On the Impedance Mismatch

Exemplified with Vector Databases

Motivation

THE OBJECT-RELATIONAL IMPEDANCE MISMATCH

NOSQL Goals

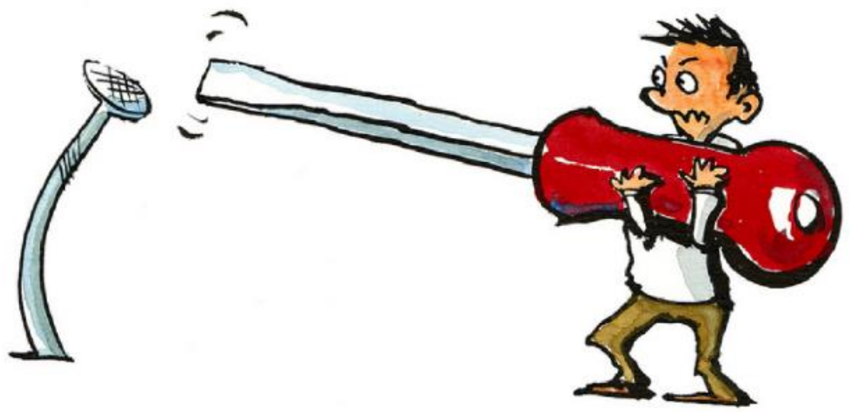
Recall the NOSQL Goals:

- Schemaless: Allow flexible (even runtime) schema definition [data structure]
- Reliability / availability: Keep delivering service even if its software or hardware components fail [distribution]
- Scalability: Continuously evolve to support a growing amount of tasks [distribution]
- Efficiency: How well the system performs, usually measured in terms of response *time* (latency) and *throughput* (bandwith) [distribution]

Impedance Mismatch: Some History

Of hammers and nails...

The Law of the Hammer



If the only tool you have is a hammer,
everything looks like a nail.

Abraham Maslow - The Psychology of Science - 1966

Petra Selmer, Advances in Data Management 2012

Impedance Mismatch: Some History

The Law of the Relational Database

If the only tool you have is a relational database, everything looks like a table.

A Walk in Graph Databases - 2012

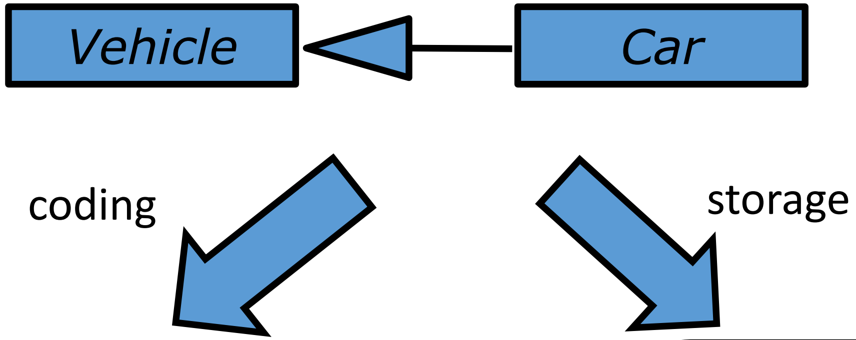
Petra Selmer, Advances in Data Management 2012

Definition

- When two tecnologies interact but they are grounded in diferent models or paradigms, a translation between the elements of both models is then mandatory to enable **interoperability**. This overhead (in performance) to execute the map between both models is known as impedance mismatch

- In this course, we will study the potential impedance mismatch between an application and the database

