Big Data Integration

Prof. Giovanni Simonini

Università degli Studi di Modena e Reggio Emilia

simonini@unimore.it

- Postdoc. Researcher

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- PhD in ICT – 2024

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Visiting Positions

HPI

- Assoc. Prof. DIEF

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Postdoc. Researcher

Massachusetts Institute of Technology

- PhD in ICT – 2016

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Visiting Positions

UNIVERSITY OF MICHIGAN

QCRI
معهد قطر لبحوث الحوسبة
Qatar Computing Research Institute

www.lucazecchini.it ***www.giovannisimonini.com***

Ferrari

MASERATI

PAGANI Automobili Modena

LAMBORGHINI

DUCATI

Example of Data Integration (1)

1. Company A acquires company B and wants to merge the customer DB, **but**:
   - DBs have different schemas
     - E.g., “Contact” vs “name” + “email” + “phone”
   - Entities are duplicated and there is no unique ID
     - E.g., “Bill Clinton” vs “William Jefferson Clinton”
   - Inconsistencies
     - E.g., “Main St. 11, BOS, MA” vs “11 Main st., Boston, Massachusetts”

Example of Data Integration (2)

2. A regional public health organization has many suppliers that provides the same products, catalogs are available in CSV, **but**:
   - The schemas are different
     - E.g., price per unit vs. price for batch
   - Entities are duplicated and there is no unique ID
     - E.g., is “Bayer” and “Bayer Italia” the same customer?
   - Achronyms and Synonym
     - E.g., “paracetamol” vs “acetaminophen”

Example of Data Integration (3)

3. A data scientist wants to build a model to suggest the price for an e-commerce marketplace with some data gathered from the web, **but**:
   - The schemas are different
     - E.g., the Mega Pixel information is in the title vs. the actual “MP” attribute
   - Synonyms
     - E.g., “Reflex” vs “DSLR”
   - Errors
     - E.g., “1.2 MP” instead of “12 MP”

Why Big Data Integration?

- Big data integration =
  Big Data + Data Integration

- “Traditional” Data Integration is about “*easy access to multiple data sources*”

- Big data is all about the V’s

Why Big Data Integration?

- "Traditional" Data Integration is about "*easy access to multiple data sources*"
  - Virtual: mediated schema, query reformulation, link + fuse answers
  - Warehouse: materialized data, easy querying, consistency issues
  - **ETL**: Extract, Transform, and Load

- Big data is all about the V's
  - **Volume** of the data, collected and analyzed at high **Velocity**
    - E.g., Data lakes, Sensors Sta, Web Data, Linked Open Data, etc.
  - huge **Variety** of heterogenous data sources
    - E.g., Relational DB, KGs, CSV, NoSQL, etc.
  - data has a **Value** given by the application consuming it
    - Timely *almost correct* analysis is often better than an outdated correct analysis

Data Integration and ETL

- All the hevy-lisfting is done upfront
  - With limitations

1. Carefully design the final schema and map each source to it
   - Immutable schema for all the applications (we need to choose wisely)

1. Detect all the duplicated entities as we insert them
   - No distinction btw “hot data” and “cold data”
   - All the entities have the same *value* for all the application
   - We assume to know what a duplicate is, e.g., “iPhone XV 32GB grey” vs “iPhone XV 32GB blue”

1. Fuse each cluster of duplicate to generate one single record
   - We decide what the “truth” is for all the application
     - E.g., keep the most recent value for iPhone
       but what if the application is “predict the initial price to the market”

Shift from ETL to ELT

- What's changed?
  1. Storage has become cheaper and cheaper and the cloud had made it easy to manage large data infrastructure
     - Opportunity: Organizations can now store and analyze the Big Data
       - The Vs were manageble only for very large player (e.g., Google)
  2. Democratization of data science and "Data is the new oil" awareness:
     - Opportunity: Store the data and analyze it on-demand
  3. Data Lakes

Data Lakes (Big Data Vs)

- Are large heterogeneous collection of datasets
  - Allows to store structured, semi-structured and unstructured data
  - Volume
- Data is stored as-is
  - Without having to first structure it, and run different types of analytics
  - Metadata is not centralized and uniform
  - Variety
- Datasets in the collection change autonomously over time
  - Fast change propagation, no need to perform ETL
  - Velocity
- The data are transformed when the application is consuming it
  - Value

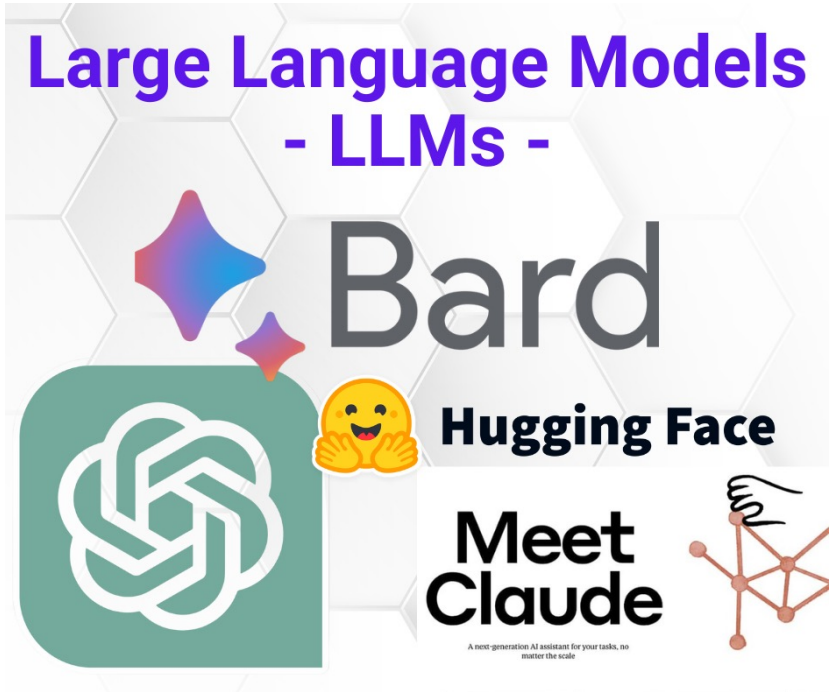
Data Lake (Big Data Vs)

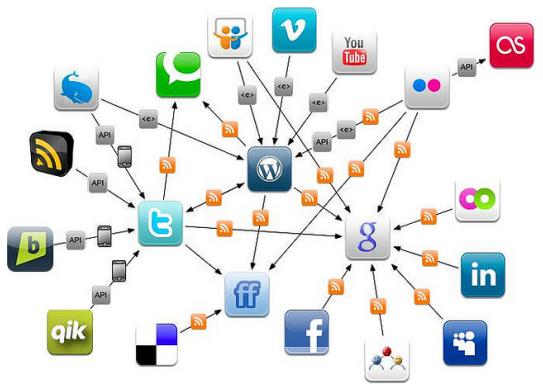
- Are large heterogeneous collection of datasets
  - Allows to store structured, semi-structured and unstructured data
  - Volume
- Data is stored as-is
  - Without having to first structure it, and run different types of analytics
  - Metadata is not centralized and uniform
  - Variety
- Datasets in the collection change autonomously over time
  - Fast change propagation, no need to perform ETL
  - Velocity
- The data are transformed when the application is consuming it
  - Value

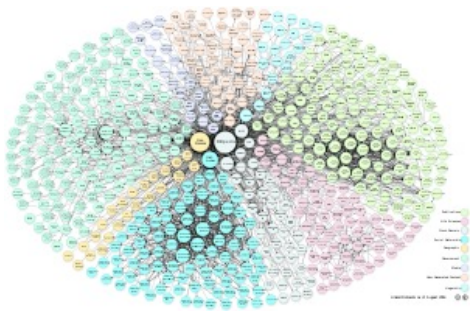
1. No unique, unified view: we need data discovery

2. New challenges for DI on heterogeneous data

3. On-demand integration is needed (Pay-as-you-go)



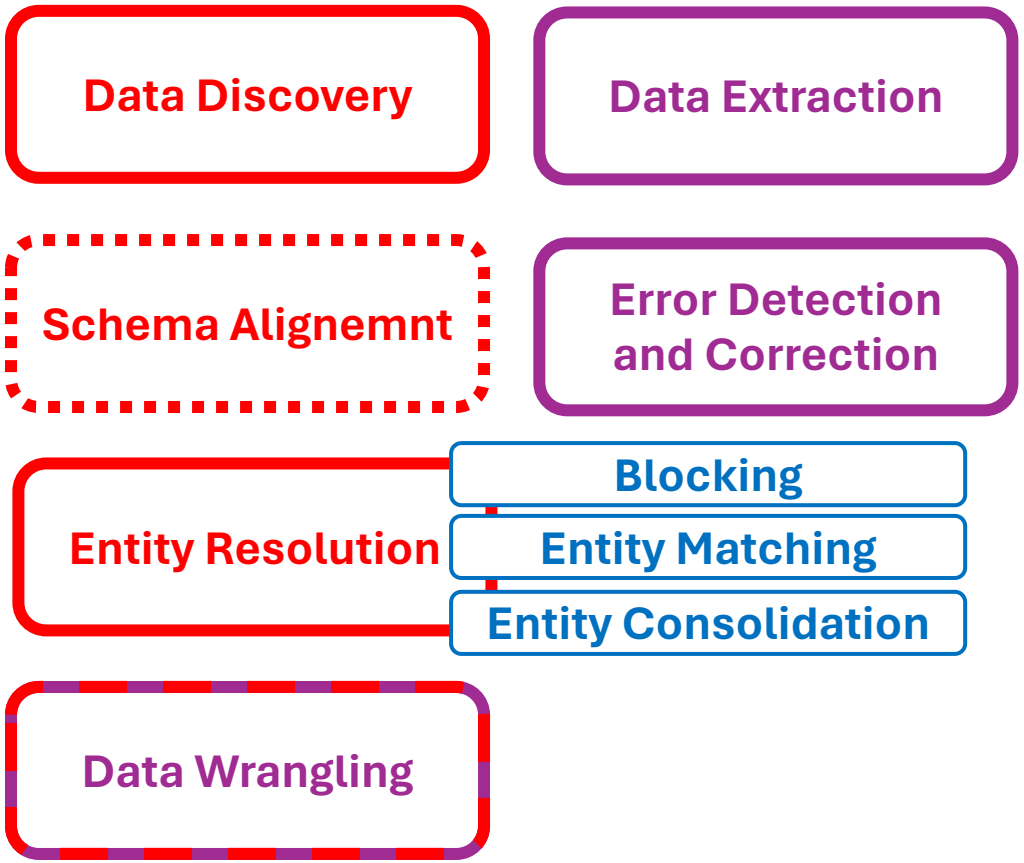




Not only Data Lakes

- Data of the Web
- Open Data
- Linked Data
- Data Markets
- LLMs training, finetuning and RAG

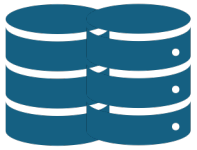
BDI problems and related problems



- Data discovery (tomorrow)
- Schema alignment
  - with large and hetherogenous dataset is often impractical
  - Schema agnostic techniques
    - No schema at all
    - Embedded with deep learning
- Entity Resolution
  - Quadratic problem, hard to scale!

