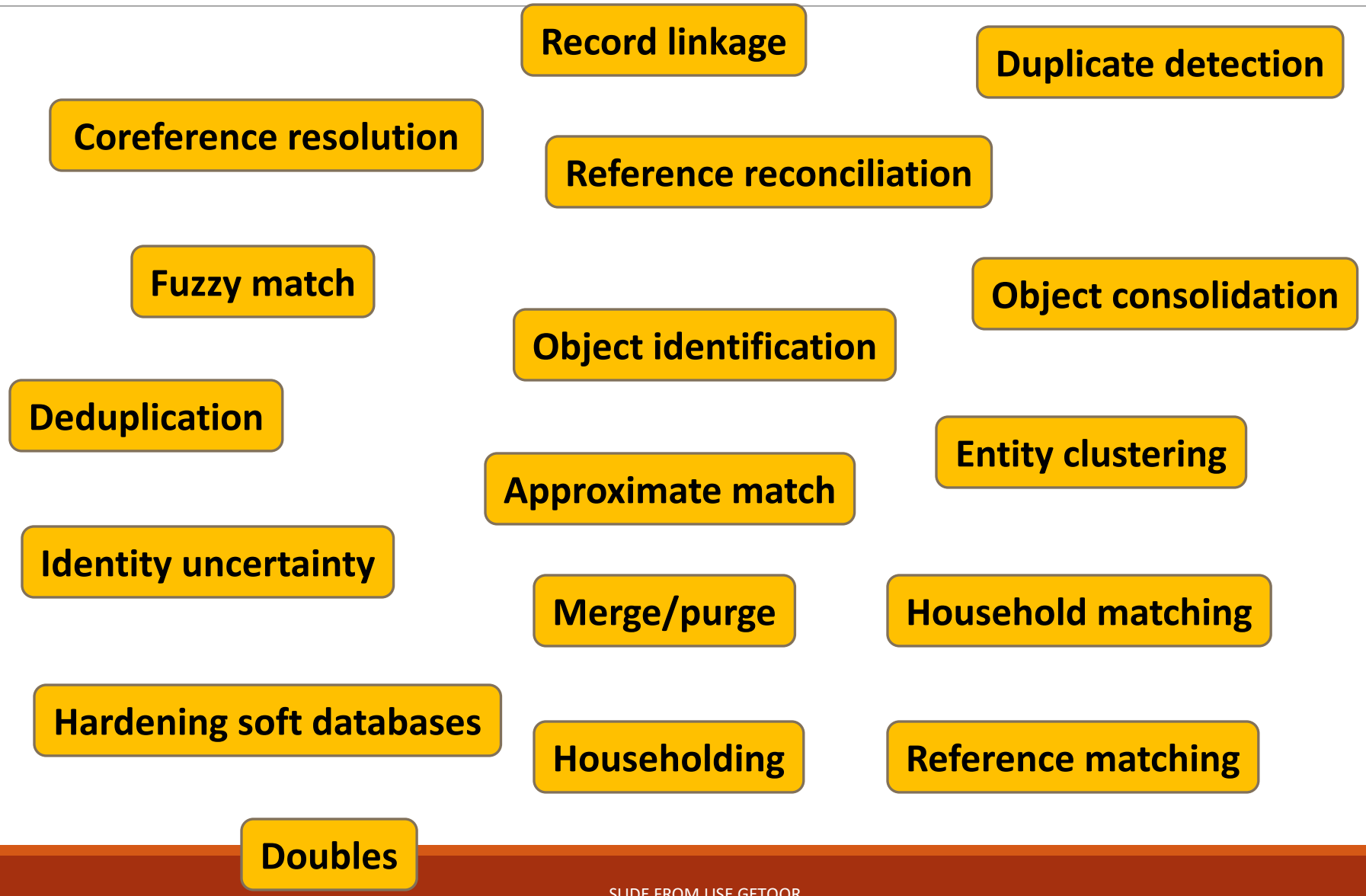
Entity Resolution

JAY PUJARA

DSCI-558 SPRING 2021

SLIDES FROM HTTPS://KGTUTORIAL.GITHUB.IO/

Ironically, Entity Resolution has many duplicate names



Entity Resolution & Linking

…during the late 60′s and early 70′s, **Kevin Smith** worked with several local…

…the term hip-hop is attributed to **Lovebug Starski**. What does it actually mean…

Like Back in 2008, the Lions drafted **Kevin Smith**, even though Smith was badly…

… backfield in the wake of **Kevin Smith**′s knee injury, and the addition of Haynesworth…

The filmmaker **Kevin Smith** returns to the role of Silent Bob…

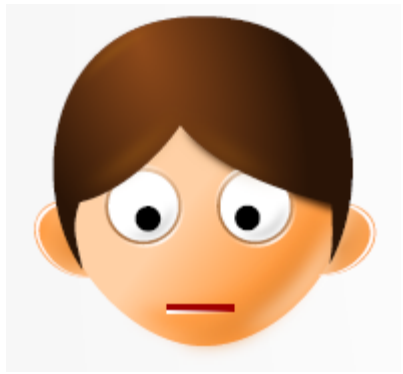
Nothing could be more irrelevant to **Kevin Smith**′s audacious ′′Dogma′′ than ticking off…

… The Physiological Basis of Politics,” by **Kevin Smith**, Douglas Oxley, Matthew Hibbing…

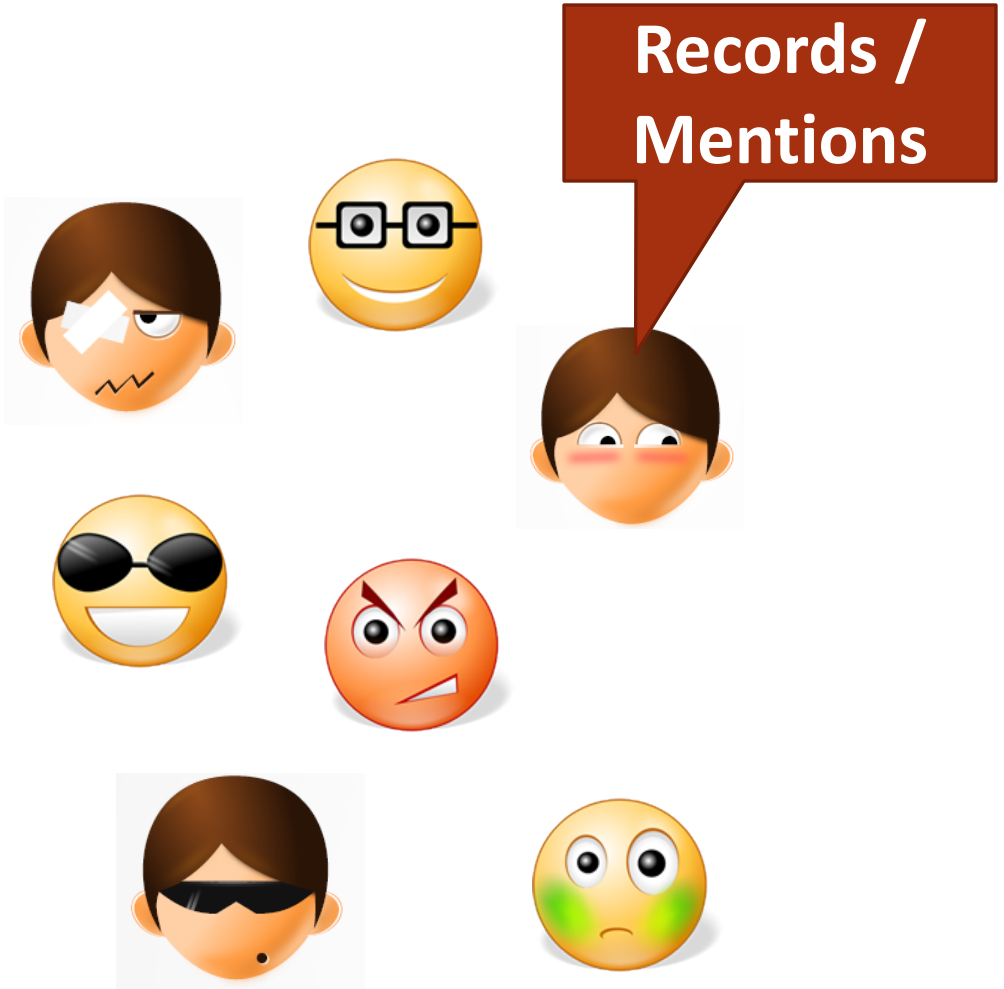
When do we need entity resolution?