Example: The Object-Relational Impedance Mismatch



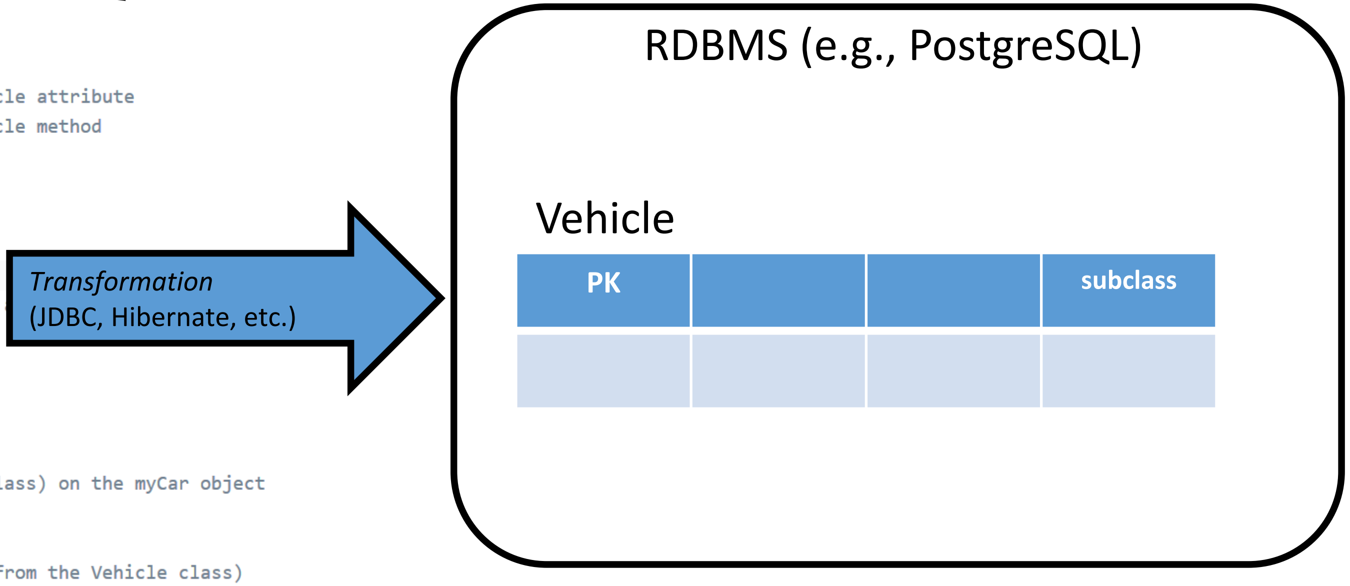

```java
class Vehicle {
  protected String brand = "Ford";        // Vehicle attribute
  public void honk() {                    // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";   // Car
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar object
    myCar.honk();

    // Display the value of the brand attribute (from the Vehicle class)
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
}
```



Other Types of Impedance Mismatch

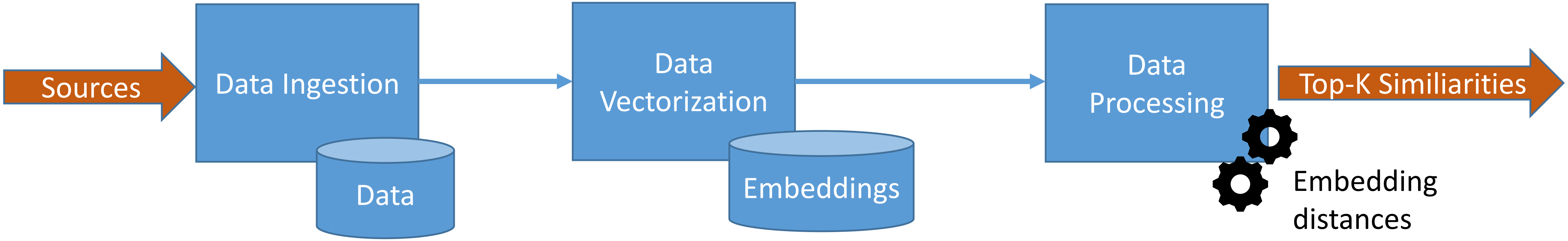
- NOSQL has introduced new data models
  - Graph data model
  - Document-oriented databases
  - Key-value (~hash tables)
  - Embeddings (~vectorized data representations)
- Other programming paradigms have also appeared or become trendy
- As such, we may have impedance mismatch between any programming paradigm and any database data model
- The impedance mismatch is accordingly a complex issue that may compromise the performance of any [Big Data] system
  - **As such, during this course we will learn how to model the NOSQL databases to reduce the impedance mismatch**