Data Discovery

Find related data

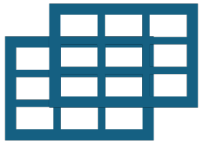


Joinable datasets

Data enrichment

Unionable datasets

Data augmentation



Overalp

Joinable and Unionable

Schema Alignment

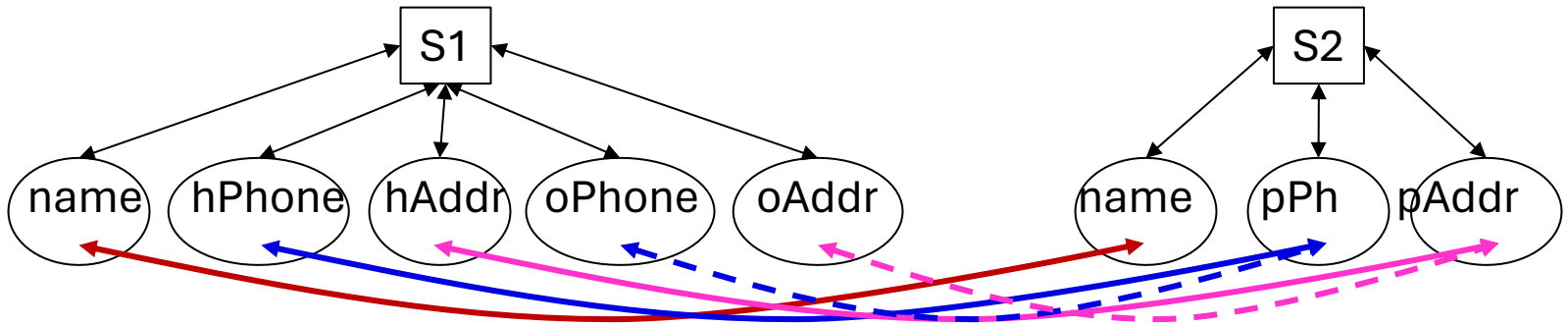
Schema Aligment in BDI

- Completely automated schema alignment is unfeasible
  - Cannot scale to 1000s of data sources
    - e.g., each dataset is a data source
    - Remember Velocity and Variety
  - Depends on the application
    - Why to align everything if we do not need certain attributes?

- When needed, we want to minimize human involvement:
  - Automatically create a mediated schema from a set of sources
  - Model uncertainty about semantics of attributes in sources
    - Uncertainty → probabilistic mediated schemas
    - Probabilistic mappings use weighted attribute correspondences

Probabilistic Mediated Schemas*

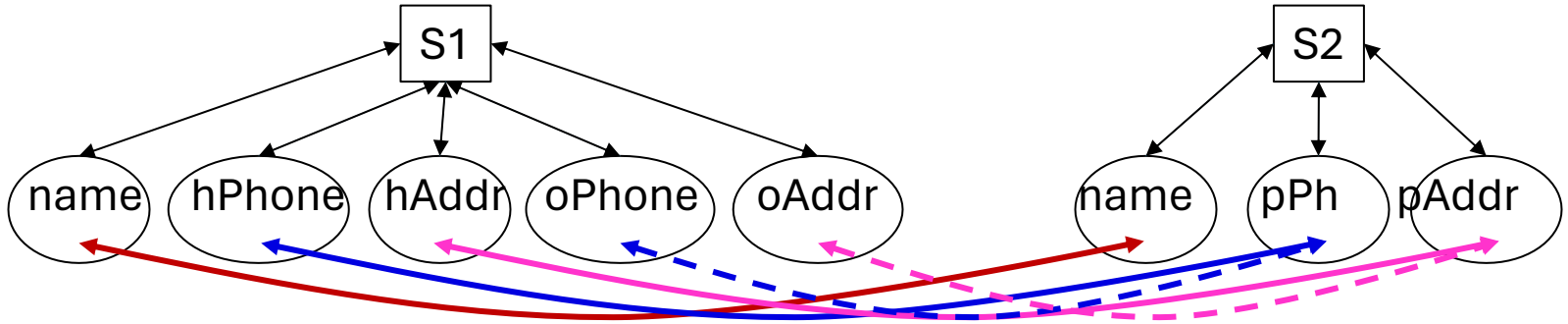
- Mediated schemas: automatically created by inspecting sources
  - Clustering of source attributes
    - based on (text) similarity
  - Volume, Variety of sources → uncertainty in accuracy of clustering



*Das Sarma, A., Dong, X., & Halevy, A. Bootstrapping pay-as-you-go data integration systems. SIGMOD 2008

Probabilistic Mediated Schemas*

- In S2, **pPh/pAdd** can either be a home or office phone number/address
- There are multiple ways to cluster the attributes and they correspond to different mediated schemas:
  - M1({name}, {phone, hP, oP}, {address, hA, oA})
  - M2({name}, {phone, hP}, {oP}, {address, oA}, {hA})
  - M3({name}, {phone, hP}, {oP}, {address, hA}, {oA})
  - M4({name}, {phone, oP}, {hP}, {address, oA}, {hA})
  - M5({name}, {phone}, {hP}, {oP}, {address}, {hA}, {oA})
- None of the listed mediated schemas is perfect



*Das Sarma, A., Dong, X., & Halevy, A. Bootstrapping pay-as-you-go data integration systems. SIGMOD 2008

Probabilistic Mediated Schemas*

- In S2, **pPh/pAdd** can either be a home or office phone number/address
- There are multiple ways to cluster the attributes and they correspond to different mediated schemas:
  - M1({name}, {phone, hP, oP}, {address, hA, oA})
  - M2({name}, {phone, hP}, {oP}, {address, oA}, {hA})
  - M3({name}, {phone, hP}, {oP}, {address, hA}, {oA})
  - M4({name}, {phone, oP}, {hP}, {address, oA}, {hA})
  - M5({name}, {phone}, {hP}, {oP}, {address}, {hA}, {oA})
- None of the listed mediated schemas is perfect
  - E.g., M1 cannot return correct answers for queries that contain both hP and oP (same cluster)
  - E.g., M3 favours home address and phone over office address and phone (similar M4)
- Probabilistic Mapping assigns a probability to each mapping
  - Answers have a probability
  - The final user (or application) will choose the “right” answer for the application at hand

*Das Sarma, A., Dong, X., & Halevy, A. Bootstrapping pay-as-you-go data integration systems. SIGMOD 2008

Schema alignemnt and Data Discovery

- In the Data Lake we store each source “**as-is**”

- Techniques for Table Union Search
  - To align the schemas of tables that we need to align
  - Typically, we have a starting table of interest and we search for “unionable” tables

- Techniques for Join Discovery
  - To align only the attributes that we need to perform a JOIN (or fuzzy-join)
  - Typically, we have a starting table of interest and we search for “joinable” tables

- Both “traditional” schema alignment and data discovery rely on similarities:
  1. Meta-data similarity (attributes name, documentation, table captions)
  2. Instance similarity (cell attribute values, entities, columns)
  - Typically, for 1. and 2. we employ string similarity

Entity Resolution

The “Father” of Entity Resolution

Halbert L. Dunn, M.D. (1896-1975), the leading figure in establishing a **national vital statistics** system in the United States.

Vital statistics: births, deaths, migration, marriages, divorces, etc.

- Medical doctor and statistician;
- Chief of the National Office of Vital Statistics from 1935 through 1960;
- Co-founder of the National Association for Public Health Statistics and Information Systems (NAPHSIS) and the Inter-American Statistics Institute (IASI).

H. Dunn: *Record Linkage*. American Journal of Public Health (AJPH) 36(12): 1412-1416 (1946)

EACH person in the world creates a Book of Life. This Book starts with birth and ends with death. Its pages are made up of the records of the principal events in life. Record linkage is the name given to the process of assembling the pages of this Book into a volume.

The Book has many pages for some and is but a few pages in length for others. In the case of a stillbirth, the entire volume is but a single page.

The person retains the same identity throughout the Book. Except for advancing age, he is the same person. Thinking backward he can remember the important pages of his Book even though he may have forgotten some of the words. To other persons, however, his identity must be proven. "Is the John Doe who enlists today in fact the same John Doe who was born eighteen years ago?"

Events of importance worth recording in the Book of Life are frequently put on record in different places since the person moves about the world throughout his lifetime. This makes it difficult to assemble this Book into a single compact volume. Yet, sometimes it is necessary to examine all of an individual's important records simultaneously. No one would read a novel, the pages of which were not assembled. Just so, it is necessary at times to link the various important records of a person's life.

The two most important pages in the Book of Life are the first one and the last one. Consequently, in the process of record linkage the uniting of the fact-of-death with the fact-of-birth has been given a special name, "death clearance."

