditional Approach

- ▶ A naïve approach would be $O(n)$
- ▶ A less naïve approach typically involves *kd*-trees

¹Jacob E. Goodman, Joseph O'Rourke and Piotr Indyk (Ed.) (2004).
“Chapter 39 : Nearest neighbours in high-dimensional spaces”. Handbook of
Discrete and Computational Geometry (2nd ed.). CRC Press

Traditional Approach

- ▶ A naïve approach would be $O(n)$
- ▶ A less naïve approach typically involves *kd*-trees
- ▶ These tend to scale poorly in very high dimensions

¹Jacob E. Goodman, Joseph O'Rourke and Piotr Indyk (Ed.) (2004).
“Chapter 39 : Nearest neighbours in high-dimensional spaces”. Handbook of
Discrete and Computational Geometry (2nd ed.). CRC Press

Traditional Approach

- ▶ A naïve approach would be $O(n)$
- ▶ A less naïve approach typically involves *kd*-trees
- ▶ These tend to scale poorly in very high dimensions
 - ▶ The rule of thumb¹ is, for dimension k and number of points N , $N \gg 2^k$

¹Jacob E. Goodman, Joseph O'Rourke and Piotr Indyk (Ed.) (2004).
“Chapter 39 : Nearest neighbours in high-dimensional spaces”. Handbook of
Discrete and Computational Geometry (2nd ed.). CRC Press

Traditional Approach

- ▶ A naïve approach would be $O(n)$
- ▶ A less naïve approach typically involves *kd*-trees
- ▶ These tend to scale poorly in very high dimensions
 - ▶ The rule of thumb¹ is, for dimension k and number of points N , $N \gg 2^k$
 - ▶ Otherwise you end up doing a nearly exhaustive search most of the time

¹Jacob E. Goodman, Joseph O'Rourke and Piotr Indyk (Ed.) (2004).
“Chapter 39 : Nearest neighbours in high-dimensional spaces”. Handbook of
Discrete and Computational Geometry (2nd ed.). CRC Press

Traditional Approach

- ▶ A naïve approach would be $O(n)$
- ▶ A less naïve approach typically involves *kd*-trees
- ▶ These tend to scale poorly in very high dimensions
 - ▶ The rule of thumb¹ is, for dimension k and number of points N , $N \gg 2^k$
 - ▶ Otherwise you end up doing a nearly exhaustive search most of the time
 - ▶ In these situations, approximation algorithms are typically used

¹Jacob E. Goodman, Joseph O'Rourke and Piotr Indyk (Ed.) (2004).
“Chapter 39 : Nearest neighbours in high-dimensional spaces”. Handbook of
Discrete and Computational Geometry (2nd ed.). CRC Press

Traditional Approach

- ▶ A naïve approach would be $O(n)$
- ▶ A less naïve approach typically involves *kd*-trees
- ▶ These tend to scale poorly in very high dimensions
 - ▶ The rule of thumb¹ is, for dimension k and number of points N , $N \gg 2^k$
 - ▶ Otherwise you end up doing a nearly exhaustive search most of the time
 - ▶ In these situations, approximation algorithms are typically used
- ▶ It's also not clear that they work for non- L_2 metrics

¹Jacob E. Goodman, Joseph O'Rourke and Piotr Indyk (Ed.) (2004).
"Chapter 39 : Nearest neighbours in high-dimensional spaces". Handbook of
Discrete and Computational Geometry (2nd ed.). CRC Press

“Big Data”

- ▶ Sometimes we have to deal with large amounts of data

“Big Data”

- ▶ Sometimes we have to deal with large amounts of data
- ▶ Traditionally we’ve put that data in SQL tables

“Big Data”

- ▶ Sometimes we have to deal with large amounts of data
- ▶ Traditionally we've put that data in SQL tables
- ▶ Scaling SQL databases is a pain in the ass

“Big Data”

- ▶ Sometimes we have to deal with large amounts of data
- ▶ Traditionally we've put that data in SQL tables
- ▶ Scaling SQL databases is a pain in the ass
 - ▶ Explicit sharding breaks joins

“Big Data”

- ▶ Sometimes we have to deal with large amounts of data
- ▶ Traditionally we've put that data in SQL tables
- ▶ Scaling SQL databases is a pain in the ass
 - ▶ Explicit sharding breaks joins
 - ▶ Have to worry about node availability yourself

“Big Data”

- ▶ Sometimes we have to deal with large amounts of data
- ▶ Traditionally we’ve put that data in SQL tables
- ▶ Scaling SQL databases is a pain in the ass
 - ▶ Explicit sharding breaks joins
 - ▶ Have to worry about node availability yourself
 - ▶ A lot of engineering work

“Big Data”

- ▶ Sometimes we have to deal with large amounts of data
- ▶ Traditionally we’ve put that data in SQL tables
- ▶ Scaling SQL databases is a pain in the ass
 - ▶ Explicit sharding breaks joins
 - ▶ Have to worry about node availability yourself
 - ▶ A lot of engineering work
- ▶ It’s a hard problem and you have to give up many of the benefits of SQL to solve it.