Abstract Problem Statement

Real World

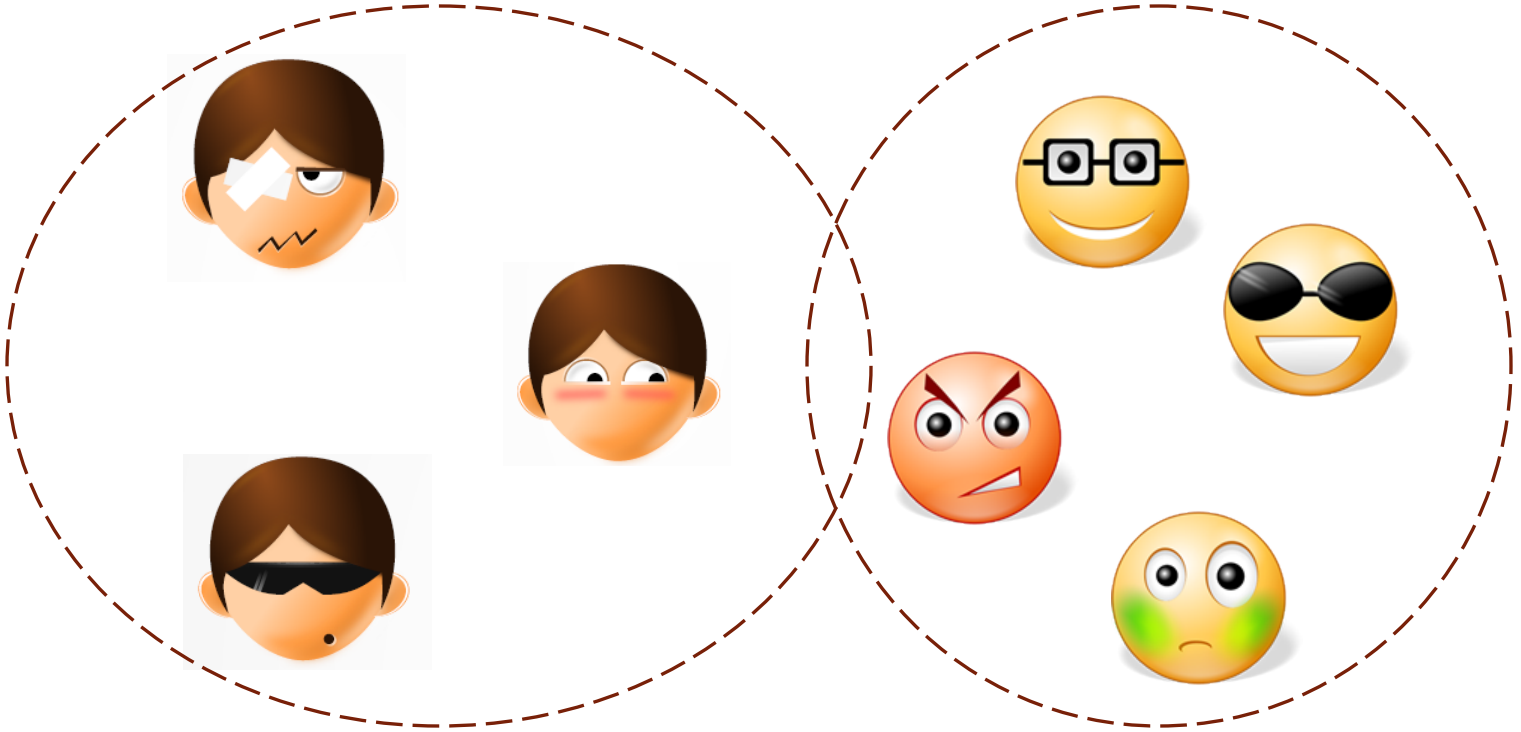


Digital World

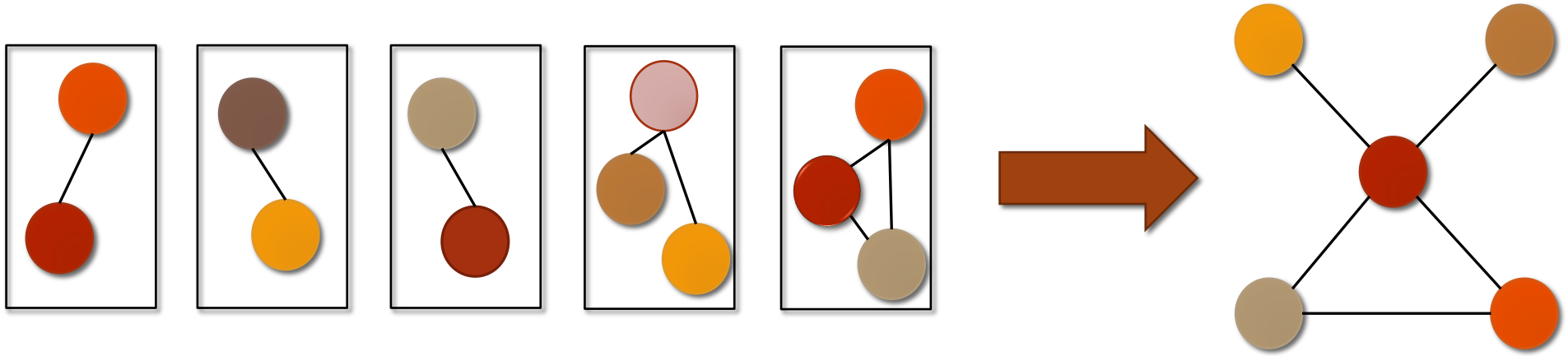


Deduplication Problem Statement

- Cluster the records/mentions that correspond to same entity

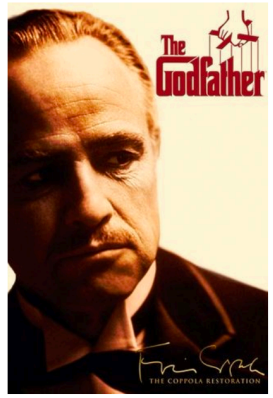


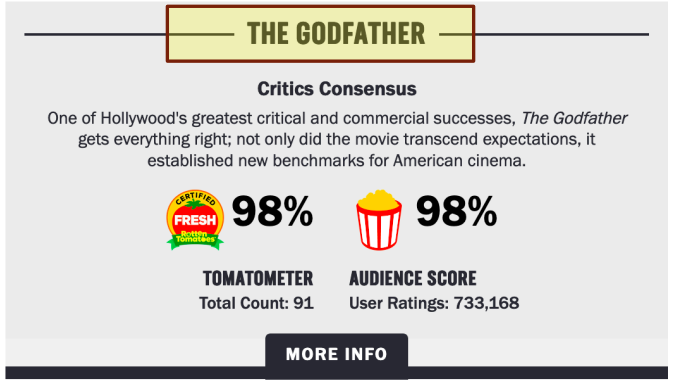
Deduplication

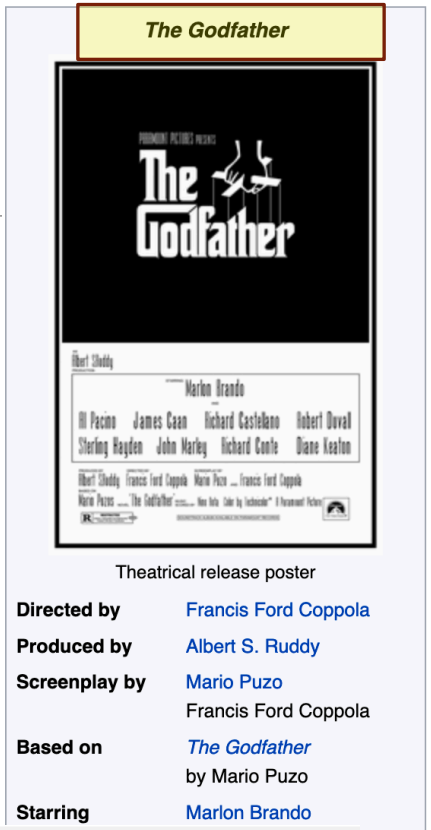


Merging Ambiguous Entities

Deduplication Example







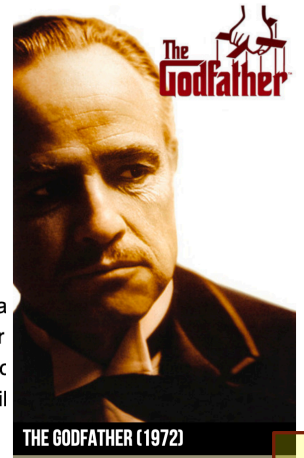
Movie Details & Credits

Paramount Pictures | Release Date: March 11, 1972 | R

Starring: Al Pacino, Marlon Brando

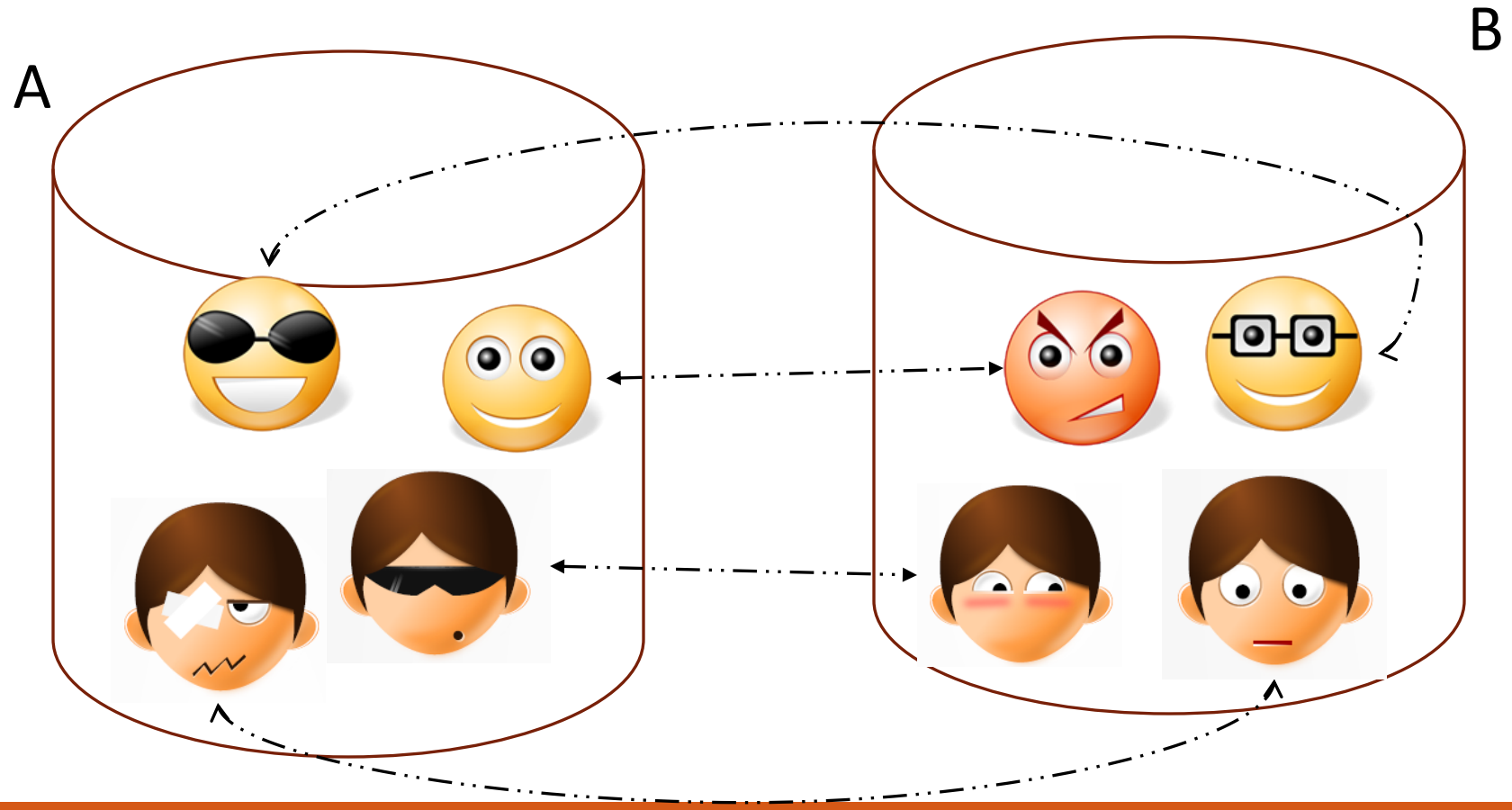
Summary: Francis Ford Coppola's epic features Marlon Bra winning role as the patriarch of the Corleone family. Director chilling portrait of the Sicilian clan's rise and near fall from pc masterfully balancing the story between the Corleone's famil business in which they are… **Expand**

Director: Francis Ford Coppola
Genre(s): Drama, Thriller, Crime
Rating: R
Runtime: 175 min

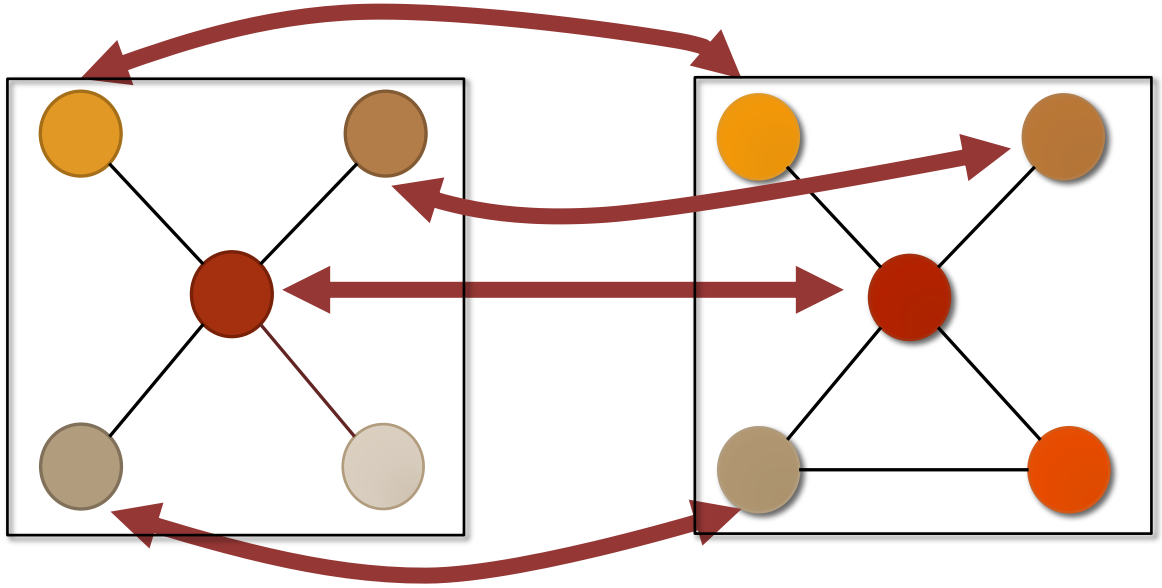


Record Linkage Problem Statement

- Link records that match across databases



Record Linkage



Combining KGs

Record Linkage Example

FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE

The Godfather (1972)
9.2/10 1,505,041
R | 2h 55min | Crime, Drama | 24 March 1972 (USA)

FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE

The Godfather: Part II (1974)
9.0/10 1,054,002
R | 3h 22min | Crime, Drama | 20 December 1974 (USA)

FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE

Pulp Fiction (1994)
8.9/10 1,716,255
R | 2h 34min | Crime, Drama | 14 October 1994 (USA)

FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE

The Silence of the Lambs (1991)
8.6/10 1,184,877
R | 1h 58min | Crime, Drama, Thriller | 14 February 1991 (USA)

THE GODFATHER

Critics Consensus

One of Hollywood's greatest critical and commercial successes, *The Godfather* gets everything right; not only did the movie transcend expectations, it established new benchmarks for American cinema.

CERTIFIED FRESH 98% 98%

THE GODFATHER, PART II

Critics Consensus

Drawing on strong performances by Al Pacino and Robert De Niro, Francis Ford Coppola's continuation of Mario Puzo's Mafia saga set new standards for sequels that have yet to be matched or broken.

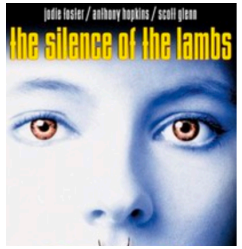
CERTIFIED FRESH 97% 97%

PULP FICTION

Critics Consensus

One of the most influential films of the 1990s, *Pulp Fiction* is a delirious post-modern mix of neo-noir thrills, pitch-black humor, and pop-culture touchstones.

CERTIFIED FRESH 92% 96%



THE SILENCE OF THE LAMBS

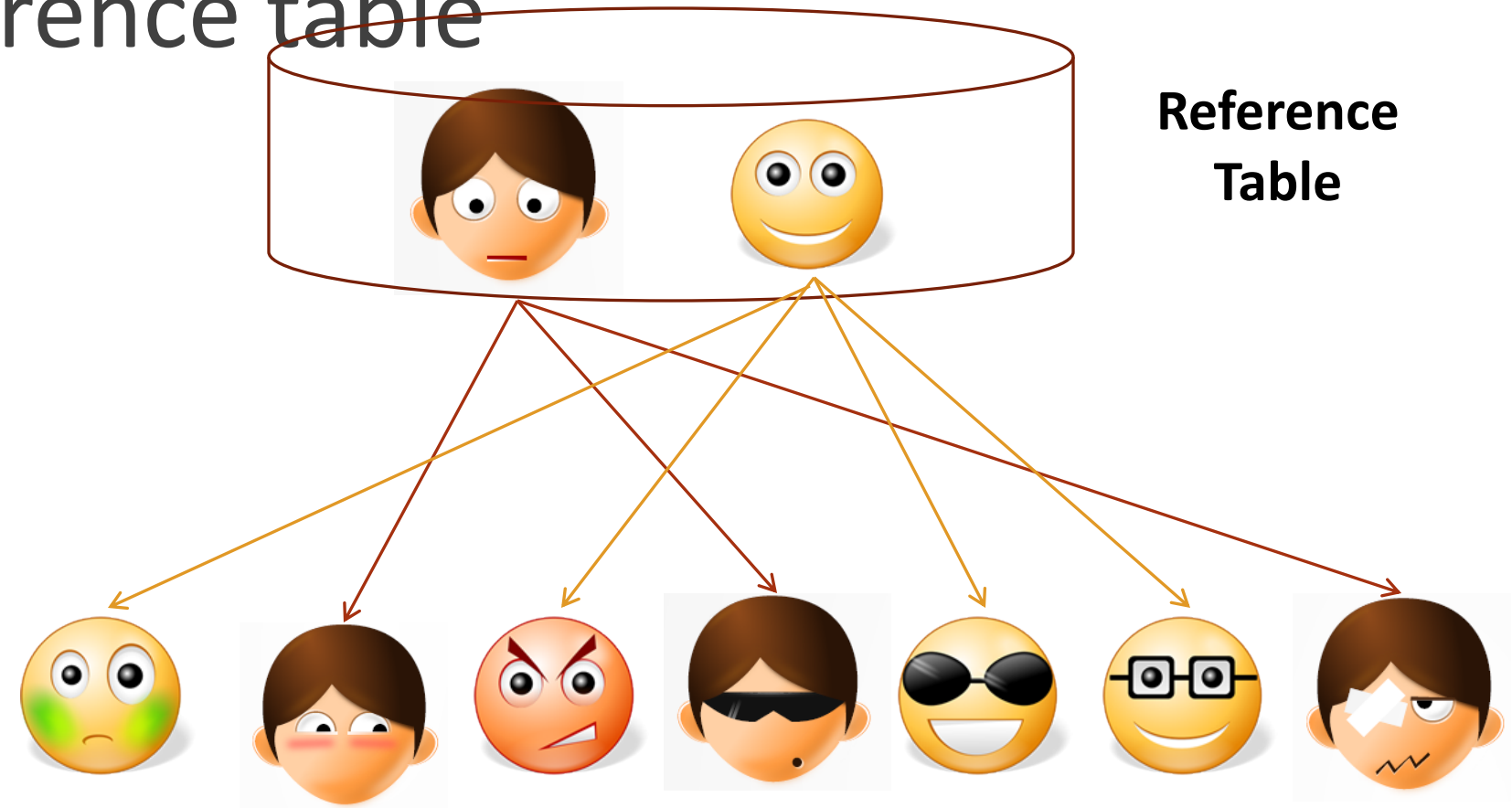
Critics Consensus

Director Jonathan Demme's smart, taut thriller teeters on the edge between psychological study and all-out horror, and benefits greatly from stellar performances by Anthony Hopkins and Jodie Foster.