Main focus on **death clearances**.

ER Formalization (1969)

Ivan P. Fellegi (1935), a Hungarian-Canadian statistician

I. Fellegi, A. Sunter: *A Theory for Record Linkage*. Journal of the American Statistical Association (JASA), 64(328), 1183-1210 (1969)

A mathematical model is developed to provide a theoretical framework for a computer-oriented solution to the problem of recognizing those records in two files which represent identical persons, objects or events (said to be *matched*).

A comparison is to be made between the recorded characteristics and values in two records (one from each file) and a decision made as to whether or not the members of the comparison-pair represent the same person or event, or whether there is insufficient evidence to justify either of these decisions at stipulated levels of error. These three decisions are referred to as *link* (A_1), a *non-link* (A_3), and a *possible link* (A_2). The first two decisions are called positive dispositions.

The two types of error are defined as the error of the decision A_1 when the members of the comparison pair are in fact unmatched, and the error of the decision A_3 when the members of the comparison pair are, in fact matched. The probabilities of these errors are defined as

$$\mu = \sum_{\gamma \in \Gamma} u(\gamma) P(A_1 | \gamma)$$

and

$$\lambda = \sum_{\gamma \in \Gamma} m(\gamma) P(A_3 | \gamma)$$

respectively where $u(\gamma)$, $m(\gamma)$ are the probabilities of realizing γ (a comparison vector whose components are the coded agreements and disagreements on each characteristic) for unmatched and matched record pairs respectively. The summation is over the whole comparison space Γ of possible realizations.

A *linkage rule* assigns probabilities $P(A_1|\gamma)$, and $P(A_2|\gamma)$, and $P(A_3|\gamma)$ to each possible realization of $\gamma \in \Gamma$. An optimal linkage rule $L(\mu, \lambda, \Gamma)$ is defined for each value of (μ, λ) as the rule that minimizes $P(A_2)$ at those error levels. In other words, for fixed levels of error, the rule minimizes the probability of failing to make positive dispositions.

A theorem describing the construction and properties of the optimal linkage rule and two corollaries to the theorem which make it a practical working tool are given.

A Real Use Case Scenario

- A logistic company that works with medical supplies acquires new customers.
- These customers already own their product catalogs with thousands of products.
- In each catalog the same product has a different identifier, and a similar but not equal description.
- The logistic company wants to unify the catalogs (i.e., giving a unique id to the same product) in order to better organize its warehouse.

Catalog A

PID	Title	Description
P123X	Syringe 10x10	Syringe 10 ml – 10 pack
P123Y	Syringe 10x100	Syringe 10 ml – 100 pack
P456A	Insulin needle 4x10	Hypodermic insulin needle 4 mm – 10 pack
…	…	…

Catalog B

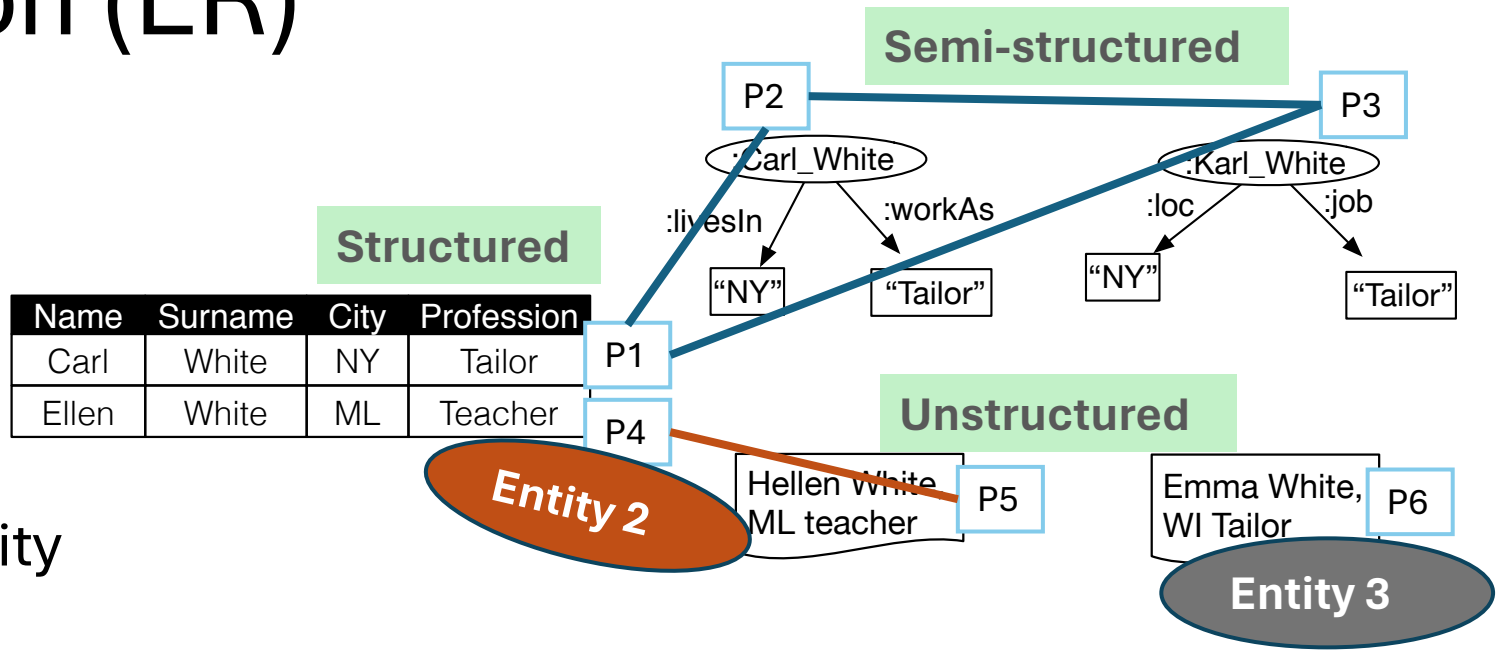
ID	Name	Description
1	Syringes 10 ml	Syringe 10 ml x 10 pieces
2	Syringes 10 ml big pack	Syringe 10 ml x 100 pieces
3	Small needle	Needle for insulin 4 mm 10 pieces
…	…	…

Why ER is a neverending problem?

1. **New paradigm to “share” the data**
   - E.g., Data Lake, Data Mesh, Data Marketplaces, etc.
2. **Everyone design their own database**
   - Different representation for the same entities (e.g., “name” vs “name” + “surname”)
3. **Real-world is ambiguous**
   - “ellen@cs.uni.edu” alias for “ellen.mylongsurname@uni.edu”
   - “Bill Clinton” same as “William Jefferson Clinton”
   - “iPhone XV pro 256GB natural titanium” vs “iPhone XV pro 256GB black titanium”
4. **Errors are often out of our control**
   - Too expensive to catch all errors during data entry / data loading
   - Bugs in data pipelines
   - Outdated data, etc.
5. **We are eager for Data**
   - Storage price bounds the data that we collect, not the resource available to fix it
     - Resources are: time, computational power, human-in-the-loop, money, etc.

Entity Resolution (ER)

- Each entity can be represented in different ways in different data sources
  - Called *profiles* of the entity



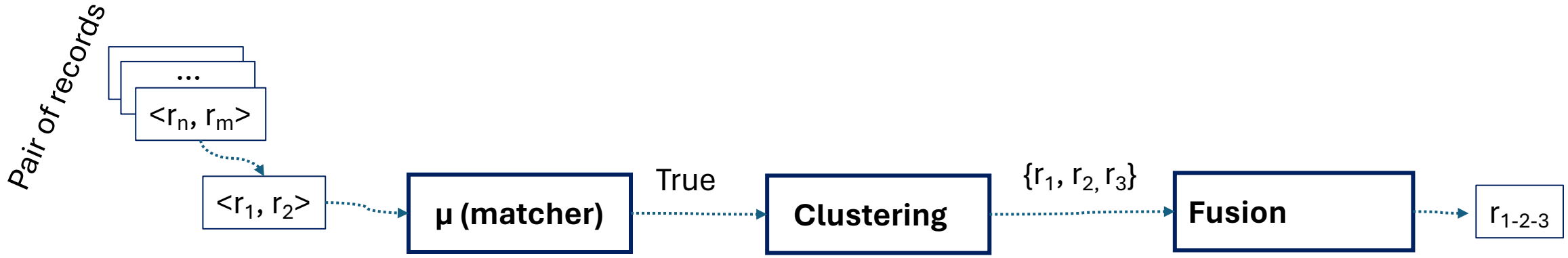
- Entity Resolution:
  - Task of *resolving* different profiles of the same real-world entity in databases
  - Resolving means:
    1. To cluster them
    2. To return a single record representing the cluster

Entity Resolution Workflow

1. **Candidate Generation:**
   a. Given a dataset (or more datasets) with possible duplicates we analyze a pair of record at a time
2. **Matching:**
   a. A binary matching function is employed to assess if a pair of profile is “matching”, which can be:
      - $\mu : D \times D \rightarrow \{0,1\}$, i.e., returns match/non-match
      - $\mu : D \times D \rightarrow [0,1]$, i,e., return a probability
3. **Clustering**:
   a. A matching graph is built using the output of the μ
   b. A clustering algorithm is applied to partition the graph
      - e.g., with $\mu : D \times D \rightarrow \{0,1\}$, a transitive closure is a common chouice
4. **Fusion**:
   1. Choose the final schema of the representative record
   2. Apply a consolidation function for each attribute to remove ambiguities

Entity Resolution Workflow (example)