Schema-less NoSQL data stores

- ▶ Recently there has been a movement to use distributed schema-less data stores instead

Schema-less NoSQL data stores

- ▶ Recently there has been a movement to use distributed schema-less data stores instead
- ▶ These also happen to be a pain in the ass

Schema-less NoSQL data stores

- ▶ Recently there has been a movement to use distributed schema-less data stores instead
- ▶ These also happen to be a pain in the ass
 - ▶ Don't hate the player, hate the scale

Schema-less NoSQL data stores

- ▶ Recently there has been a movement to use distributed schema-less data stores instead
- ▶ These also happen to be a pain in the ass
 - ▶ Don't hate the player, hate the scale
- ▶ Conform to a map interface typically

Schema-less NoSQL data stores

- ▶ Recently there has been a movement to use distributed schema-less data stores instead
- ▶ These also happen to be a pain in the ass
 - ▶ Don't hate the player, hate the scale
- ▶ Conform to a map interface typically
 - ▶ `put(Key k, Value v)`
 - ▶ `get(Key k)`
 - ▶ `delete(Key k)`

Schema-less NoSQL data stores

- ▶ Recently there has been a movement to use distributed schema-less data stores instead
- ▶ These also happen to be a pain in the ass
 - ▶ Don't hate the player, hate the scale
- ▶ Conform to a map interface typically
 - ▶ `put(Key k, Value v)`
 - ▶ `get(Key k)`
 - ▶ `delete(Key k)`
- ▶ Examples of these are HBase, Cassandra, MongoDB, MemcacheDB

Schema-less NoSQL data stores

- ▶ Recently there has been a movement to use distributed schema-less data stores instead
- ▶ These also happen to be a pain in the ass
 - ▶ Don't hate the player, hate the scale
- ▶ Conform to a map interface typically
 - ▶ `put(Key k, Value v)`
 - ▶ `get(Key k)`
 - ▶ `delete(Key k)`
- ▶ Examples of these are HBase, Cassandra, MongoDB, MemcacheDB
- ▶ It would be very nice to be able to use this to find nearest neighbors, but how?

Near Neighbor Searches

- ▶ Often we **need** high dimension (see previous talk)

Near Neighbor Searches

- ▶ Often we **need** high dimension (see previous talk)
- ▶ Often we have **many** points

Near Neighbor Searches

- ▶ Often we **need** high dimension (see previous talk)
- ▶ Often we have **many** points
- ▶ Often we'll accept an approximation

Near Neighbor Searches

- ▶ Often we **need** high dimension (see previous talk)
- ▶ Often we have **many** points
- ▶ Often we'll accept an approximation
- ▶ Often we are looking for data in a fixed radius.

Locality Sensitive Hashing

- ▶ **Locality Sensitive Hashing** is a probabilistic technique to group “close” vectors according to a given metric.

Locality Sensitive Hashing

- ▶ **Locality Sensitive Hashing** is a probabilistic technique to group “close” vectors according to a given metric.
- ▶ We can use this to group our “near” vectors into buckets

Locality Sensitive Hashing

- ▶ **Locality Sensitive Hashing** is a probabilistic technique to group “close” vectors according to a given metric.
- ▶ We can use this to group our “near” vectors into buckets
- ▶ You can construct multiple hash functions (i.e. families) and compose to increase the accuracy at the expense of runtime complexity

Locality Sensitive Hashing

- ▶ **Locality Sensitive Hashing** is a probabilistic technique to group “close” vectors according to a given metric.
- ▶ We can use this to group our “near” vectors into buckets
- ▶ You can construct multiple hash functions (i.e. families) and compose to increase the accuracy at the expense of runtime complexity
- ▶ You can use multiple LSH functions and put the same input data into each bucket, thereby increasing accuracy at the expense of space complexity

Locality Sensitive Hashing: The Cons

- ▶ Different hash function for different distance metrics (not all have them)