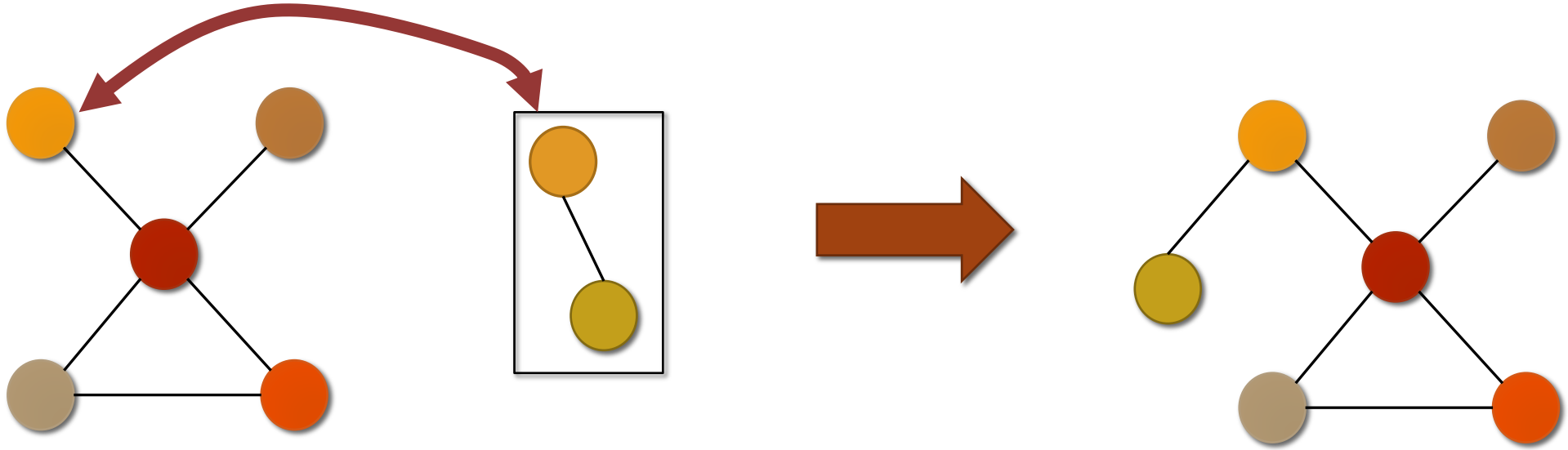
CERTIFIED FRESH 96% 95%

Entity Linking Problem

- Match noisy records to clean records in a reference table



Entity Mapping/Linking



Integrating New Candidates

Entity Linking Example

TIL that in the movie The Godfather, the reason the word 'mafia' is not mentioned a single time is because mafia boss Joe Colombo, along with Frank Sinatra, threatened the films production and would only back the filming if they could change the script to their liking.

Untangling terminology

Coreference

Unifying mentions of the same entity in a single document

Entity Linking

Mapping a mentions to the canonical entity in a KG

De-duplication

Unifying mentions from many sources into a single view

Record Linkage

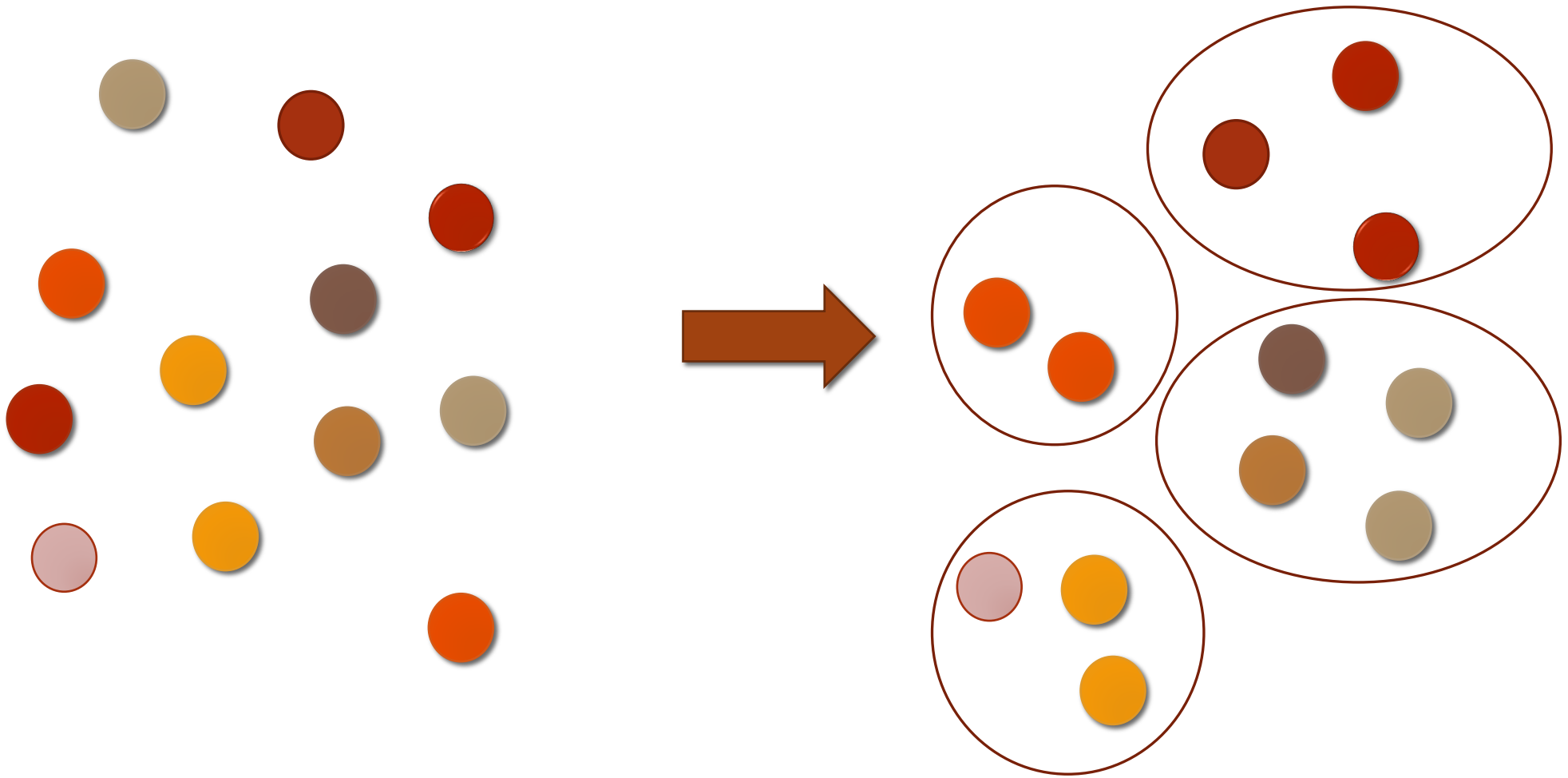
Matching entities across two different KGs

ER Motivating Examples

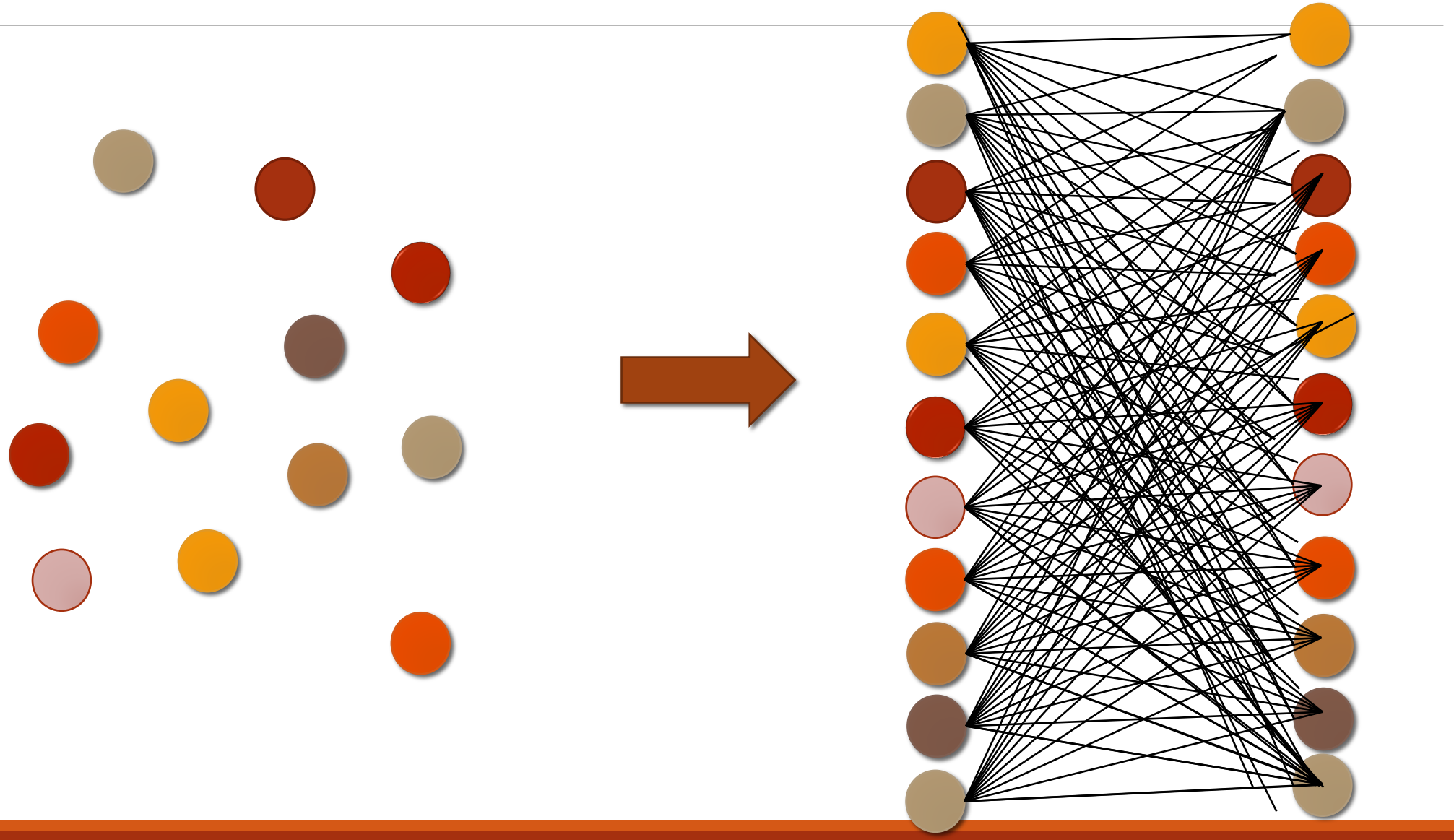
- *Linking Census Records*
- *Public Health*
- *Medical records*
- *Web search*
- *Comparison shopping*
- *Maintaining customer databases*
- *Law enforcement and Counter-terrorism*
- *Scientific data*
- *Genealogical data*
- *Bibliographic data*

How do we perform entity resolution?