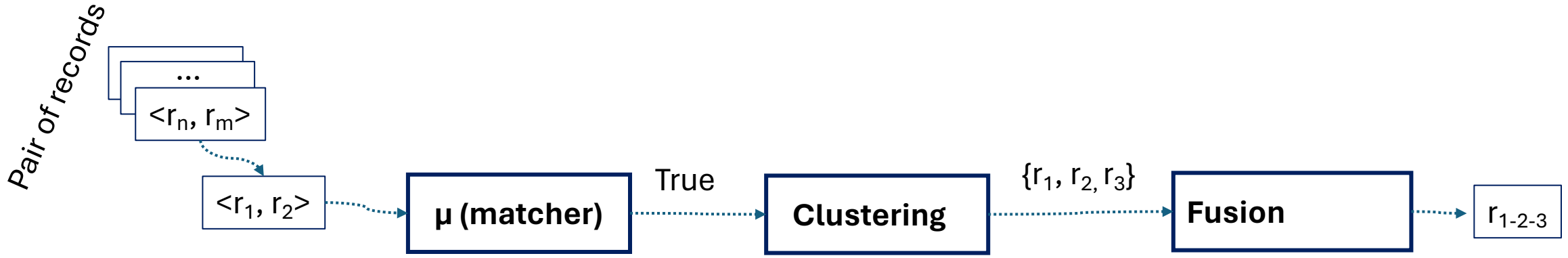
1. **Candidate Generation:**
   pairs of candidate match
2. **Matching:**
   deciding wether a pair is an actual match
3. **Clustering:**
   resolve inconsistencies of match, e.g., $r_1 \approx r_2$, $r_2 \approx r_3$, but $r_1 \not\approx r_2$
4. **Fusion (aka Consolidation):**
   $\{r_1, r_2, r_3\} \rightarrow r_{gold}$



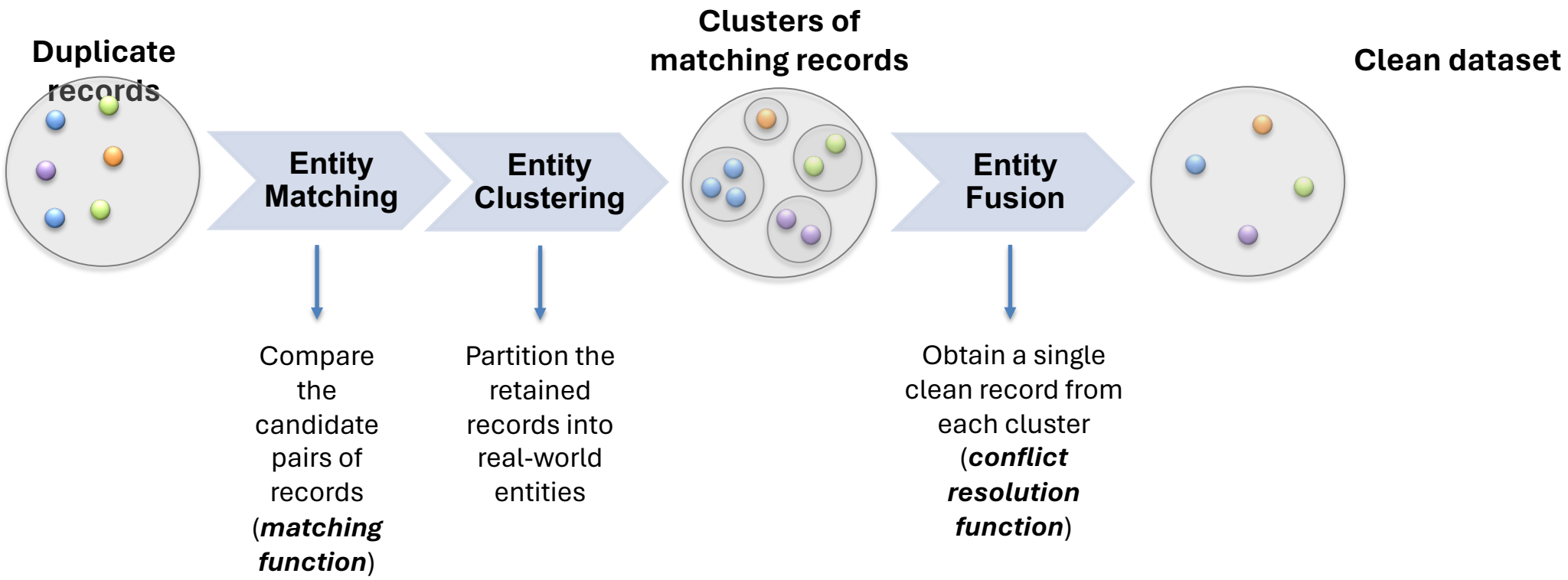
Entity Resolution Workflow (example)

Naïve solution is to compare all possible pairs of profiles **$O(n^2)$**

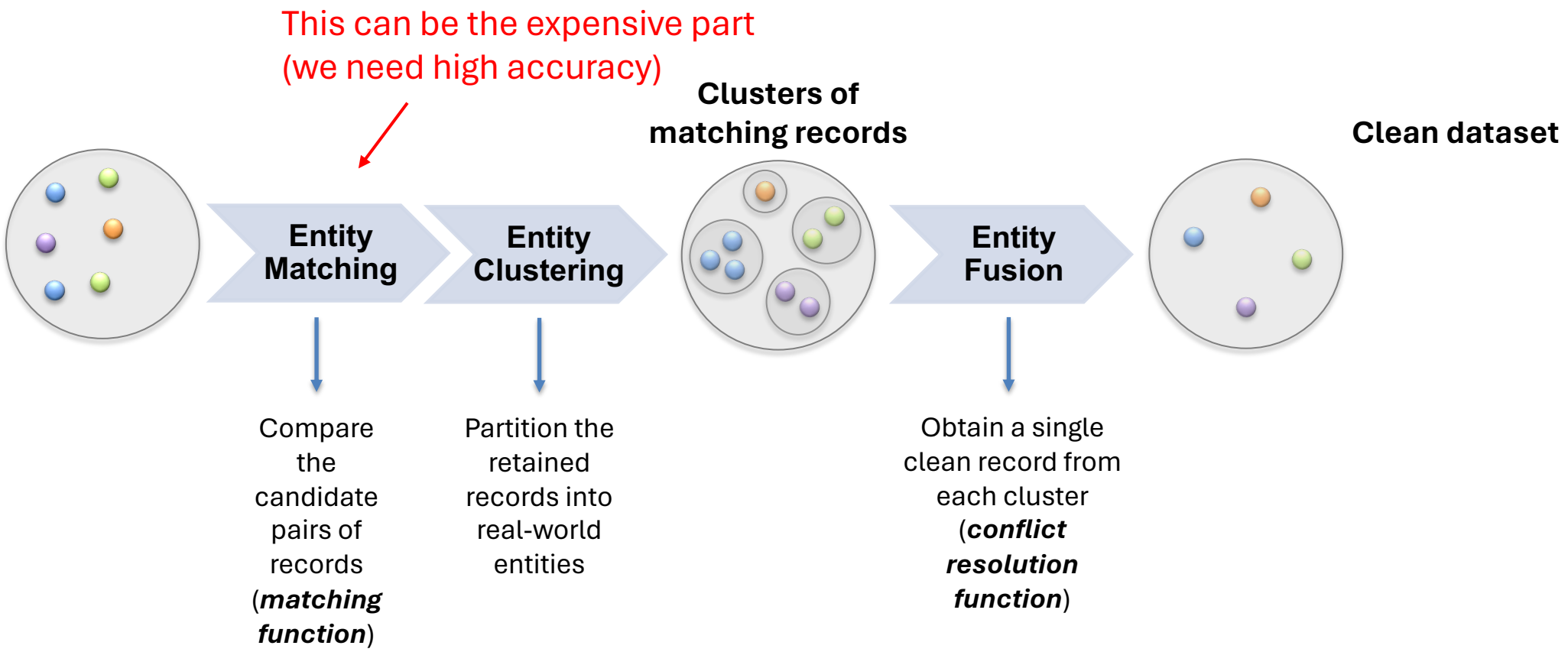
1. **Candidate Generation:**
   pairs of candidate match
2. **Matching:**
   deciding wether a pair is an actual match
3. **Clustering:**
   resolve inconsistencies of match, e.g., $r_1 \approx r_2$, $r_2 \approx r_3$, but $r_1 \not\approx r_2$
4. **Fusion (aka Consolidation):**
   $\{r_1, r_2, r_3\} \rightarrow r_{gold}$