Impedance Mismatch on Vector Data

An Example Based on Embeddings

An Introduction to Embeddings

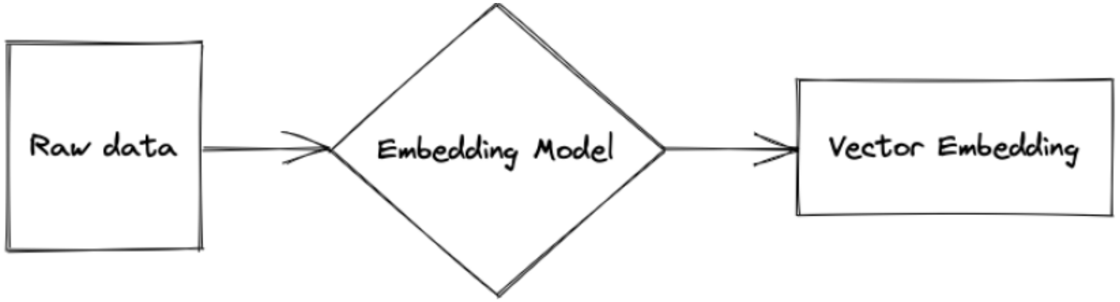
- Embeddings are grounded on geometry and they are an alternative to data mining and machine learning
  - Represent the data in terms of a fixed-size vector
  - Compare the vectors (via distances: e.g., cosine distance)

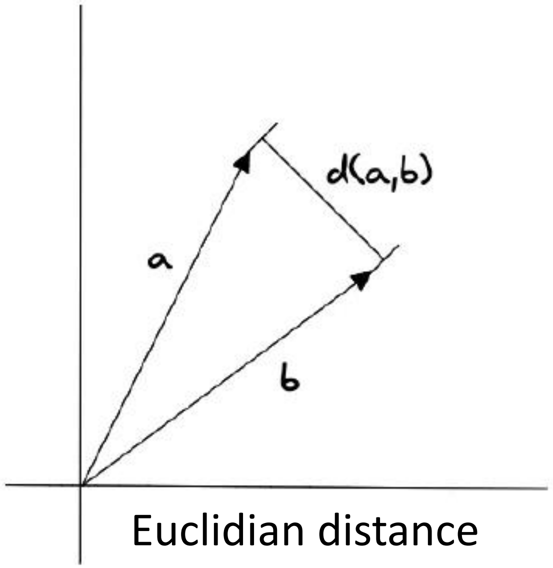


Embeddings are nowadays massively used for recommenders, sentiment analysis, chatbots, search engines, etc.

An Introduction to Embeddings

- The heavy lifting is done by a model generating the embeddings
  - Suppose a space (also called *feature space*) then:
    - Two similar instances should be represented with similar vectors and thus, close to each other in the space (i.e., short distance)
    - Two dissimilar instances should be represented with dissimilar vectors and therefore, far away from each other in the space (i.e., large distance)