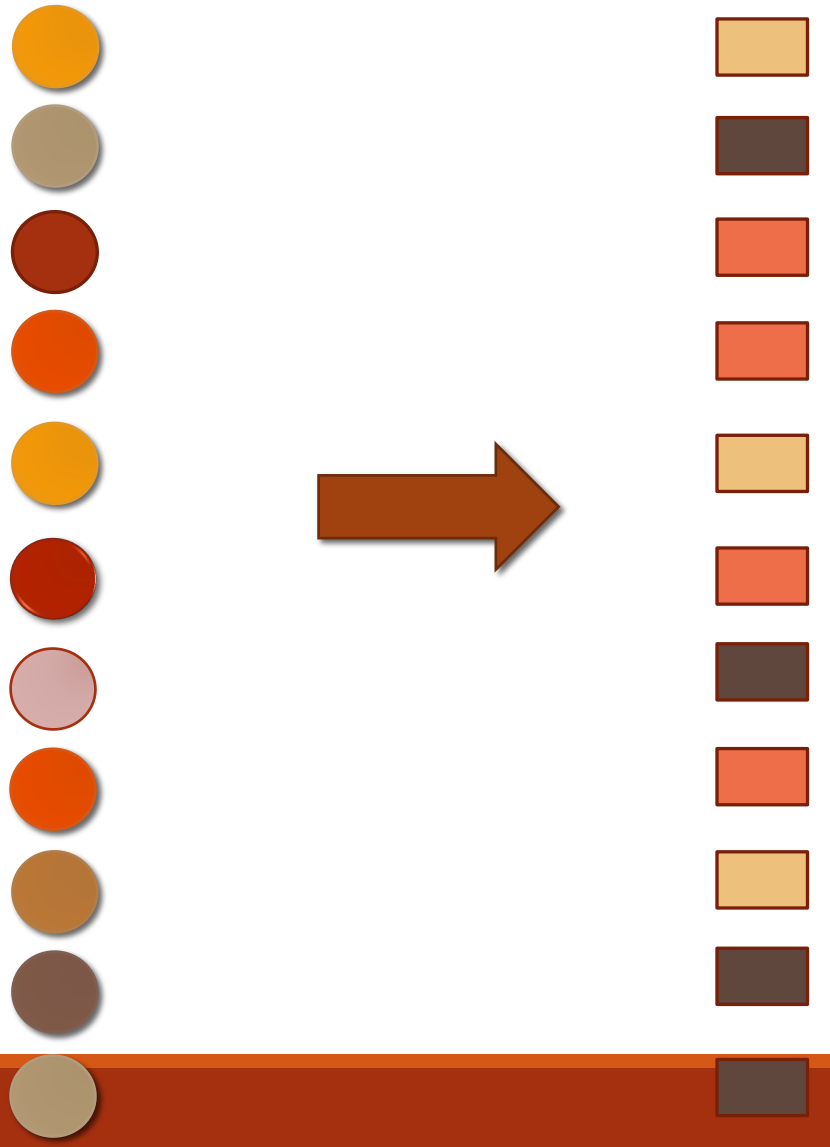
Clustering



Pairwise Prediction

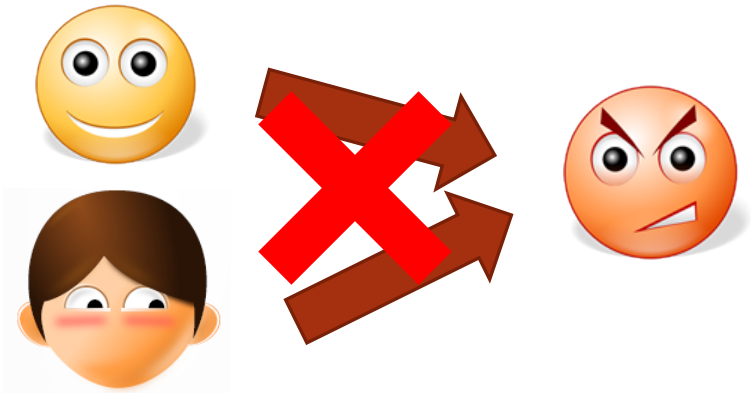


Classification

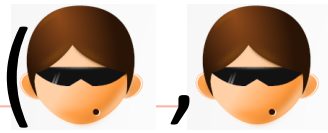


Typical Assumptions Made

- *Each record/mention is associated with a single real world entity.*



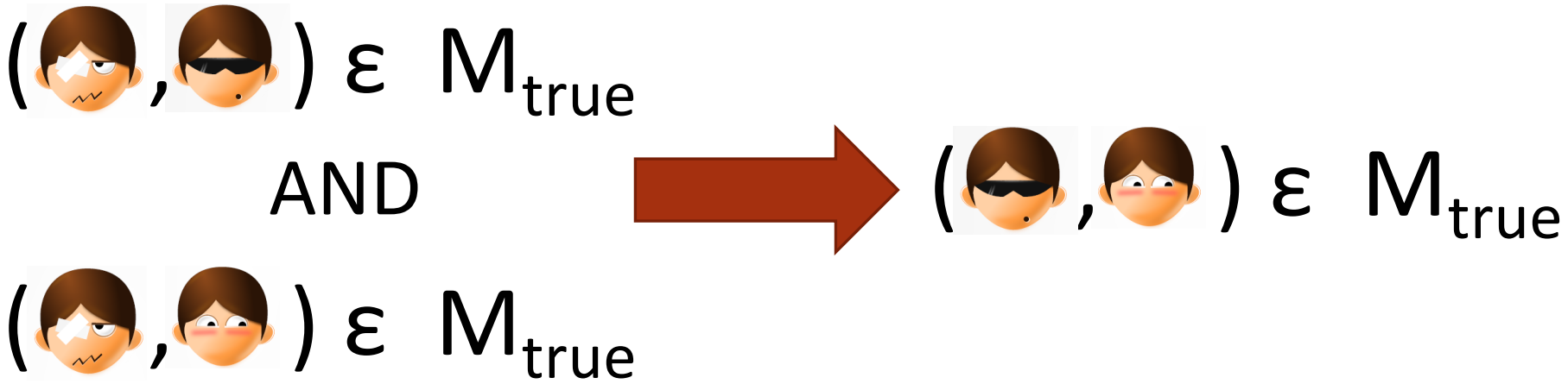
- *If two records/mentions are identical, then they are true matches*

 = M_{true}

ER versus Classification

Finding matches vs non-matches is a classification problem

- Imbalanced: typically O(R) matches, $O(R^2)$ non-matches
- Instances are pairs of records. Pairs are not IID



ER vs (Multi-relational) Clustering

Computing entities from records is a clustering problem

- In typical clustering algorithms (k-means, LDA, etc.) *number of clusters is a constant or sublinear in R.*

- In ER: *number of clusters is linear in R, and average cluster size is a constant. Significant fraction of clusters are singletons.*

Constraints

- Important forms of constraints:
  - **Exclusivity**: If M1 matches with M2, then M3 cannot match with M2
  - **Transitivity**: If M1 and M2 match, M2 and M3 match, then M1 and M3 match
  - **Functional Dependency**: If M1 and M2 match, then M3 and M4 must match

- Exclusivity is key to **record linkage**
- Transitivity is key to **deduplication**
- Functional dependencies used in **collective ER**

Canonical model for entity resolution

- Blocking:
  - Assign entities to “blocks” where matches are more likely
  - Can use clustering or classification approaches
- Matching
  - Use pairwise predictor to resolve co-referent mentions
- Validation
  - Apply constraints like 1-1 matching, identify canonical form

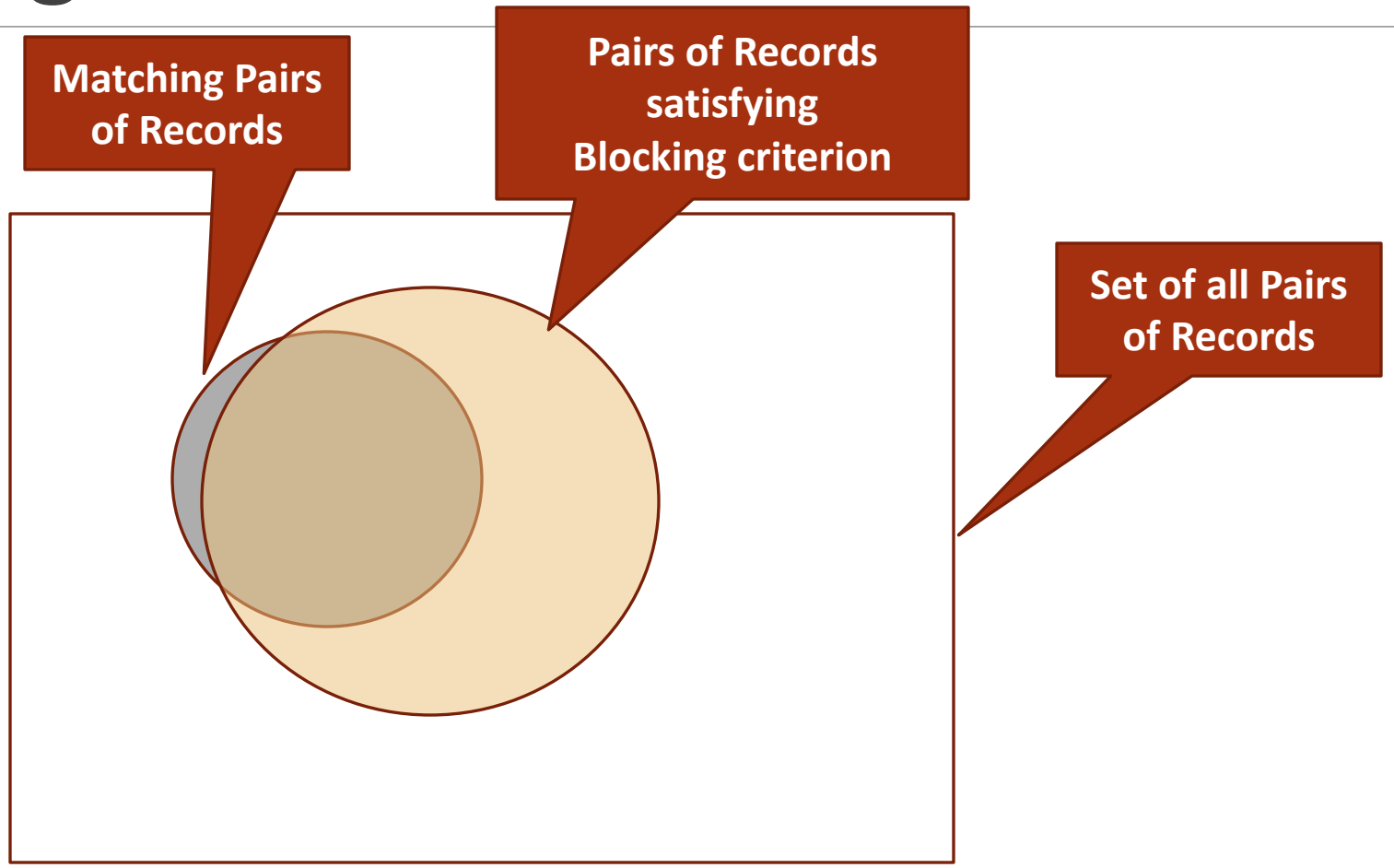
Blocking

Blocking: Motivation

- Naïve pairwise: $|R|^2$ pairwise comparisons
  - 1000 business listings each from 1,000 different cities across the world
  - 1 trillion comparisons
  - 11.6 days (if each comparison is 1 µs)

- Mentions from different cities are unlikely to be matches
  - **Blocking Criterion: City**
  - 1 billion comparisons
  - 16 minutes (if each comparison is 1 µs)

Blocking: Motivation



Blocking: Problem Statement

Input: Set of records *R*

Output: Set of *blocks/canopies*

$$\{C_1, C_2, \ldots, C_k\}, where\ \forall_i C_i \subset R\ and \bigcup_i C_i = R$$

Variants:

- *Disjoint Blocking*: Each mention appears in one block.

$$\forall_{i,j} C_i \cap C_j = \emptyset$$

- *Non-disjoint Blocking*: Mentions can appear in more than one block.

How do we block?

- Feature-based blocking keys
- Clustering
- Hashing

Blocking: Problem Statement

Metrics:

- Efficiency (or reduction ratio) : $$\frac{number\ of\ pairs\ compared}{total\ number\ of\ pairs\ in\ R \times R}$$
- Recall* (or pairs completeness) : $$\frac{number\ of\ true\ matches\ compared}{number\ of\ true\ matches\ in\ R \times R}$$
- Precision* (or pairs quality) : $$\frac{number\ of\ true\ matches\ compared}{number\ of\ matches\ compared}$$
- Max Canopy Size: $$max_i\ |C_i|$$

**Need to know ground truth in order to compute this metric*

Simple Blocking: Inverted Index on a Predicate

Examples of blocking predicates (keys):

- First three characters of last name
- City + State + Zip
- Character or Token n-grams
- Minimum infrequent n-grams

Learning Optimal Blocking Functions

- Using one or more blocking predicates may be insufficient
  - 2,376,206 American's shared the surname Smith in the 2000 US
  - NULL values may create large blocks.

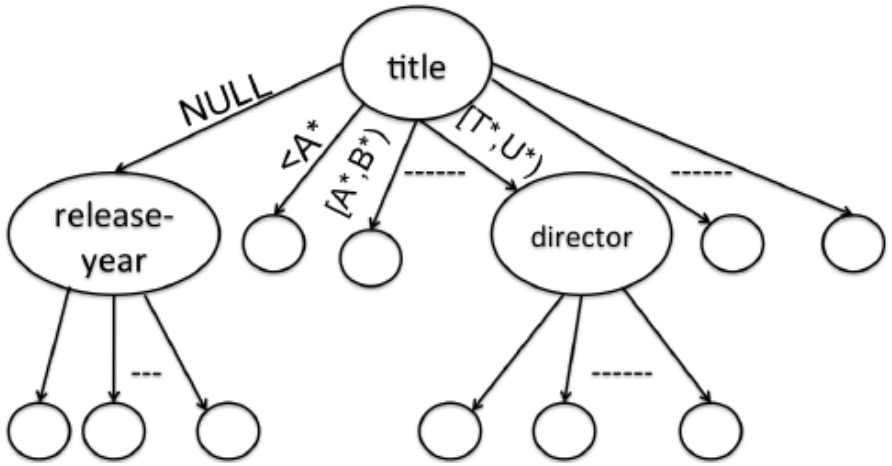
- Solution: Construct blocking predicates by combining simple predicates

Complex Blocking Predicates

- Conjunction of predicates [Michelson et al AAAI'06, Bilenko et al ICDM'06]
  - {City} AND {last four digits of phone}

- Chain-trees [Das Sarma et al CIKM'12]
  - **If** ({City} = NULL or LA) **then** {last four digits of phone} AND {area code}
    else {last four digits of phone} AND {City}

- BlkTrees [Das Sarma et al CIKM'12]



Matching

Pairwise Match Score

Problem: Given a vector of component-wise similarities for a pair of records (x,y), compute P(x and y match).