ER: the standard pipeline for big data



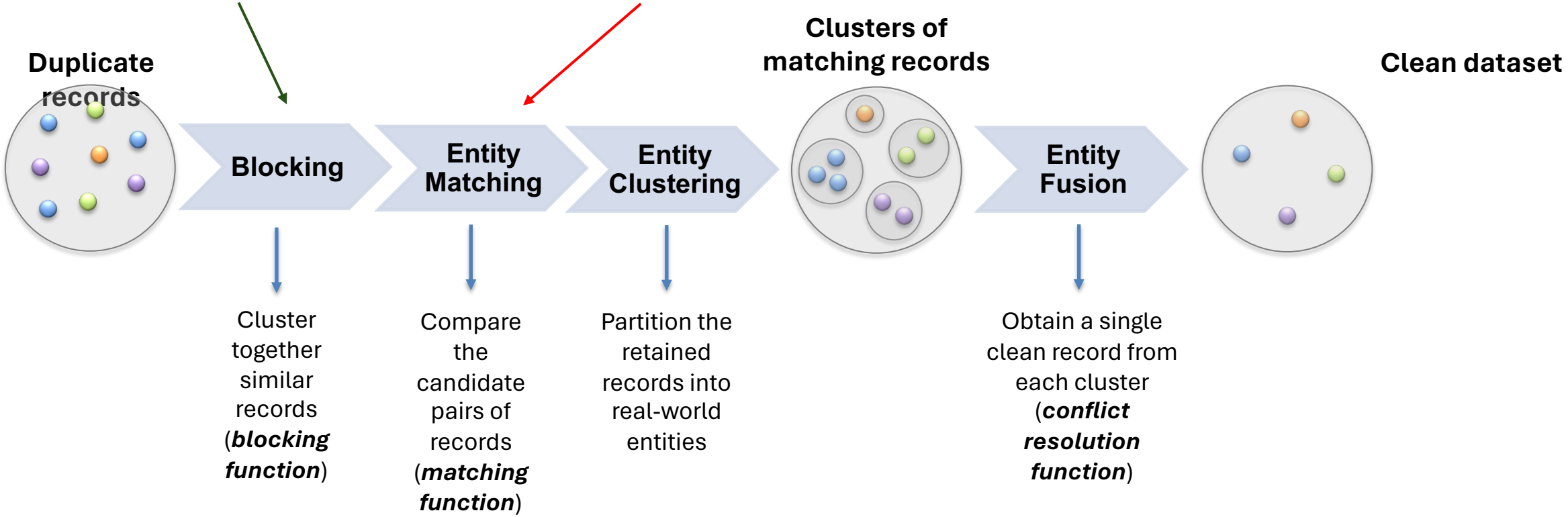
ER: the standard pipeline for big data



ER: the standard pipeline for big data

This has to be “cheap”
High recall and
very low precision is generally OK

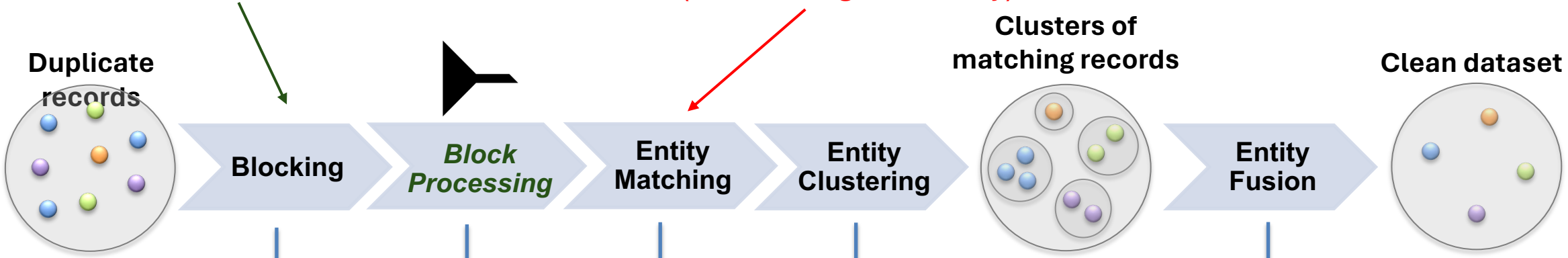
This can be the expensive part
(we need high accuracy)



ER: the standard pipeline for big data

This has to be "cheap"
High recall and
very low precision is generally OK

This can be the expensive part
(we need high accuracy)



Cluster together similar records (***blocking function***)

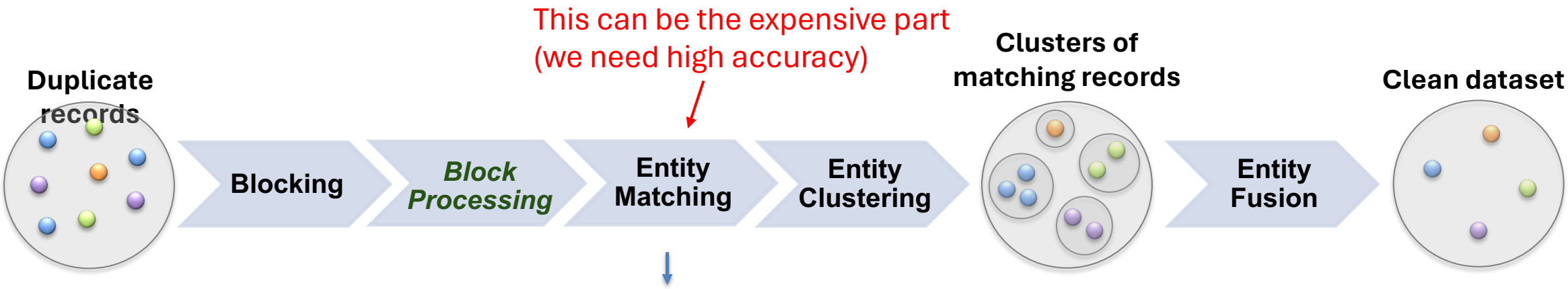
1. **Increase precision w/o affecting the recall**
2. **Redefine the priority for BDI on-demand application**

Compare the candidate pairs of records (***matching function***)

Partition the retained records into real-world entities

Obtain a single clean record from each cluster (***conflict resolution function***)

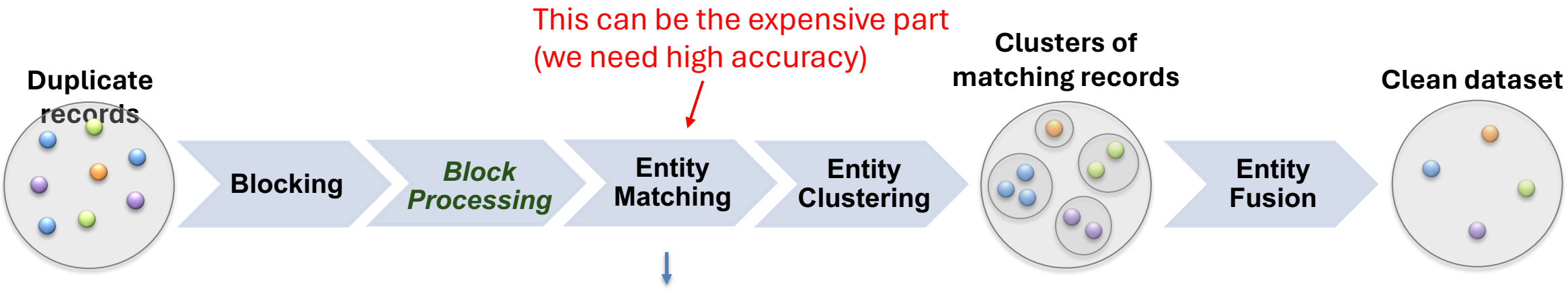
ER: the standard pipeline for big data



State-of-the-art approaches are ML-based, i.e., train a binary classifier

1. **ML-based**
   - Requires some labelled data (positive and negative matches)
   - ⛔ Each pair of records has to be represented as a set of features
2. **DL-based**
   - ⛔ Requires significant amount of labelled data
     - 🛟 Can use transfer learning
   - Avoid feature engineering

ER: the standard pipeline for big data



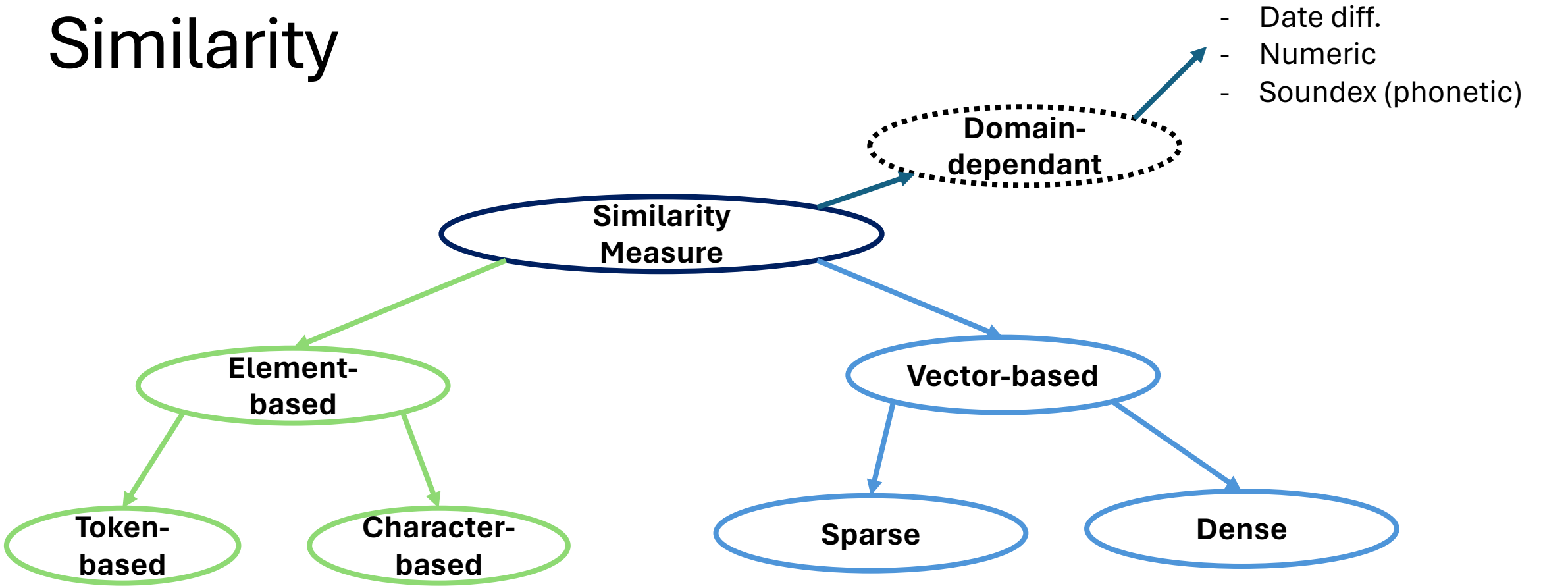
State-of-the-art approaches are ML-based, i.e., train a binary classifier

1. **ML-based**
   - Requires some labelled data (positive and negative matches)
   - ⛔ Each pair of records has to be represented as a set of features
2. **DL-based**
   - ⛔ Requires significant amount of labelled data
   - 🛟 Can use transfer learning
   - Avoid feature engineering