Locality Sensitive Hashing: The Cons

- ▶ Different hash function for different distance metrics (not all have them)
 - ▶ Exist for the biggest ones

Locality Sensitive Hashing: The Cons

- ▶ Different hash function for different distance metrics (not all have them)
 - ▶ Exist for the biggest ones
 - ▶ L_k for $k > 0$, cosine-distance, min-hash, kernel-based metrics (i.e. machine learned distance metrics)

Locality Sensitive Hashing: The Cons

- ▶ Different hash function for different distance metrics (not all have them)
 - ▶ Exist for the biggest ones
 - ▶ L_k for $k > 0$, cosine-distance, min-hash, kernel-based metrics (i.e. machine learned distance metrics)
- ▶ Not all LSH functions have theoretical bounds about accuracy

Locality Sensitive Hashing: The Cons

- ▶ Different hash function for different distance metrics (not all have them)
 - ▶ Exist for the biggest ones
 - ▶ L_k for $k > 0$, cosine-distance, min-hash, kernel-based metrics (i.e. machine learned distance metrics)
- ▶ Not all LSH functions have theoretical bounds about accuracy
 - ▶ Almost all research focuses on **nearest** neighbor searches

Locality Sensitive Hashing: The Cons

- ▶ Different hash function for different distance metrics (not all have them)
 - ▶ Exist for the biggest ones
 - ▶ L_k for $k > 0$, cosine-distance, min-hash, kernel-based metrics (i.e. machine learned distance metrics)
- ▶ Not all LSH functions have theoretical bounds about accuracy
 - ▶ Almost all research focuses on **nearest** neighbor searches
 - ▶ Practical alternative is to sample your data and measure to determine parameters empirically

Stable Distributions and the L_k metric

- ▶ Based on the research of Piotr Indyk, et. al.²

²Datar, M.; Immorlica, N., Indyk, P., Mirrokni, V.S. (2004).
“Locality-Sensitive Hashing Scheme Based on p-Stable Distributions”.
Proceedings of the Symposium on Computational Geometry.

Stable Distributions and the L_k metric

- ▶ Based on the research of Piotr Indyk, et. al.²
- ▶ They found that 1-stable and 2-stable distributions could be used to construct families of locality sensitive hashes for L_1 and L_2 metrics

²Datar, M.; Immorlica, N., Indyk, P., Mirrokni, V.S. (2004).
“Locality-Sensitive Hashing Scheme Based on p-Stable Distributions”.
Proceedings of the Symposium on Computational Geometry.

Stable Distributions and the L_k metric

- ▶ Based on the research of Piotr Indyk, et. al.²
- ▶ They found that 1-stable and 2-stable distributions could be used to construct families of locality sensitive hashes for L_1 and L_2 metrics
- ▶ What the hell are p -stable distributions?

²Datar, M.; Immorlica, N., Indyk, P., Mirrokni, V.S. (2004).
“Locality-Sensitive Hashing Scheme Based on p -Stable Distributions”.
Proceedings of the Symposium on Computational Geometry.

Stable Distributions and the L_k metric

- ▶ Based on the research of Piotr Indyk, et. al.²
- ▶ They found that 1-stable and 2-stable distributions could be used to construct families of locality sensitive hashes for L_1 and L_2 metrics
- ▶ What the hell are p -stable distributions?
 - ▶ It's complicated and can be considered a black-box if you like (tune out for the next minute or so)

²Datar, M.; Immorlica, N., Indyk, P., Mirrokni, V.S. (2004).
“Locality-Sensitive Hashing Scheme Based on p -Stable Distributions”.
Proceedings of the Symposium on Computational Geometry.

Stable Distributions and the L_k metric

- ▶ Based on the research of Piotr Indyk, et. al.²
- ▶ They found that 1-stable and 2-stable distributions could be used to construct families of locality sensitive hashes for L_1 and L_2 metrics
- ▶ What the hell are p -stable distributions?
 - ▶ It's complicated and can be considered a black-box if you like (tune out for the next minute or so)
 - ▶ If you draw a vector \mathbf{a} from a p -stable distribution X , $\mathbf{a} \cdot (\mathbf{v}_1 - \mathbf{v}_2)$ is distributed exactly as $\|\mathbf{v}_1 - \mathbf{v}_2\|_p X$

²Datar, M.; Immorlica, N., Indyk, P., Mirrokni, V.S. (2004).
“Locality-Sensitive Hashing Scheme Based on p -Stable Distributions”.
Proceedings of the Symposium on Computational Geometry.