Fellegi & Sunter Model [FS, Science '69]

- Record pair: *r = (x,y)* in A x B

- $\gamma = \gamma(r)$ is a comparison vector
  - E.g., γ = ["Is x.name = y.name?", "Is x.address = y.address?" …]
  - Assume binary vector for simplicity

- *M* : set of matching pairs of records
- *U* : set of non-matching pairs of records

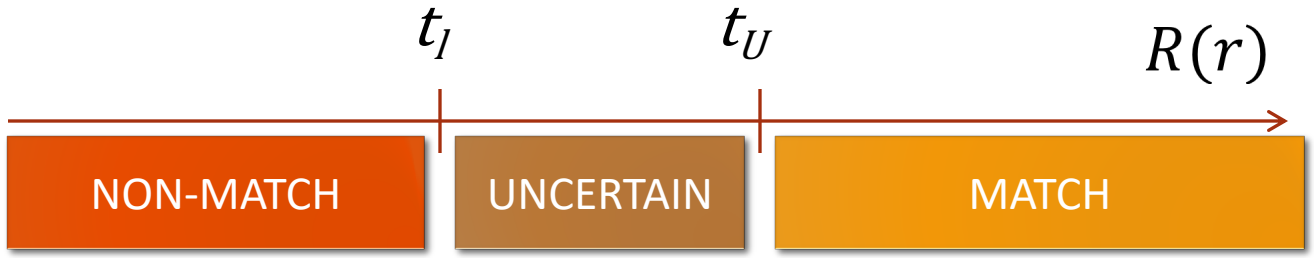
Fellegi & Sunter Model [FS, Science '69]

- *r = (x,y)* is record pair, γ is comparison vector, *M* matches, *U* non-matches

- Linkage decisions are based on:

$$R(r) = \frac{m(\gamma)}{u(\gamma)} = \frac{P(\gamma \mid r \in M)}{P(\gamma \mid r \in U)}$$

- **Linkage Rule: L(t_l, t_u)**



Error due to a Linkage Rule

- Type I Error: *r = (x,y)* in U, but the linkage rule calls it a match

$$P(L_{match}|U) = \sum_{\gamma \epsilon \Gamma} u(\gamma) \cdot P(L_{match}|\gamma)$$

- Type II Error: *r = (x,y)* in M, but the linkage rule calls it a non-match

$$P(L_{non}|M) = \sum_{\gamma \epsilon \Gamma} m(\gamma) \cdot P(L_{non}|\gamma)$$

Optimal Linkage Rule

- $L^* = (t_l^*, t_u^*)$ is an optimal decision rule for comparison space Γ with error bounds μ and λ, if
  - L^* meets the type I and type II requirements

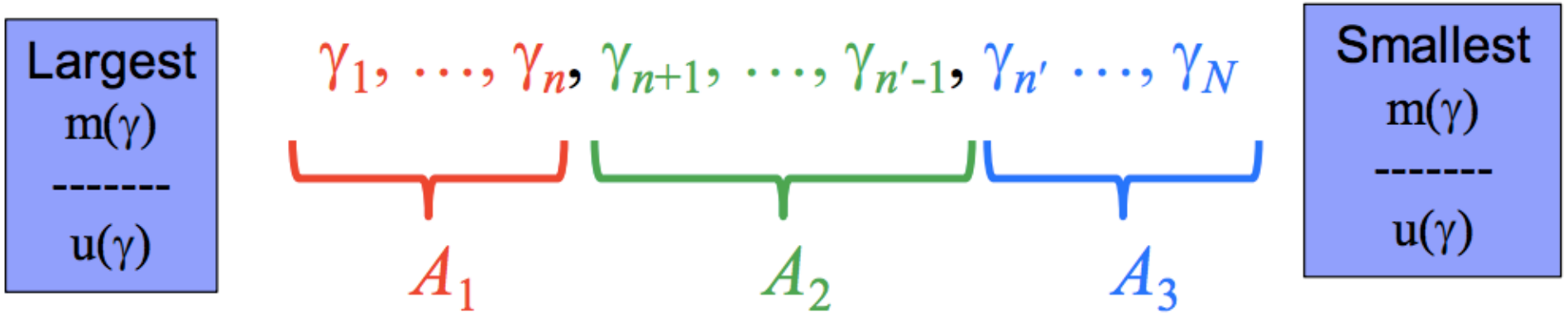
$$P(L_{match}|U) \leq \mu, \quad P(L_{non}|M) \leq \lambda$$

  - L^* has the least conditional probabilities of *not making a decision.* That is for all other decision rules L (with error bounds μ and λ),

$$P(L^*_{uncertain}|U) \leq P(L_{uncertain}|U)$$
$$P(L^*_{uncertain}|M) \leq P(L_{uncertain}|M)$$

Finding the Optimal Linkage Rule

- Suppose there are N comparison vectors
- Sort them in decreasing order of $m(\gamma) / u(\gamma)$



- Pick the largest n $\mu \geq \sum_{i=1}^{n} u(\gamma_i)$, $\lambda \geq \sum_{i=1}^{n} m(\gamma_i)$

Using Fellegi Sunter in Practice

- Γ is usually high dimensional (computing m(γ) and u(γ) is inefficient)
  - Use conditional independence of features in γ given match or non-match
  - Naïve Bayes assumption

- Computing P(γ | r ε M) requires some knowledge of matches.
  - Supervised learning (assume a training set is provided)
  - EM-based techniques can used to learn the parameters jointly while identifying matches.

ML Pairwise Approaches

- Supervised machine learning algorithms
  - Decision trees
    - [Cochinwala et al, IS01]
  - Support vector machines
    - [Bilenko & Mooney, KDD03]; [Christen, KDD08]
  - Ensembles of classifiers
    - [Chen et al., SIGMOD09]
  - Conditional Random Fields (CRF)
    - [Gupta & Sarawagi, VLDB09]
  - … and many others.

- Issues:
  - **Training set generation**
  - Imbalanced classes – many more negatives than positives
  - Misclassification cost

Creating a Training Set is a key issue

- Constructing a training set is hard – since most pairs of records are "easy non-matches".
  - 100 records from 100 cities.
  - Only 10^6 pairs out of total 10^8 (1%) come from the same city

- Some pairs are hard to judge even by humans
  - Inherently ambiguous
    - E.g., Paris Hilton (person or business)
  - Missing attributes
    - Starbucks, Toronto vs Starbucks, Queen Street ,Toronto

Constraints

- 1-1 matching
- Transitive closure
- Relational coherency

Quick refresher on string matching

Isn’t the Problem Solved?

String.equalsIgnoreCase(String x)

Multiple John Singer Sargents?

```
dallas:John_Singer_Sargent
  a foaf:Person;
  :dateOfBirth "1856" ;
  :dateOfDeath "1925" ;
  :name "John Singer Sargent" .
```

string_match("John Singer Sargent", "John S. Sargent") = ???

```
ima:John_Singer_Sargent
  a foaf:Person;
  :dateOfBirth "1856" ;
  :dateOfDeath "1925" ;
  :name "John S. Sargent" .
```

Problem Definition

Given X and Y sets of strings

Find pairs (x, y)
such that both x and y
refer to the same real world entity

"John S. Sargent"

"John Singer Sargent"

Problem Definition

Given X and Y sets of strings

Find pairs (x, y)
such that both x and y
refer to the same real world entity

We can use precision and recall to evaluate algorithms

Problem Definition

Given X and Y sets of strings

Find pairs (x, y)
such that both x and y
refer to the same real world entity

fraction of pairs found that are correct

We can use precision and recall to evaluate algorithms

fraction of pairs found

Why Strings Don't Match Perfectly?

typos "Joh" vs "John"

OCR errors "J0hn" vs "John"

formatting conventions "03/17" vs "March 17"

abbreviations "J. S. Sargent" vs "John Singer Sargent"

nick names "John" vs "Jock"

word order "Sargent, John S." vs "John S. Sargent"

Types of Similarity Metrics

- Sequence based
- Set based
- Hybrid
- Phonetic

Sequence Based Metrics

Edit Distance

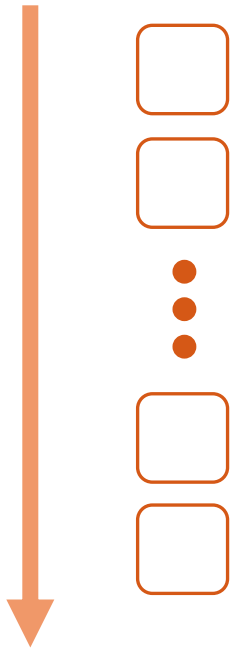
"J0n Singer Sargent"

"John S. Sargent"

- insert character
- delete character
- substitute character
- transpose character
- …

Edit Distance

"J0n Singer Sargent"

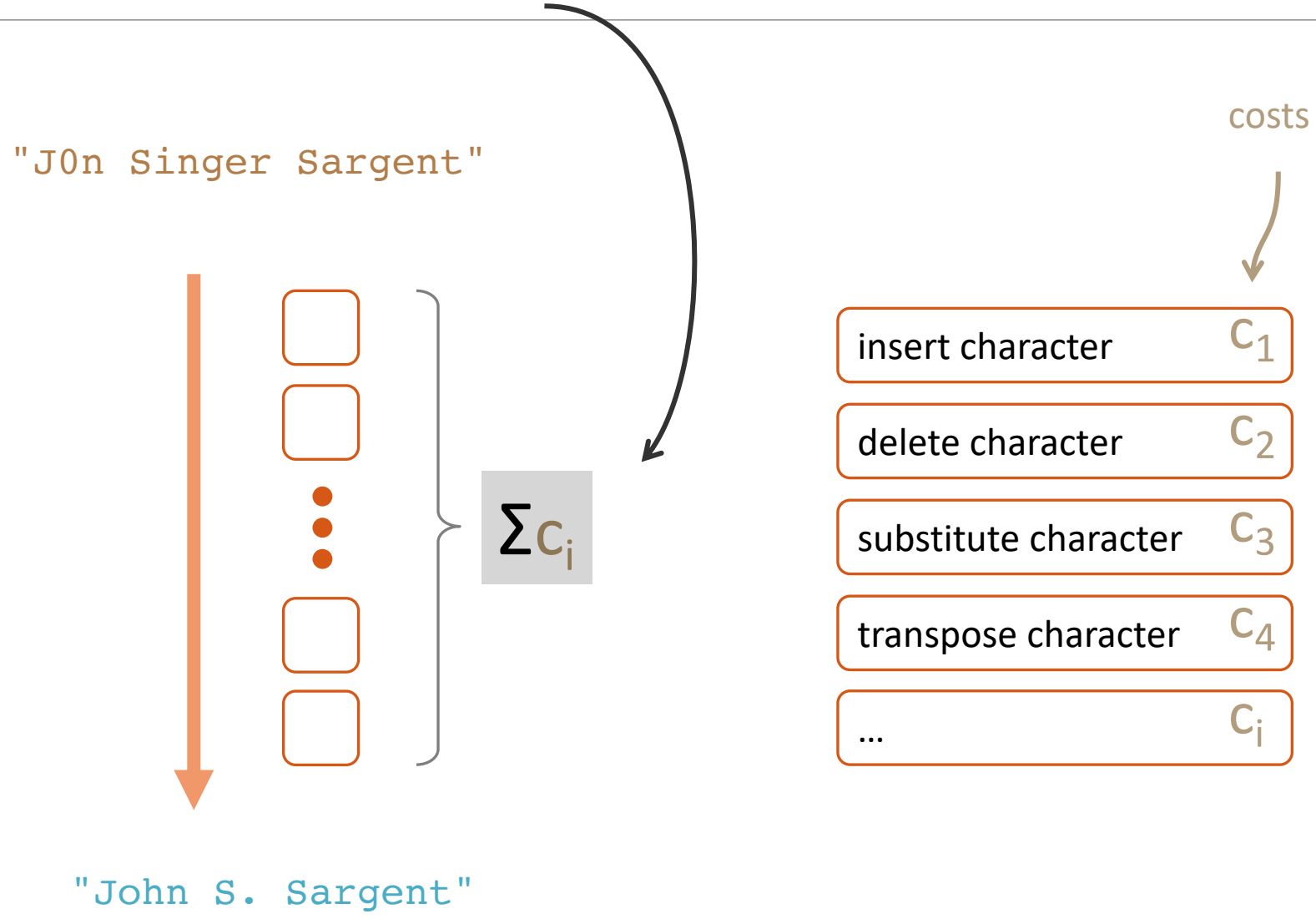


"John S. Sargent"

costs

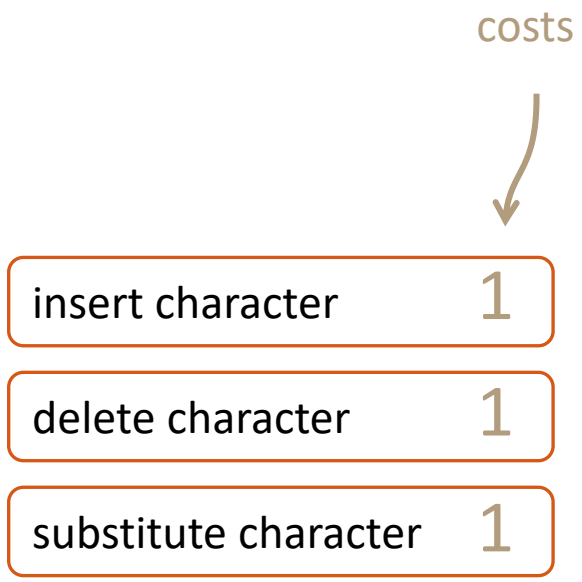
insert character	c_1
delete character	c_2
substitute character	c_3
transpose character	c_4
…	c_i