`Text Similarities`

Text Similarity

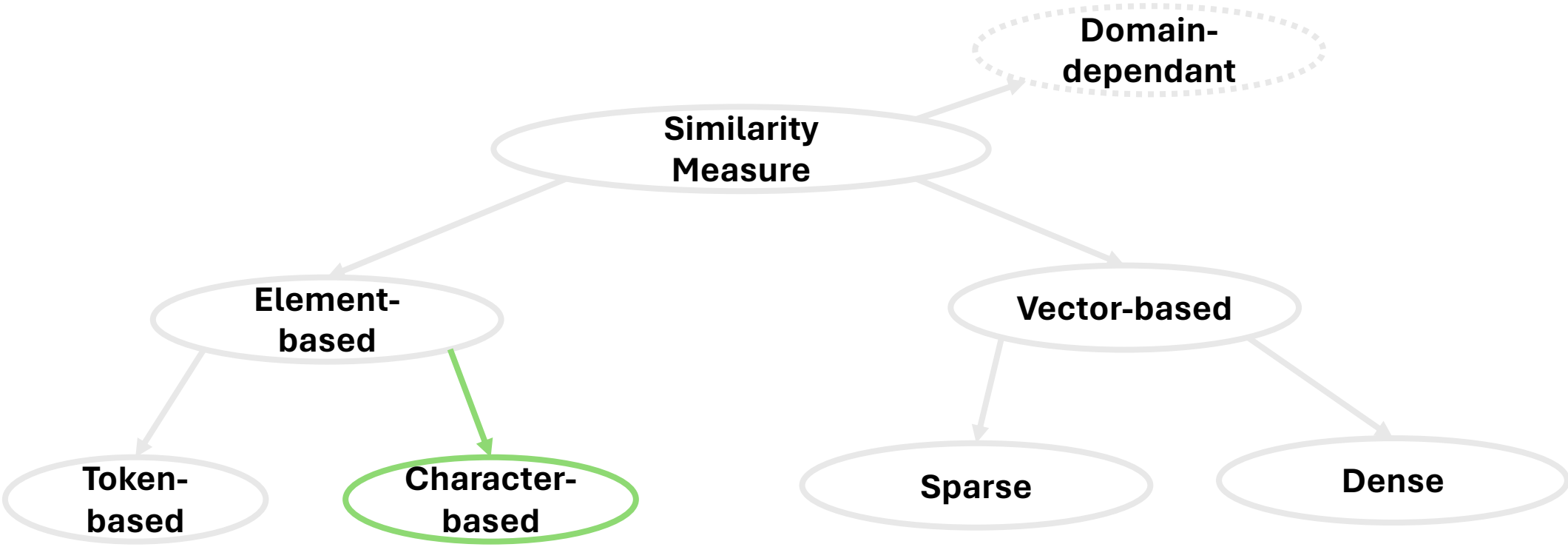
Similarity


Similarity Measure
Domain-dependant
- Date diff.
- Numeric
- Soundex (phonetic)
Element-based
Vector-based
Token-based
Character-based
Sparse
Dense
- Jaccard
- Set containment
- Overlap
- …
- Edit
- TF/IDF
- BM52
- Embeddings (e.g., BERT-based)

Similarity measure definition

- sim(x,y)
  - x and y can be strings, numbers, tuples, objects, images, etc.
- Normalized: sim(x,y) ∈ [0,1]
  - sim(x,y) = 1 for exact match
  - sim(x,y) = 0 for “completely different” x and y.
  - 0 < sim(x,y) < 1 for some approximate similarity
- Distance function / distance **metric:**
  - Reflexive: dist(x,x) = 0
  - Positive: dist(x,y) ≥ 0
  - Symmetry: dist(x,y) = dist(y,x)
  - Triangular inequality: dist(x,z) ≤ dist(x,y) + dist(y,z)
- sim(x,y) = 1-dist(x,y) if dist(x,y) ∈ [0,1]
- sim(x,y) = 1/dist(x,y) if dist(x,y) > 0

Similarity


Domain-dependant
Similarity Measure
Element-based
Vector-based
Token-based
Character-based
Sparse
Dense

Character-based

- Good choice to catch small variation and typos
  - E.g., “Aalborg” vs “Ålborg”
    - If we were using exact match, we cannot catch the match

- Generally, a good approach is to count how many “edit” are needed to transform a tring to another (i.e., **edit distance**)
  - E.g., “Aalborg”

    → remove “A” (1 edit)
    → replace “a” with “Å” (1 edit)
    → “Ålborg” (edit distance = 1+1 = 2)

  - What “edit” operation do we want to allow? (Insert, Delete, Replace)

Hamming distance

- Number of positions in which two strings (same length) differ
  - Minimum number of replacement needed to change one string into the other

- Used mostly for binary numbers and to measure communication errors
  - x, y = arrays of 1s and 0s
  - Hamming distance = x XOR y

- $\text{Hamming}_{\text{distance}}$(“Aalborg” “Ålborg_”) = 7
  - not much useful for strings

Levenshtein Distance

- Minimum number of character **insertions**, **deletions**, and **replacements** needed to transform one string **x** into another **y**
- Compute transcript based on dynamic programming algorithm
  1. Initialize matrix M of size (|x|+1) x (|y|+1)
  2. Fill matrix: $M_{i,0} = i$ and $M_{0,j} = j$
  3. Recursion: $M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } x[i] = y[i] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & \text{otherwise} \end{cases}$
  4. Distance: LevDist(a,b) = $1 - \frac{LevDist(x,y)}{\max(|x|,|y|)}$

Levenshtein Distance example

			i=1					i=6
		i=0	**L**	**E**	**V**	**E**	**N**	**S**
	j=0	0	1	2	3	4	5	6
j=1	**L**	1						
	E	2						
	I	3						
...	**V**	4						
	E	5						
	N	6						
j=7	**S**	7						

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } x[i] = y[i] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & \text{otherwise} \end{cases}$$

Levenshtein Distance example

			i=1					i=6
		i=0	**L**	**E**	**V**	**E**	**N**	**S**
	j=0	0	1	2	3	4	5	6
j=1	**L**	1						
	E	2						
	I	3						
…	**V**	4						
	E	5						
	N	6						
j=7	**S**	7						

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } x[i] = y[i] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & \text{otherwise} \end{cases}$$

Levenshtein Distance example

			i=1					i=6
		i=0	**L**	**E**	**V**	**E**	**N**	**S**
	j=0	0	1	2	3	4	5	6
j=1	**L**	1	0					
	E	2						
	I	3						
…	**V**	4						
	E	5						
	N	6						
j=7	**S**	7						

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } x[i] = y[i] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & \text{otherwise} \end{cases}$$

Levenshtein Distance example

			i=1					i=6
		i=0	**L**	**E**	**V**	**E**	**N**	**S**
	j=0	0	1	2	3	4	5	6
j=1	**L**	1	0					
	E	2						
	I	3						
…	**V**	4						
	E	5						
	N	6						
j=7	**S**	7						

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } x[i] = y[i] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & \text{otherwise} \end{cases}$$

Levenshtein Distance example

			i=1					i=6
		i=0	**L**	**E**	**V**	**E**	**N**	**S**
	j=0	0	1	2	3	4	5	6
j=1	**L**	1	0	1				
	E	2	1	0				
	I	3						
…	**V**	4						
	E	5						
	N	6						
j=7	**S**	7						

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } x[i] = y[i] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & \text{otherwise} \end{cases}$$

Levenshtein Distance example

			i=1					i=6
		i=0	**L**	**E**	**V**	**E**	**N**	**S**
	j=0	0	1	2	3	4	5	6
j=1	**L**	1	0	1				
	E	2	1	0				
	I	3			1			
…	**V**	4						
	E	5						
	N	6						
j=7	**S**	7						

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } x[i] = y[i] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & \text{otherwise} \end{cases}$$

Levenshtein Distance example

			i=1					i=6
		i=0	**L**	**E**	**V**	**E**	**N**	**S**
	j=0	0	1	2	3	4	5	6
j=1	**L**	1	0	1	2	3	4	5
	E	2	1	0	2	3	4	4
	I	3	2	1	1	2	3	4
...	**V**	4	3	2	1	2	3	4
	E	5	4	3	2	1	2	3
	N	6	5	4	3	2	1	2
j=7	**S**	7	6	5	4	3	2	1

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } x[i] = y[i] \\ 1 + \min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1}) & \text{otherwise} \end{cases}$$

Try online: https://phiresky.github.io/levenshtein-demo/

Considereations on Levenshtein Distance

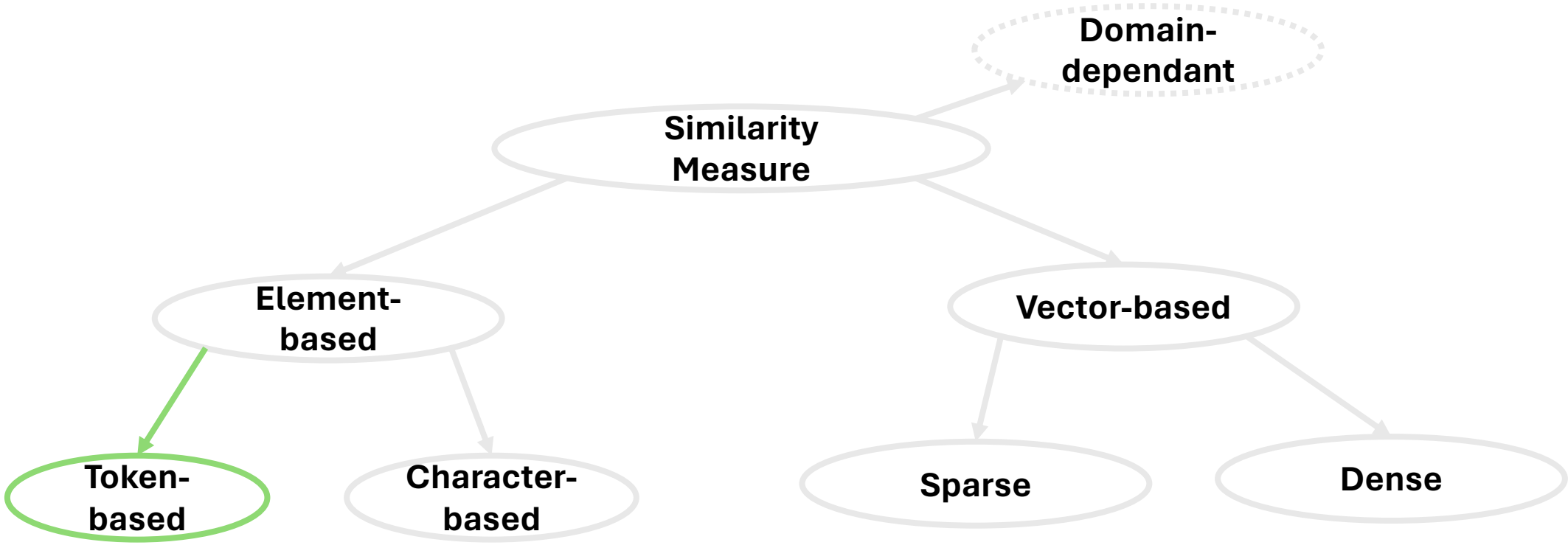
- Complexity
  - Time: $O(|x| \cdot |y|)$ → to fill in matrix
  - Space: $O(\min(x, y))$ → current i-row/j-col and the next one