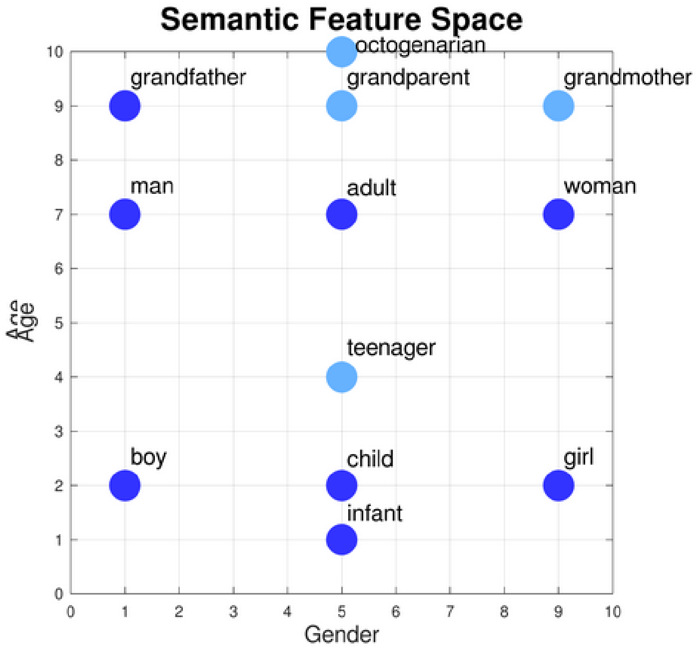




Euclidian distance

An Introduction to Embeddings

- **Word2Vec** is a popular embedding model for textual data. The embeddings it generates are called word embeddings and the space were they are placed semantic feature space
- Suppose a 2-D semantic feature space (age, gender)



Word Coordinates

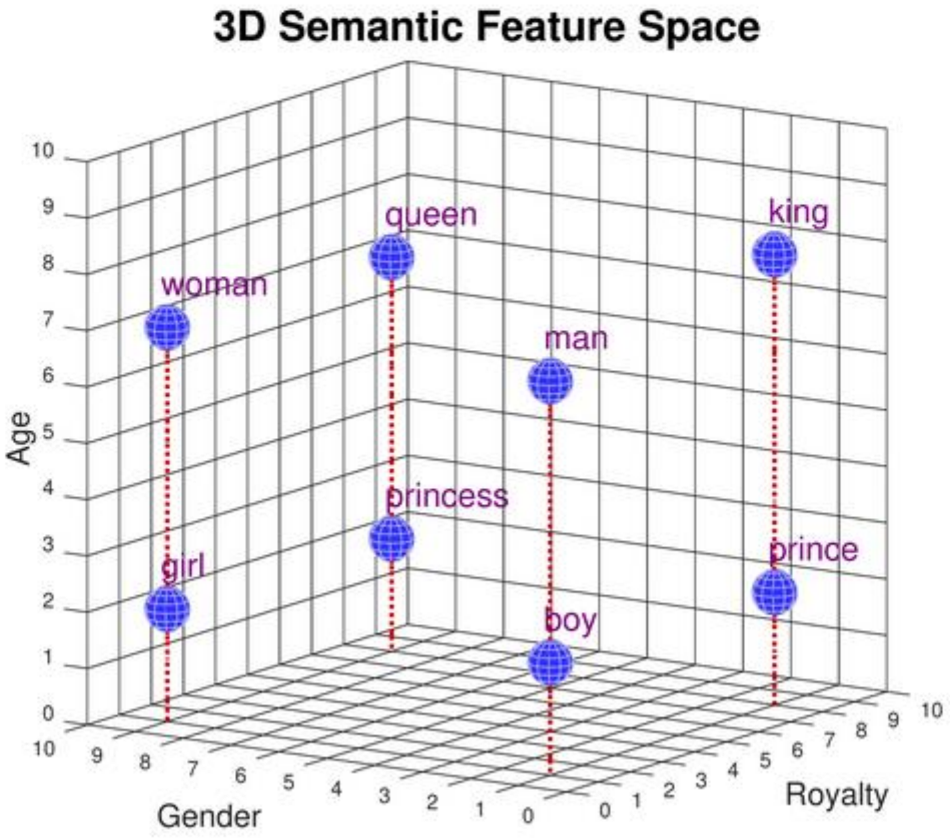
	Gender	Age
grandfather	[1,	9]
man	[1,	7]
adult	[5,	7]
woman	[9,	7]
boy	[1,	2]
child	[5,	2]
girl	[9,	2]
infant	[5,	1]

Word Coordinates

	Gender	Age
grandmother	[9,	9]
grandparent	[5,	9]
octogenarian	[5,	10]
teenager	[5,	4]

An Introduction to Embeddings

- Suppose now a 3-D semantic feature space (age, gender, royalty)



Word Coordinates

	Gender	Age	Royalty
man	[1,	7,	1]
woman	[9,	7,	1]
boy	[1,	2,	1]
girl	[9,	2,	1]
king	[1,	8,	8]
queen	[9,	7,	8]
prince	[1,	2,	8]
princess	[9,	2,	8]