Edit Distance



Levenshtein Distance

Edit distance:



lev(x, y) is the minimum cost to transform x to y

Online calculator: http://planetcalc.com/1721/

Levenshtein Distance Examples

```
lev(John Singer Sargent,
    John S. Sargent)
```

=

Levenshtein Distance Examples

```
lev(John Singer Sargent,
    John S. Sargent)
```

= 5

Levenshtein Distance Examples

```
lev(John Singer Sargent,
    John S. Sargent)
```
= 5

```
lev(John Singer Sargent,
    Jane Klinger Sargent)
```
=

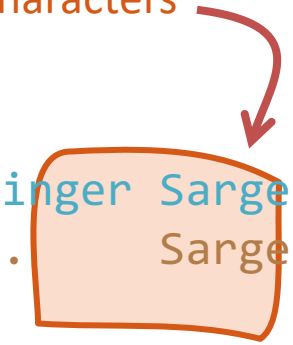
Levenshtein Distance Examples

```
lev(John Singer Sargent,
    John S. Sargent)
```
= 5

```
lev(John Singer Sargent,
    Jane Klinger Sargent)
```
= 5

Levenshtein Distance Examples

Too high a cost for deleting a sequence of characters

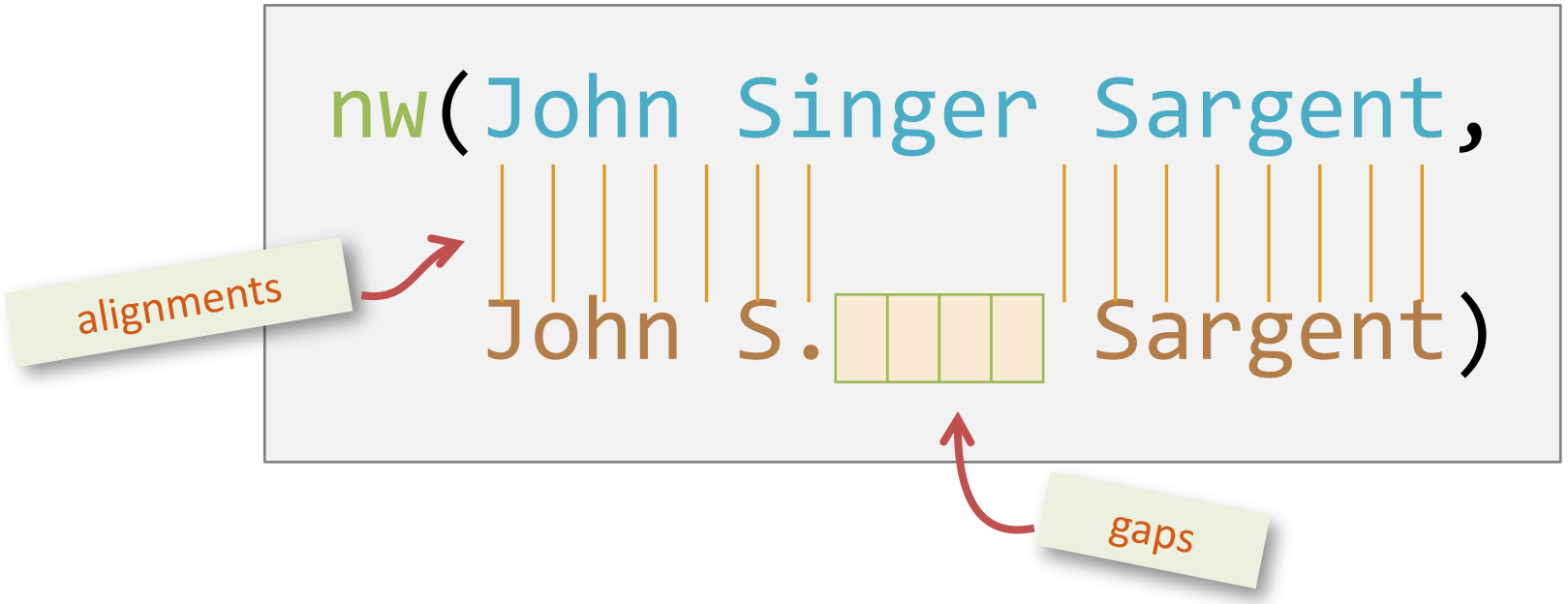


lev(John Singer Sargent,
John S. Sargent) = 5

lev(John Singer Sargent,
Jane Klinger Sargent) = 5

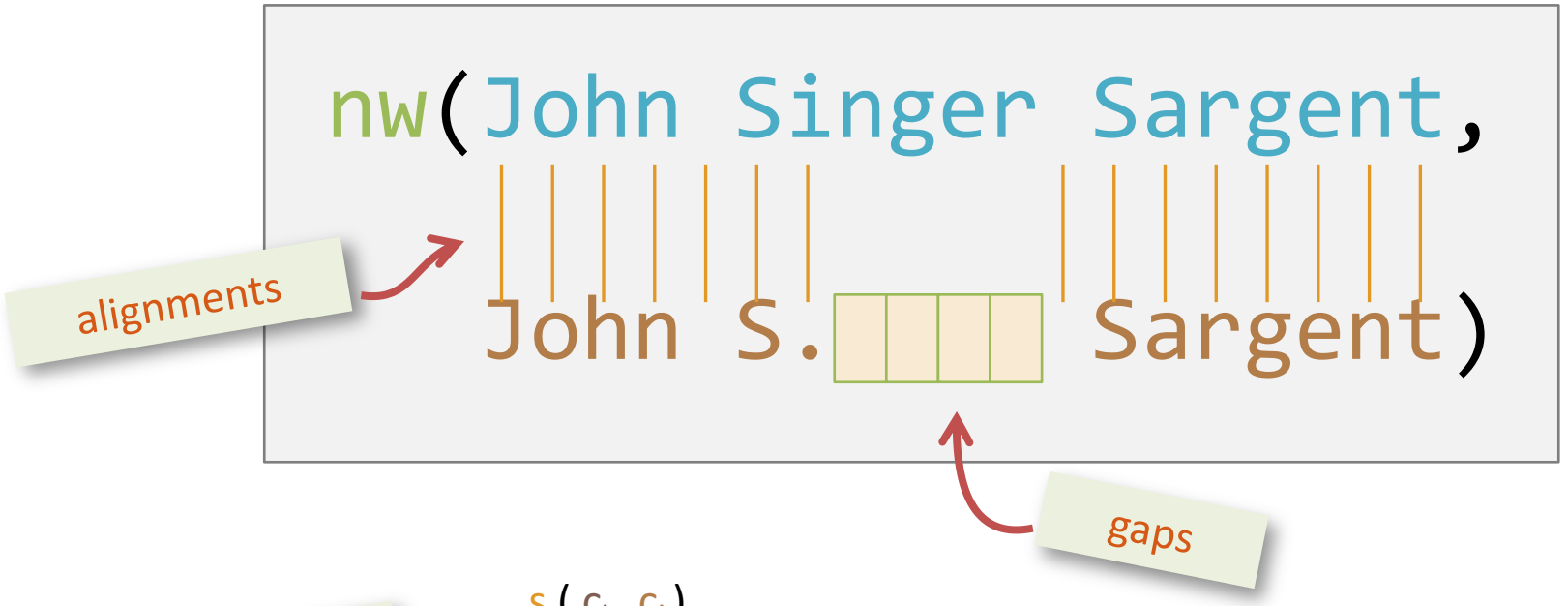
Needleman-Wunch Measure

Generalization of levenstein(x, y)



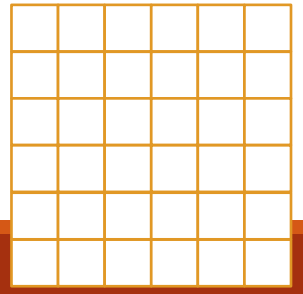
Needleman-Wunch Measure

Generalization of levenstein(x, y)



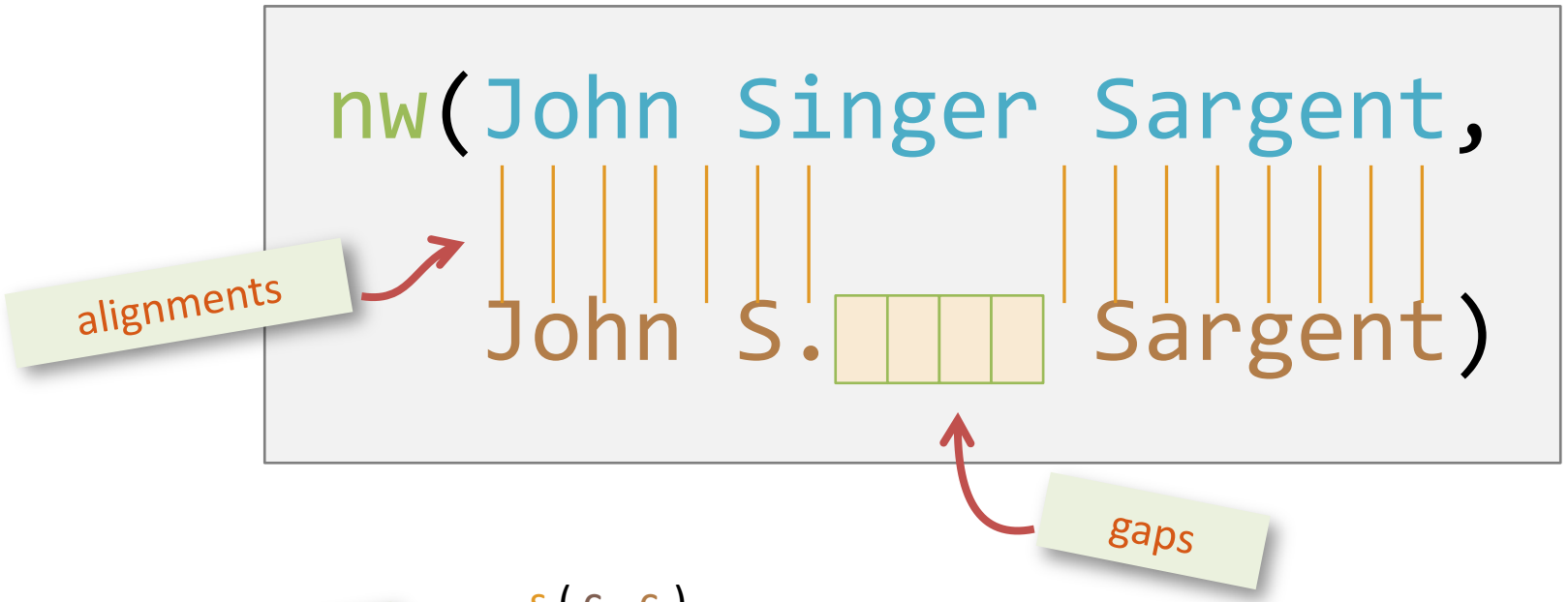
$s(c_i, c_j)$

alignment score matrix



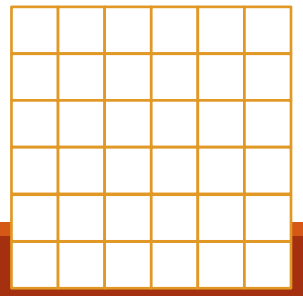
Needleman-Wunch Measure

Generalization of levenstein(x, y)



$s(c_i, c_j)$

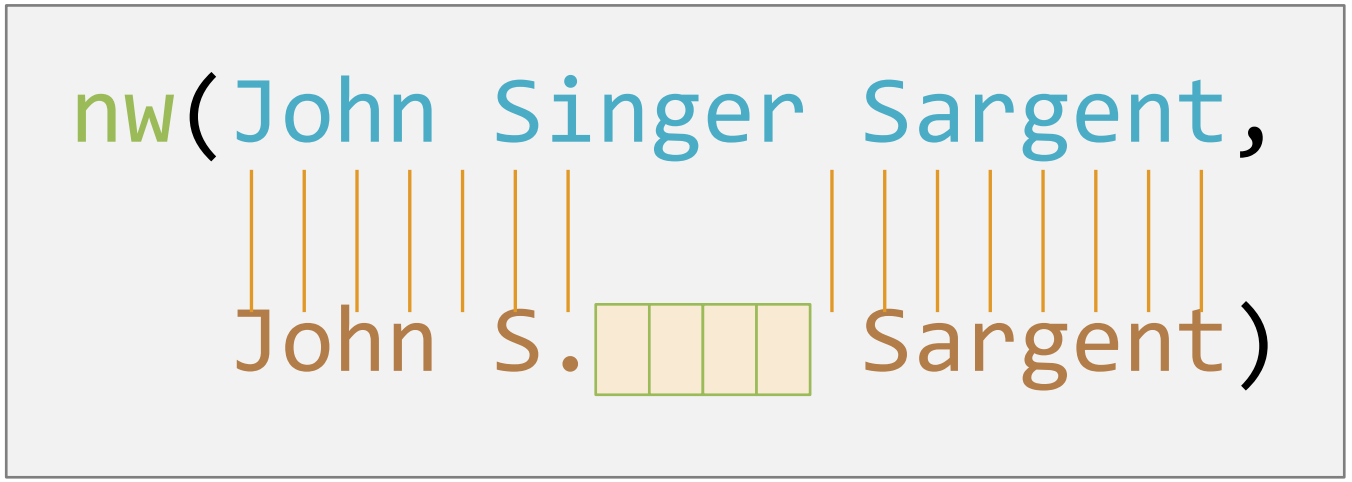
alignment score matrix



gap score

Needleman-Wunch Measure

Generalization of levenstein(x, y)