- Some properties
  - $0 \leq LeveDist(x, y) \leq \max(|x|, |y|)$
  - $\text{abs}(|x| - |y|) \leq LevenshteinDist(x, y)$
    - better to compare only strings with similar length

Other Edit Distances

- Damerau–Levenshtein distance
  - Similar to Levenshtein distance, but considers transposed characters
- Winkler similarity
  - Similarity of first few letters is most important
- Character (phonetic) encoding
  - Use some encoding for some character or group of character
    - Good for some domain and specific language
    - E.g., “ou” and “u” might be replaced with the same encoding in French
- Edit distance is good for “short” string, e.g., names, titles, places, etc.
  - Computationally expensive and ineffective with long snippets, e.g., tweets, reviews, etc.

Similarity


Domain-dependant
Similarity Measure
Element-based
Vector-based
Token-based
Character-based
Sparse
Dense

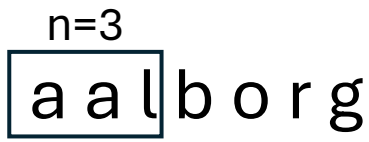
Tokenization

- **Idea**: split string into tokens according to some separator
  - Space, hyphen, punctuation, special characters, etc.
  - Usually, also convert to lower-case

- **Problems**
  - Some punctuation, such as hyphens and apostrophes
    - E.g., Both hyphenated and non-hyphenated forms of many words are common
      - Sometimes hyphen is not needed
        - E.g., new-york, mac-book, etc.
      - Sometimes hyphens should be considered part of the word
        - e-mail, token-based, t-mobile, etc.
  - Numbers can be important
    - iPhone 15, Python 3.7
  - Periods can occur in abbreviations, URLs, not only in ends of sentences
    - E.g., F.B.I., Ph.D., youtube.come, etc.

n-grams tokenization

- Instead of splitting the worlds in a sentence, we employ a “sliding window” to create equally-seized tokens (i.e., n-grams)



“aal”

n-grams tokenization

- Instead of splitting the worlds in a sentence, we employ a “sliding window” to create equally-seized tokens (i.e., n-grams)

n=3

a a l b o r g

“aal”, “alb”,

n-grams tokenization

- Instead of splitting the worlds in a sentence, we employ a “sliding window” to create equally-seized tokens (i.e., n-grams)

n=3

a a l b o r g

“aal”, “alb”, “lbo”

n-grams tokenization

- Instead of splitting the worlds in a sentence, we employ a “sliding window” to create equally-seized tokens (i.e., n-grams)

n=3

a a l b o r g

“aal”, “alb”, “lbo”, “bor”,

n-grams tokenization

- Instead of splitting the worlds in a sentence, we employ a “sliding window” to create equally-seized tokens (i.e., n-grams)

n=3

a a l b o r g

“aal”, “alb”, “lbo”, “bor”, “org”

n-grams tokenization

- Instead of splitting the worlds in a sentence, we employ a “sliding window” to create equally-seized tokens (i.e., n-grams)

n=3

a a l b o r g

“aal”, “alb”, “lbo”, “bor”, “org”

- **Pro**: help to catch similarity between words
  - E.g, “aalbord” and “ålborg”
  - We’ll see als character-based similarities for that
- **Cons**: the downstream computation can be more expensive

Token-based Similarity Measures

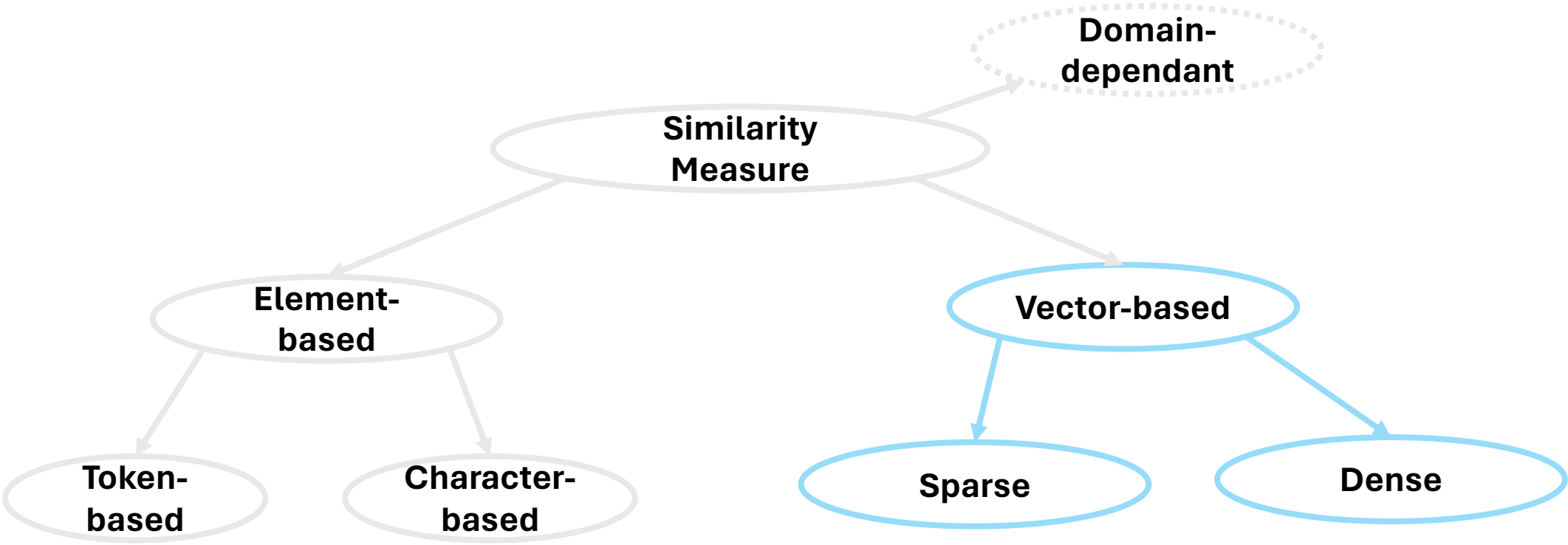
- Let's A and B be two set of tokens derived from two strings
- Jaccard Similarity $\text{JS}(A, B) = \frac{|A \cap B|}{|A| \cup |B|}$
- Jaccard Containment (aka Overalp Coefficient)

$$\text{JC}(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)}$$

- How to choose?
  - JS usually better
  - JC employed when containment is important
    - E.g., Join discovery or copy detection

- Switch to bag semantic if repetitions count for our application

Similarity

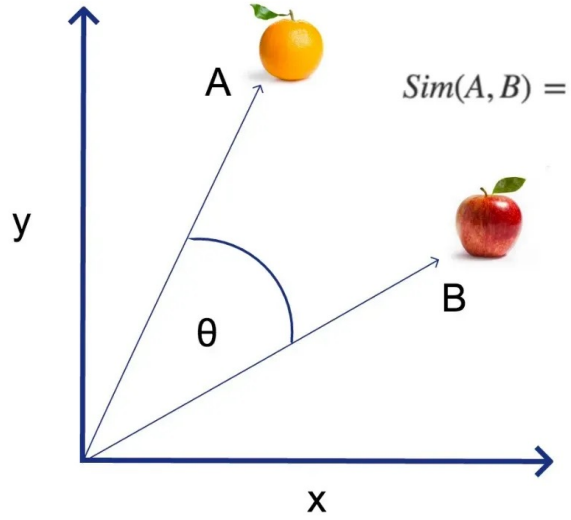

Domain-dependant
Similarity Measure
Element-based
Vector-based
Token-based
Character-based
Sparse
Dense

How to build the vectors

- **Sparse vectors**
  - “old-school”, before the deep learning wave
  - still effective solution for domain-specific data (no existing model for that)
  - very efficient in many scenario
  - Most popular: TF-IDF
  - Why sparse?
    - Select a set of tokens and that is the vector space
    - Each element will have only a portion of that set, hence its vector will be sparse
      - Each dimension is a token

- **Dense vectors**
  - Fist popular model: Word2Vec
  - Transpose a string or a sentence in a dense vector space (100s dimensions)
    - Each element has a value for each dimension
      - Each dimension is not a token

Vector Space Model



$$Sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

- Once we have vectors, we can serch them

- Two element are similar if they are “close” in the vector space
  - Coloseness is typically measured by measuring the cosine of the angle between the two vector A and B
  - Derived from the Eucledian dot product:
    $A \cdot B = ||A|| \, ||B|| \cos \theta$
    - Where ||A|| is the norm of A, and θ is the angle between A and B

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

TF-IDF

- As the name says:
  - Tearm frequency is number of occurrence of a *token i* within a document *j*
    - $TF_{i,j} = \frac{n_{ij}}{|Docj|}$
  - Inverse document frequency is how infrequnt is a *token j* in the *corpus D*
    - $IDF_i = log_{10} \frac{|D|}{|\{d : i \in d\}|}$
  - For a *token i* in a *document j*:
    $tf_{ij} = TF_{i,j} * IDF_i$