Stable Distributions and the L_k metric

- ▶ Based on the research of Piotr Indyk, et. al.²
- ▶ They found that 1-stable and 2-stable distributions could be used to construct families of locality sensitive hashes for L_1 and L_2 metrics
- ▶ What the hell are p -stable distributions?
 - ▶ It's complicated and can be considered a black-box if you like (tune out for the next minute or so)
 - ▶ If you draw a vector \mathbf{a} from a p -stable distribution X , $\mathbf{a} \cdot (\mathbf{v}_1 - \mathbf{v}_2)$ is distributed exactly as $\|\mathbf{v}_1 - \mathbf{v}_2\|_p X$
 - ▶ Know that the Normal distribution is 2-stable and the Cauchy distribution is 1-stable

²Datar, M.; Immorlica, N., Indyk, P., Mirrokni, V.S. (2004).

"Locality-Sensitive Hashing Scheme Based on p -Stable Distributions".

Proceedings of the Symposium on Computational Geometry.

Some Intuition

- ▶ Take the real number line and split it up into segments of length r , we can assign each segment an index and hash vectors into these segments: $\mathbf{v} \rightarrow \lfloor \frac{\mathbf{a} \cdot \mathbf{v}}{r} \rfloor$.

Some Intuition

- ▶ Take the real number line and split it up into segments of length r , we can assign each segment an index and hash vectors into these segments: $\mathbf{v} \rightarrow \lfloor \frac{\mathbf{a} \cdot \mathbf{v}}{r} \rfloor$.
- ▶ $\frac{\mathbf{a} \cdot (\mathbf{v}_1 - \mathbf{v}_2)}{r}$ is distributed as $\frac{\|\mathbf{v}_1 - \mathbf{v}_2\|_p}{r} X$ due to X being p -stable

Some Intuition

- ▶ Take the real number line and split it up into segments of length r , we can assign each segment an index and hash vectors into these segments: $\mathbf{v} \rightarrow \lfloor \frac{\mathbf{a} \cdot \mathbf{v}}{r} \rfloor$.
- ▶ $\frac{\mathbf{a} \cdot (\mathbf{v}_1 - \mathbf{v}_2)}{r}$ is distributed as $\frac{\|\mathbf{v}_1 - \mathbf{v}_2\|_p}{r} X$ due to X being p -stable
- ▶ So \mathbf{v}_1 and \mathbf{v}_2 should map to the same segment with high probability if they're within radius r of each other.

Some Intuition

- ▶ Take the real number line and split it up into segments of length r , we can assign each segment an index and hash vectors into these segments: $\mathbf{v} \rightarrow \lfloor \frac{\mathbf{a} \cdot \mathbf{v}}{r} \rfloor$.
- ▶ $\frac{\mathbf{a} \cdot (\mathbf{v}_1 - \mathbf{v}_2)}{r}$ is distributed as $\frac{\|\mathbf{v}_1 - \mathbf{v}_2\|_p}{r} X$ due to X being p -stable
- ▶ So \mathbf{v}_1 and \mathbf{v}_2 should map to the same segment with high probability if they're within radius r of each other.
- ▶ Different choices of \mathbf{a} can form families of these hash functions.

Spatial Search

- ▶ I've begun creating a simple library to assist in the use of these locality sensitive hashes

Spatial Search

- ▶ I've begun creating a simple library to assist in the use of these locality sensitive hashes
- ▶ It's Datastore agnostic

Spatial Search

- ▶ I've begun creating a simple library to assist in the use of these locality sensitive hashes
- ▶ It's Datastore agnostic
- ▶ L_1 and L_2 are implemented as well as a utility to assist in choosing parameters

Spatial Search

- ▶ I've begun creating a simple library to assist in the use of these locality sensitive hashes
- ▶ It's Datastore agnostic
- ▶ L_1 and L_2 are implemented as well as a utility to assist in choosing parameters
- ▶ Next up is min-hash

Spatial Search

- ▶ I've begun creating a simple library to assist in the use of these locality sensitive hashes
- ▶ It's Datastore agnostic
- ▶ L_1 and L_2 are implemented as well as a utility to assist in choosing parameters
- ▶ Next up is min-hash
- ▶ <https://github.com/cestella/SpatialSearch>

Conclusion

- ▶ Thanks for your attention
- ▶ Follow me on twitter @casey_stella
- ▶ Find me at
 - ▶ <http://caseystella.com>
 - ▶ <https://github.com/cestella>
- ▶ PS. If you like this...Explorys is hiring!