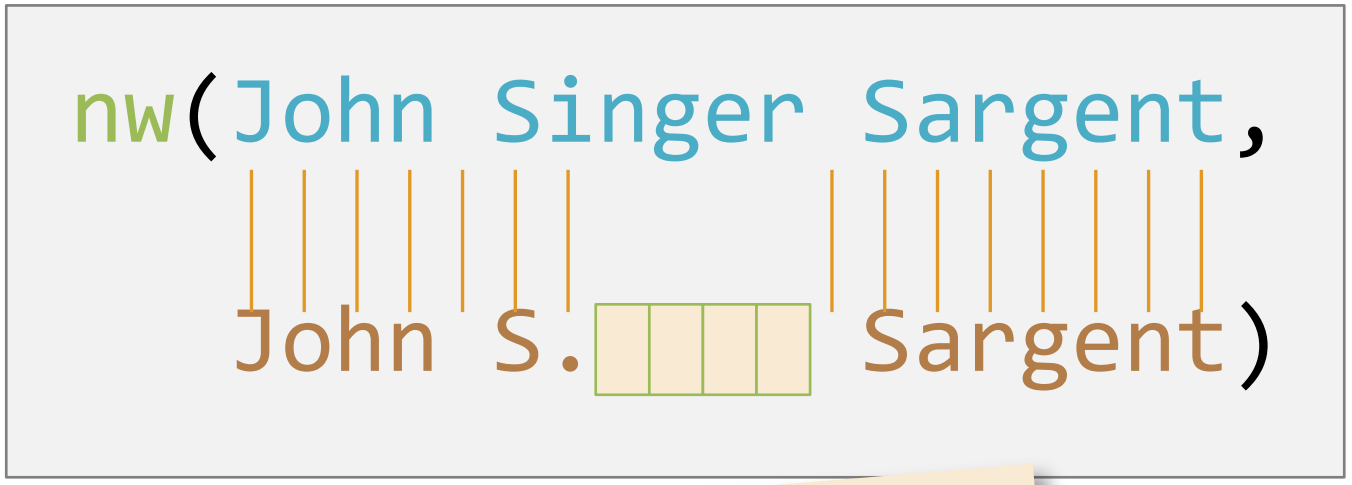


$$s(c_i, c_j) = \begin{cases} 2 & \text{if } c_i = c_j \\ -1 & \text{if } c_i \mathrel{!}= c_j \end{cases}$$

gap-score = -0.5

Needleman-Wunch Measure

Generalization of levenstein(x, y)



$$s(c_i, c_j) = \begin{cases} 2 & \text{if } c_i = c_j \\ -1 & \text{if } c_i \mathrel{!}= c_j \end{cases}$$

gap-score = -0.5

Comparison

	Levenshtein	Needleman-Wunch
Costs	1	matrix
Operations	insert/delete	gaps
Result	distance	similarity

OCR errors "J0hn" vs "John"

```
score(o,0) = -0.2
score(m,0) = -1.0
```

lower penalty

Needleman-Wunch Example

```
nw(John Singer Sargent,
   John S.      Sargent)
```

```
2 * 14  + (-1) * 1  + (-0.5) * 4 = 25
```

```
2 * 14  + (-1) * 1  + (-0.5) * 8 = 23
```

```
nw(John Stanislaus Sargent,
   John S.         Sargent)
```

Needleman-Wunch Example

```
nw(John Singer Sargent,
   John S.     Sargent)
```

(-1) * 1 + (-0.5) * 4 = 25

Longer gaps are penalized more

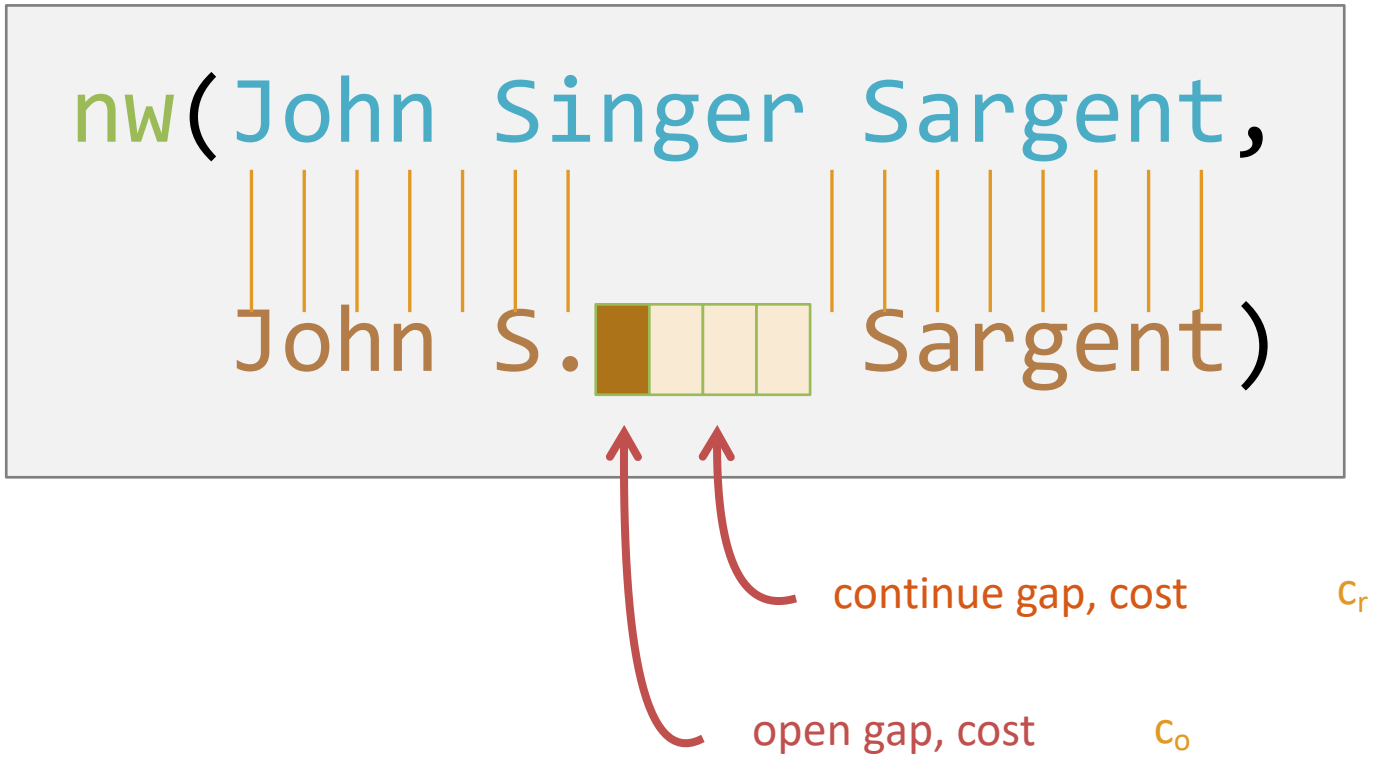
Bad for names

2 * 14 + (-1) * 1 + (-0.5) * 8 = 23

```
nw(John Stanislaus Sargent,
   John S.         Sargent)
```

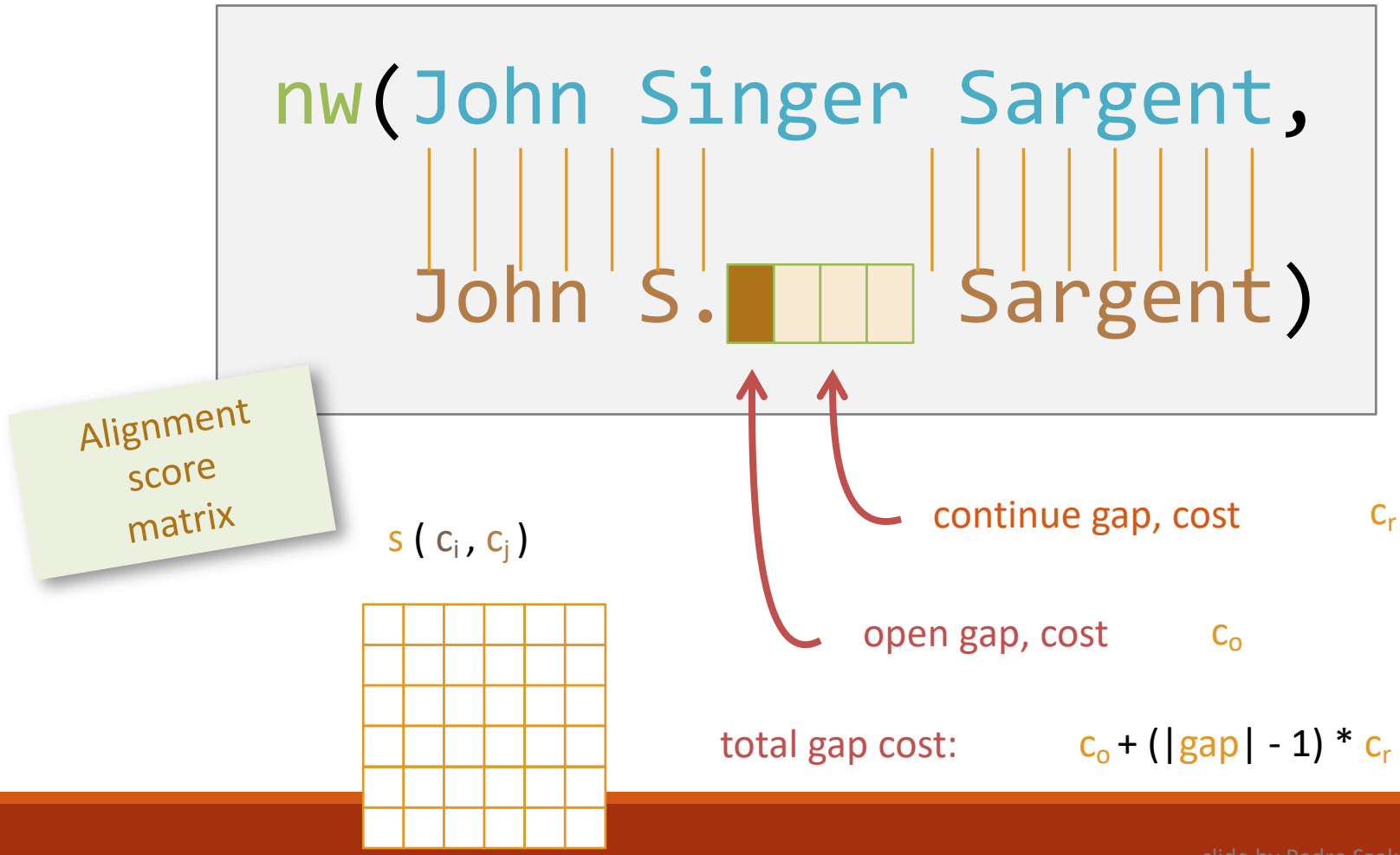
Affine Gap Measure

Generalization of needleman-wunch(x, y)



Affine Gap Measure

Generalization of needleman-wunch(x, y)



Jaro Similarity Measure

- Get points for having characters in common
  - but only if they are “close by”
- Get points for common characters in the same order
  - lose points for transpositions

Jaro Similarity Measure

jaro(x, y)

$$\text{max-distance} = \frac{\max(\,|x|\,,\,|y|\,)}{2} - 1$$

x_i matches y_j if

$$\begin{cases} x_i = y_j \\ |i - j| <= \text{max-distance} \end{cases}$$

m = number of matching characters

t = number of transpositions
(of matching characters)

Jaro Similarity Measure

$$\text{max-distance} = \frac{\max(\,|x|\,,\,|y|\,)}{2} - 1$$

m = number of matching characters

t = number of transpositions

$$\text{jaro}(x, y) = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|x|} + \frac{m}{|y|} + \frac{m - t}{m}\right) \end{cases}$$

Jaro Example

```
lev(DIXON, DICKSONX)  = 4     (4/8 = 0.5)

jaro(DIXON, DICKSONX) = ???
```

Jaro Example

	D	I	X	O	N
D	1	0	0	0	0
I	0	1	0	0	0
C	0	0	0	0	0
K	0	0	0	0	0
S	0	0	0	0	0
O	0	0	0	1	0
N	0	0	0	0	1
X	0	0	1	0	0

Jaro Example

	D	I	X	O	N
D	1	0	0	0	0
I	0	1	0	0	0
C	0	0	0	0	0
K	0	0	0	0	0
S	0	0	0	0	0
O	0	0	0	1	0
N	0	0	0	0	1
X	0	0	0	0	0

$|x| = 5$

$|y| = 8$

$\text{max-distance} = (8/2) - 1 = 3$

$m = 4$

$t = 0$

Jaro Example

	D	I	X	O	N
D	1	0	0	0	0
I	0	1	0	0	0
C	0	0	0	0	0
K	0	0	0	0	0
S	0	0	0	0	0
O	0	0	0	1	0
N	0	0	0	0	1
X	0	0	0	0	0

$|x| = 5$

$|y| = 8$

$m = 4$

$t = 0$

$$\frac{1}{3}\left(\frac{m}{|x|} + \frac{m}{|y|} + \frac{m - t}{m}\right)$$

$$\frac{1}{3}\left(\frac{4}{5} + \frac{4}{8} + \frac{4 - 0}{4}\right)$$

$= 0.767$

Set-Based Metrics

Set-Based Metrics

Generate set of tokens from the strings

Measure similarity between the sets of tokens

Tokenizing a String

Words

Tokenizing a String

Words

q-grams: substrings of length q

“david smith” 3-grams
{##d, #da, dav, avi, …, h##}

Jaccard Measure

$$B_x = \text{tokens}(x)$$

$$B_y = \text{tokens}(y)$$

$$\text{jaccard}(x,y) = \frac{|B_x \cap B_y|}{|B_x \cup B_y|}$$

jaccard(dave, dav)
B_x = {#d, da, av, ve, e#}
B_y = {#d, da, av, v#}
jaccard(x,y) = 3/6

TF/IDF Measure

TF = term frequency
IDF = inverse document frequency

x = Apple Corporation, CA
y = IBM Corporation, CA
z = Apple Corp
…
blah blah Corporation

lev(x, y) > = < lev(x, z)

???