- In practice:
  - select the least *N* frequent tokens (e.g.,1000)
  - build vector for each *document* considering its *tokens*

- Limitations:
  1. Frequency of a token increase in a document → final score increase linearly → use BM25
  2. If we add document to the corpus, the measure might become inaccurate → use language models

Entity Matching

Text similarity and Entity Matching

id	brand	model	type	mp	price
R1	canon	eos 400d	dslr	10.1	165.00
R2	canon	rebel xti	reflex	1.01	185.00
R3	eos canon	400 d	dslr	10.0	115.00
R4	nikon	d-200	dslr	-	150.00
R5	nikon	d200	-	10.2	130.00
R6	nikon	d40	digital	-	100.00
R7	kodak	dc3200	dslr	1.3	75.00
R8	kodak	dc-3200	-	1.3	80.00

1. Candidate Pair

eos canon	400 d	dslr	10.1	115.00

canon	eos 400d	dslr	10.1	165.00

2. Feature Extraction

brand_JS _3grams	brand_JS _4grams	model _tf-idf	type_ tf-idf	...	mp_lev _sim	label
0.7	0.65	0.9	1.0		0.75	1

3. Model Training

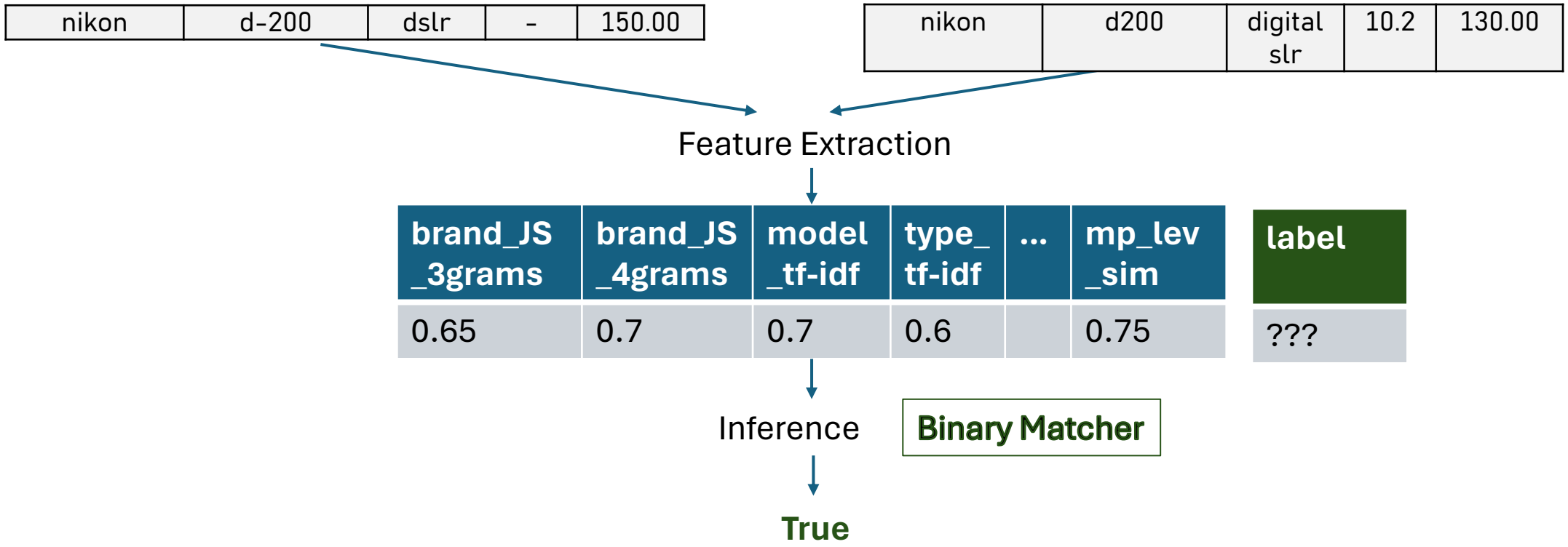
- Random Forest
- SVM
- ...

Binary Matcher

4. Inference on new pairs

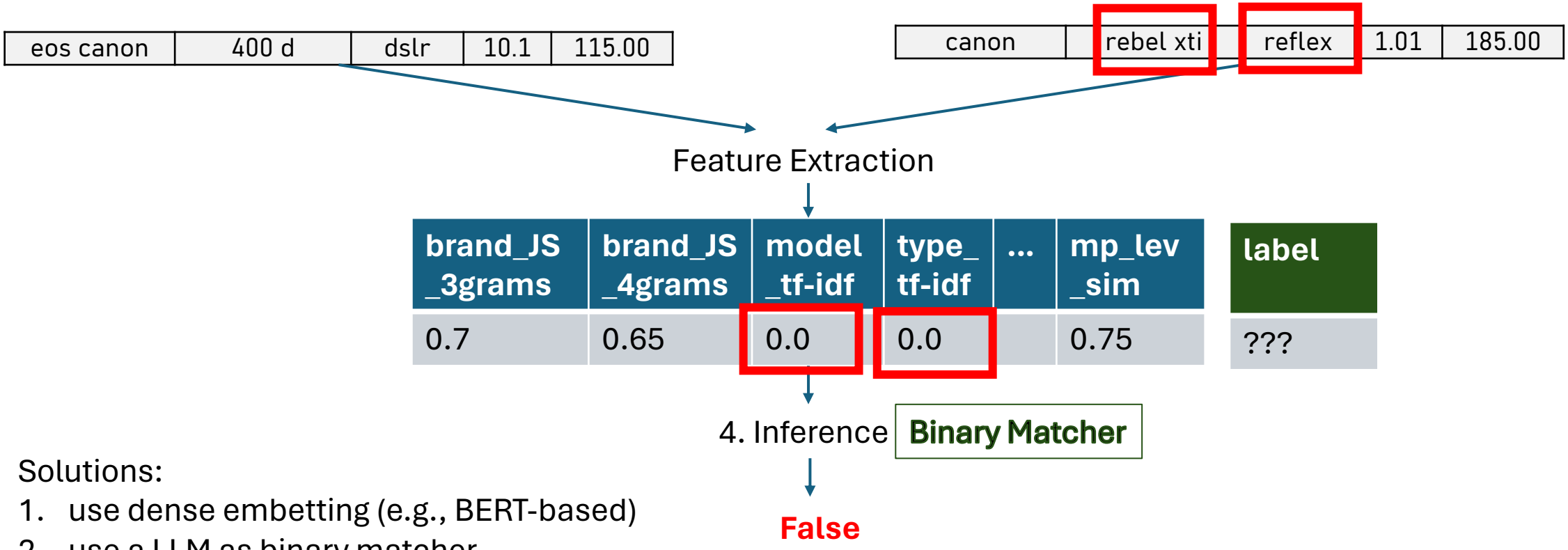
Text similarity and Entity Matching

id	brand	model	type	mp	price
R1	canon	eos 400d	dslr	10.1	165.00
R2	canon	rebel xti	reflex	1.01	185.00
R3	eos canon	400 d	dslr	10.0	115.00
R4	nikon	d-200	dslr	-	150.00
R5	nikon	d200	-	10.2	130.00
R6	nikon	d40	digital	-	100.00
R7	kodak	dc3200	dslr	1.3	75.00
R8	kodak	dc-3200	-	1.3	80.00



Text similarity and Entity Matching

id	brand	model	type	mp	price
R1	canon	eos 400d	dslr	10.1	165.00
R2	canon	rebel xti	reflex	1.01	185.00
R3	eos canon	400 d	dslr	10.0	115.00
R4	nikon	d-200	dslr	-	150.00
R5	nikon	d200	-	10.2	130.00
R6	nikon	d40	digital	-	100.00
R7	kodak	dc3200	dslr	1.3	75.00
R8	kodak	dc-3200	-	1.3	80.00



Solutions:

1. use dense embetting (e.g., BERT-based)
2. use a LLM as binary matcher

Bibliography

1. Dong XL, Srivastava D. Big data integration. In2013 IEEE 29th international conference on data engineering (ICDE) 2013 Apr 8 (pp. 1245-1248). IEEE.
2. Bleiholder J, Naumann F. Data fusion. ACM computing surveys (CSUR). 2009 Jan 15;41(1):1-41.
3. Christophides V, Efthymiou V, Palpanas T, Papadakis G, Stefanidis K. End-to-end entity resolution for big data: A survey. arXiv preprint arXiv:1905.06397. 2019 May 15.
4. Rahm E, Bernstein PA. A survey of approaches to automatic schema matching. the VLDB Journal. 2001 Dec;10:334-50.
5. Doan A, Halevy A, Ives Z. Principles of data integration. Elsevier; 2012 Jun 25.
6. Doan A, Domingos P, Halevy AY. Reconciling schemas of disparate data sources: A machine-learning approach. InProceedings of the 2001 ACM SIGMOD international conference on Management of data 2001 May 1 (pp. 509-520).
7. https://www.biggorilla.org/software_cat/schema-matching-and-mapping/index.html
8. Nargesian F, Zhu E, Miller RJ, Pu KQ, Arocena PC. Data lake management: challenges and opportunities. Proceedings of the VLDB Endowment. 2019 Aug 1;12(12):1986-9.
9. Stonebraker M, Ilyas IF. Data Integration: The Current Status and the Way Forward. IEEE Data Eng. Bull.. 2018 Jun;41(2):3-9.
10. Deng D, Fernandez RC, Abedjan Z, Wang S, Stonebraker M, Elmagarmid AK, Ilyas IF, Madden S, Ouzzani M, Tang N. The Data Civilizer System. InCidr 2017 Jan 8.
11. Chu X, Ilyas IF, Krishnan S, Wang J. Data cleaning: Overview and emerging challenges. InProceedings of the 2016 international conference on management of data 2016 Jun 26 (pp. 2201-2206).
12. Gomaa WH, Fahmy AA. A survey of text similarity approaches. international journal of Computer Applications. 2013 Apr 13;68(13):13-8.