Computing Embeddings

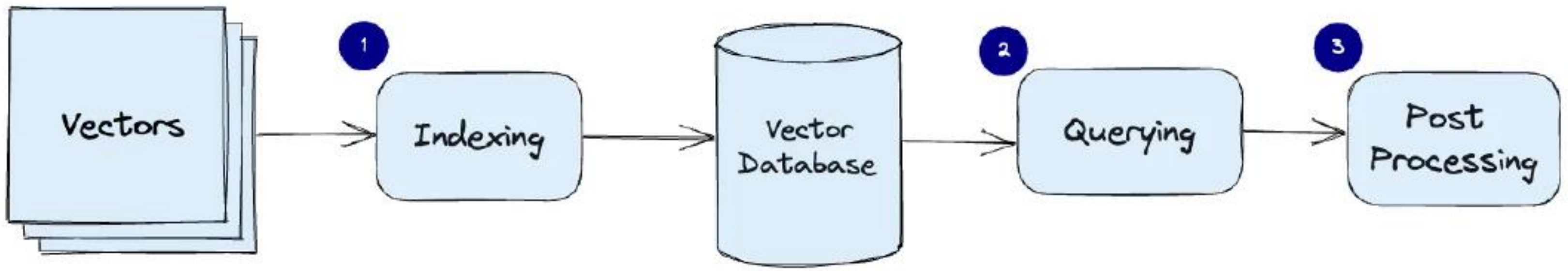
- Deciding the number of features is hard! Current models automatically create the feature space (dimensions of the space) and learn how to assign values to each dimension for each word
  - Word2Vec is a 2-layer neural network that originally created a 300-features space
  https://code.google.com/archive/p/word2vec/
  - To know more about how embeddings are computed check a nice tutorial at: https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/tutorial.html

Computing Distances Between Embeddings

- The generated vectors can be operated as regular vector arithmetic
  - Addition, substraction
  - Distances:
    - Euclidian Distance $d(\mathbf{a},\mathbf{b}) = \sqrt{(\mathbf{a_1}-\mathbf{b_1})^2 + (\mathbf{a_2}-\mathbf{b_2})^2 + \ldots + (\mathbf{a_n}-\mathbf{b_n})^2}$
    - Cosine Similarity $\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\cdot\sqrt{\sum_{i=1}^{n} B_i^2}},$
    - Manhattan distance $D(A,B) = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n| = \sum_{i=1}^{n} |a_i - b_i|$

Vector Databases

- Specialized databases to **store** and **process** vectors



- Indexing: meant to facilitate computing distances (i.e., vector neighbours)
  - Tree-based indexing
    - KD-trees
    - R-trees
  - Hash-based indexing
    - LSH: Locality-sensitive hashing

On the Impedance Mismatch of Vectors

- Vector databases were born to eliminate the huge impedance mismatch generated when storing and processing vectors in relational databases
- To understand the impedance mismatch, vectors are a good candidate because relational databases do not deal well with them!

- In this lab, you are asked to download a corpus (textual data), generate its embeddings and compute distances between them
    - With a relational database: PostgreSQL
    - With a vector database: Chroma
    - With a PostgreSQL extension: Pgvector [optional]

  And measure the impedance mismatch on time (note that we may also measure it on memory used, number of code lines needed or any other relevant factor of your interest)

Summary

- Impedance mismatch
- Practicing the impedance mismatch: exemplified with vector data
  - Introduction to embeddings
  - Computing embeddings
  - Processing embeddings
- Vector databases
  - Native vector data storage and processing

Bibliography

- Word2Vec: https://code.google.com/archive/p/word2vec/
- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding: https://arxiv.org/abs/1810.04805
- Attention Is All You Need: https://arxiv.org/abs/1706.03762
- Pinecone: https://www.pinecone.io/
- Chroma: https://www.trychroma.com/
- Pgvector: https://github.com/pgvector/pgvector
- Dietrich S.W. and Urban S.D. An Advanced Course in Database Systems: Beyond Relational Databases. Prentice Hall, Upper Saddle River, NJ, 2005.
- Object Database Management Systems: The Resource Portal for Education and Research, http://odbms.org