TF/IDF Measure

TF = term frequency
IDF = inverse document frequency

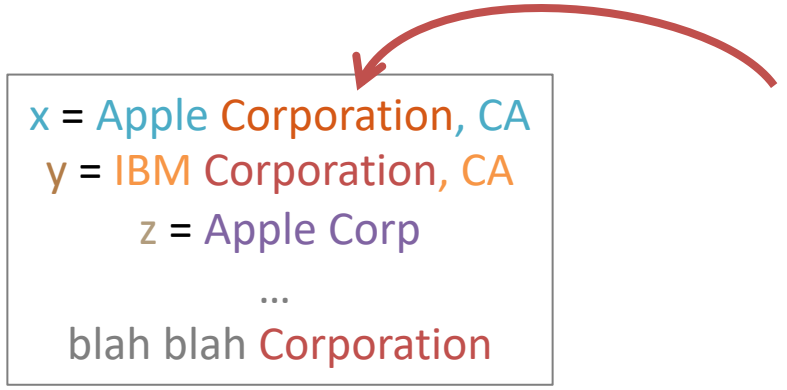
x = Apple Corporation, CA
y = IBM Corporation, CA
z = Apple Corp
…
blah blah Corporation

lev(x, y) < lev(x, z)

… but intuitively (x, z) is a better match

TF/IDF Measure

TF = term frequency
IDF = inverse document frequency



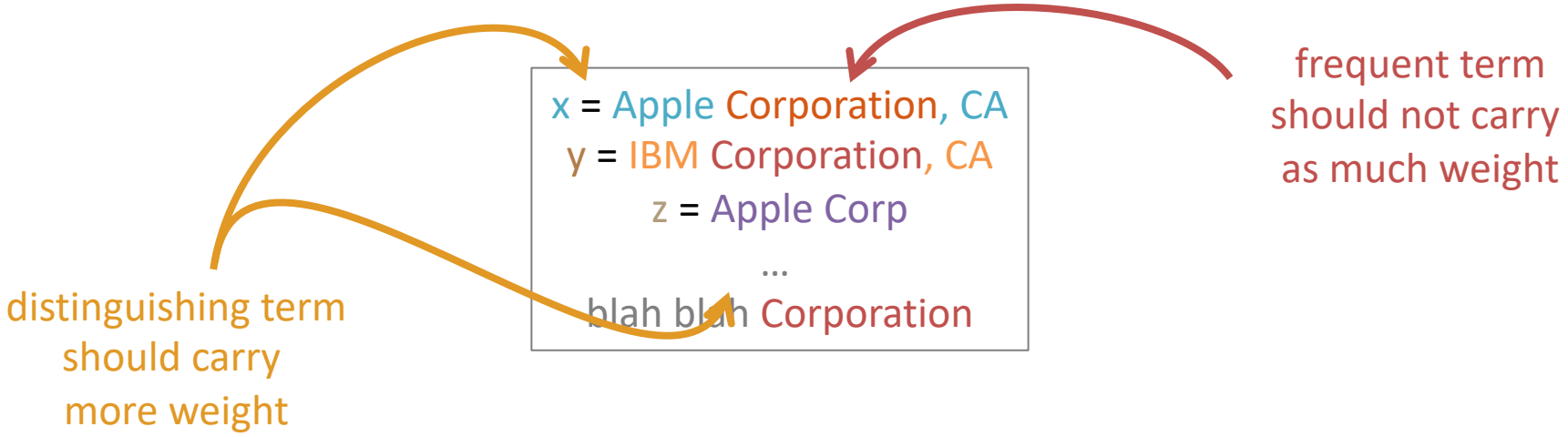
frequent term should not carry as much weight

lev(x, y) > lev(x, z)

… but intuitively (x, z) is a better match

TF/IDF Measure

TF = term frequency
IDF = inverse document frequency



lev(x, y) > lev(x, z)

… but intuitively (x, z) is a better match

Term Frequencies and Inverse Document Frequencies

- Assume x and y are taken from a collection of strings
- Each string is converted into a bag of terms called a document
- term frequency tf(t,d) =
  - number of times term t appears in document d
- inverse document frequency idf(t) =
  - N / $\mathbf{N}_d$, number of documents in collection divided by number of documents that contain t
  - note: in practice, idf(t) is often defined as log(N / $\mathbf{N}_d$)

Example

$x = aab \Rightarrow B_x = \{a, a, b\}$

$y = ac \Rightarrow B_y = \{a, c\}$

$z = a \Rightarrow B_z = \{a\}$

$tf(a, x) = 2$

$tf(b, x) = 1$

...

$tf(c, z) = 0$

$idf(a) = 3/3 = 1$

$idf(b) = 3/1 = 3$

$idf(c) = 3/1 = 3$

Feature Vectors

- Each document d is converted into a feature vector $\mathbf{v}_d$
- $\mathbf{v}_d$ has a feature $\mathbf{v}_d(\mathbf{t})$ for each term t
  - value of $\mathbf{v}_d(\mathbf{t})$ is a function of TF and IDF scores
  - here we assume $\mathbf{v}_d(\mathbf{t})$ = tf(t,d) * idf(t)

$x = aab \Rightarrow B_x = \{a, a, b\}$

$y = ac \Rightarrow B_y = \{a, c\}$

$z = a \Rightarrow B_z = \{a\}$

tf(a, x) = 2
tf(b, x) = 1
...
tf(c, z) = 0

idf(a) = 3/3 = 1
idf(b) = 3/1 = 3
idf(c) = 3/1 = 3

	a	**b**	**c**
$\mathbf{v}_x$	2	3	0
$\mathbf{v}_y$	3	0	3
$\mathbf{v}_z$	3	0	0

TF/IDF Similarity Score

- Let p and q be two strings, and T be the set of all terms in the collection
- Feature vectors $\mathbf{v}_p$ and $\mathbf{v}_q$ are vectors in the |T|-dimensional space wher each dimension corresponds to a term
- TF/IDF score of p and q is the cosine of the angle between $\mathbf{v}_p$ and $\mathbf{v}_q$
  - $s(p,q) = \sum_{t \in T} v_p(t) * v_q(t) \; / [\sqrt{\sum_{t \in T} v_p(t)^2} * \sqrt{\sum_{t \in T} vq(t)^2}]$

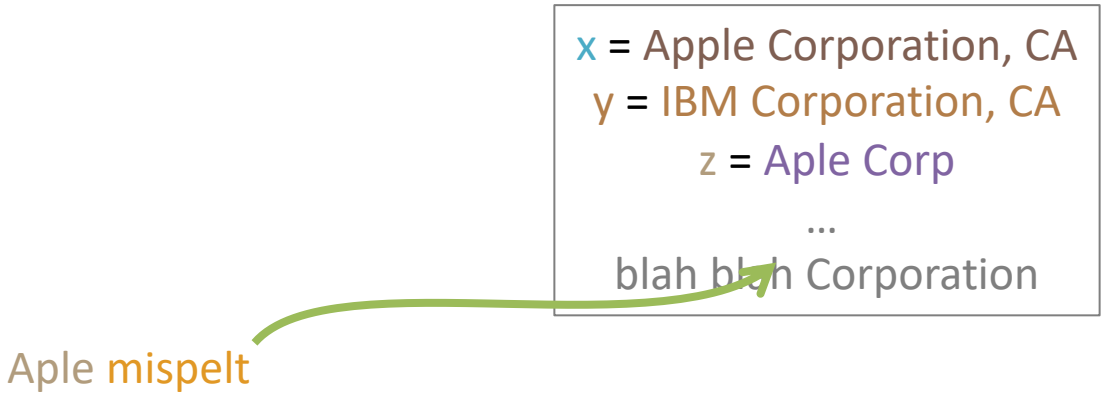
TF/IDF Similarity Score

- Score is high if strings share many frequent terms
  - terms with high TF scores
- Unless these terms are common in other strings
  - i.e., they have low IDF scores
- Dampening TF and IDF as commonly done in practice
  - use v_d(t) = log(tf(t,d) + 1) * log(idf(t)) instead of
    v_d(t) = tf(t,d) * idf(t)
- Normalizing feature vectors
  - v_d(t) = v_d(t) / $\sqrt{\sum_{\{t \in T\}} v_d(t)^2}$

Hybrid Similarity Measures

Hybrid Measures

Do the set-based thing
but
use a similiarity metric for each element of the set



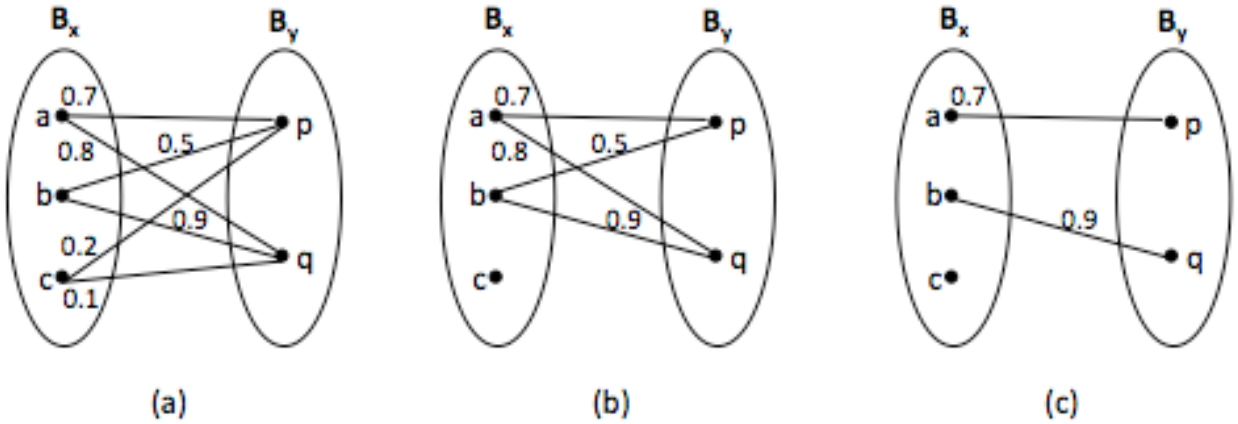
Generalized Jaccard Measure

- Jaccard measure
  - considers overlapping tokens in both x and y
  - a token from x and a token from y must be identical to be included in the set of overlapping tokens
  - this can be too restrictive in certain cases
- Example:
  - matching taxonomic nodes that describe companies
  - “Energy & Transportation” vs. “Transportation, Energy, & Gas”
  - in theory Jaccard is well suited here, in practice Jaccard may not work well if tokens are commonly misspelled
    - e.g., energy vs. eneryg
  - generalized Jaccard measure can help such cases

Generalized Jaccard Measure

- Let $B_x = \{x_1, \ldots, x_n\}$, $B_y = \{y_1, \ldots, y_m\}$
- Step 1: find token pairs that will be in the "softened" overlap set
  - apply a similarity measure s to compute sim score for each pair (x_i, y_j)
  - keep only those score > a given threshold α, this forms a bipartite graph G
  - find the maximum-weight matching M in G
- Step 2: return normalized weight of M as generalized Jaccard score
  - $GJ(x,y) = \sum_{(xi,yj) \text{ in } M} s(x_i,y_j) / (|B_x| + |B_y| - |M|)$

Generalized Jaccard Example



$\alpha = 0.5$

- Generalized Jaccard score: $(0.7 + 0.9)/(3 + 2 - 2) = 0.53$

Phonetic Similarity Measures

Phonetic Similarity Measures

- Match strings based on their sound, instead of appearances
- Very effective in matching names, which often appear in different ways that sound the same
  - e.g., Meyer, Meier, and Mire; Smith, Smithe, and Smythe
- Soundex is most commonly used

The Soundex Measure

- Used primarily to match surnames
  - maps a surname x into a 4-letter code
  - two surnames are judged similar if share the same code
- Algorithm to map x into a code:
  - Step 1: keep the first letter of x, subsequent steps are performed on the rest of x
  - Step 2: remove all occurences of W and H. Replace the remaining letters with digits as follows:
    - replace B, F, P, V with 1, C, G, J, K, Q, S, X, Z with 2, D, T with 3, L with 4, M, N with 5, R with 6
  - Step 3: replace sequence of identical digits by the digit itself
  - Step 4: Drop all non-digit letters, return the first four letters as the soundex code

The Soundex Measure

- Example: x = Ashcraft
  - after Step 2: A226a13, after Step 3: A26a13, Step 4 converts this into A2613, then returns A261
  - Soundex code is padded with 0 if there is not enough digits
- Example: Robert and Rupert map into R163
- Soundex fails to map Gough and Goff, and Jawornicki and Yavornitzky
  - designed primarily for Caucasian names, but found to work well for names of many different origins
  - does not work well for names of East Asian origins
    - which uses vowels to discriminate, Soundex ignores vowels