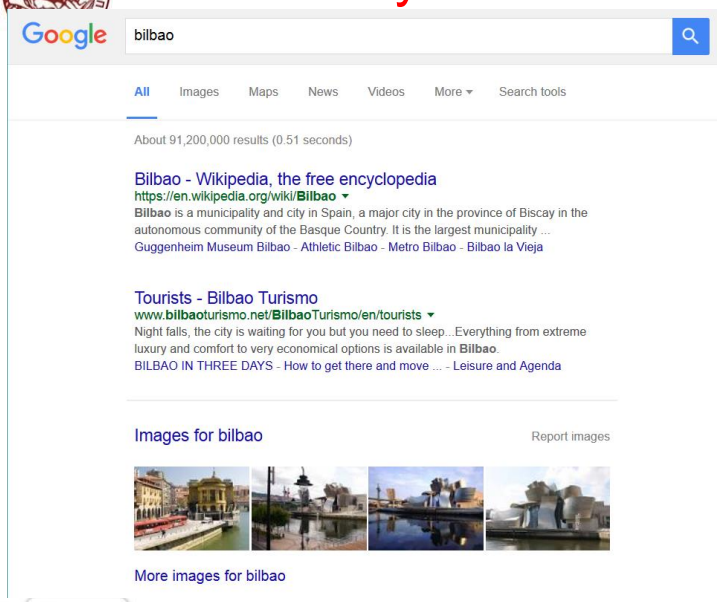
Entity Resolution in the Web of Data

Vassilis Christophides
christop@csd.uoc.gr
http://www.csd.uoc.gr/~hy562
University of Crete, Fall 2019

Fall 2019

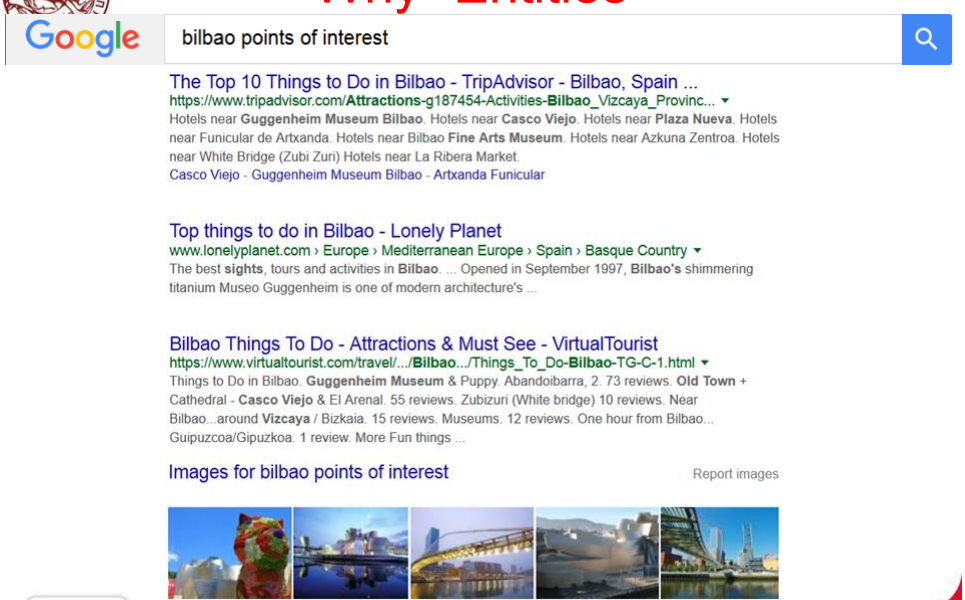
Describing and Linking Entities: Entity-Centric Applications & Knowledge Bases

Why “Entities”



Fall 2019

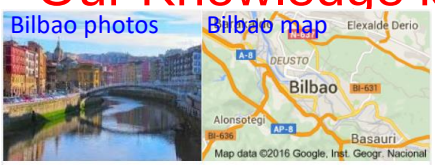
Why “Entities”



"Entities" is What a Large Part of Our Knowledge is About

Bilbao photos

Bilbao map



One single Entity Pane can answer many user queries and satisfy users' diverse information needs

Bilbao

Municipality in Spain

Bilbao, an industrial port city in northern Spain, is
mountains. It's the de facto capital of Basque Cou
filled downtown. It's famed for the Frank Gehry–d
Museum Bilbao, which sparked revitalization whe
museum houses modern and contemporary work
but it's the curvy, titanium-clad building that receiv

Getting there: 1 h 30 min flight, around €200. Vie

Weather: 19°C, Wind W at 5 km/h, 79% Humidity

Local time: Monday 9:47 PM

Population: 346,574 (2014) Instituto Nacional de Estadística

Colleges and Universities: University of the Basque Country, University of Deusto

Points of interest View 10+ more

Guggenh... Museum | Bilbao Fine Arts | Plaza Nueva | Zubizuri | Doña Casilda

Flights from Paris (all airports) to Bilbao, Spain (BIO)
www.google.fr/flights
Sponsored

Paris (all airports)	Bilbao, Spain (BIO)		
Wed, June 22	Sun, June 26		
Non-stop	Air France	1h 35m	from €303
	Air Europa	1h 35m	from €447
Connecting	Multiple airlines	1h 35m+	from €181
	Iberia	3h 35m+	from €186
	Other airlines	1h 35m+	from €210

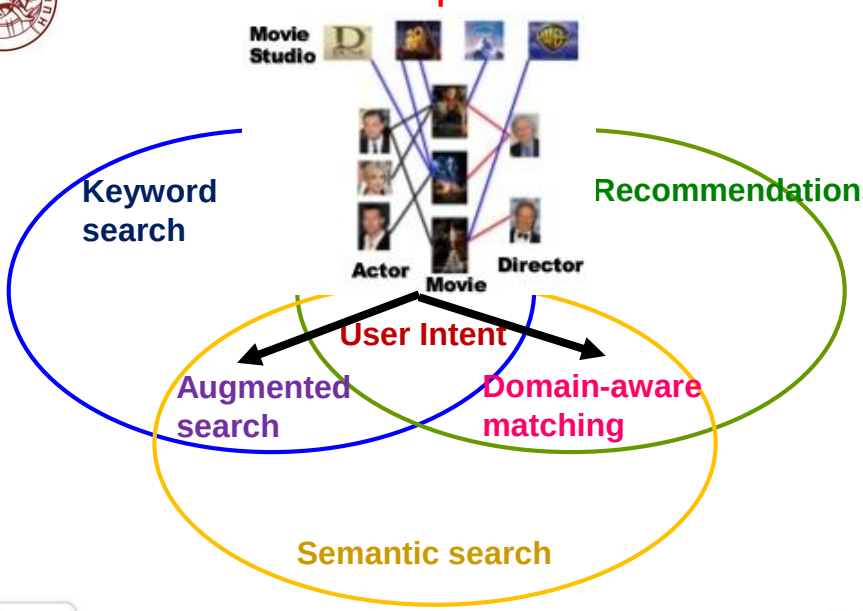
More Google flight results »

Weather in Bilbao
Time in Bilbao
Bilbao Population

Popular schools in Bilbao

Bilbao places to go

Push/Pull Techniques for Web Content

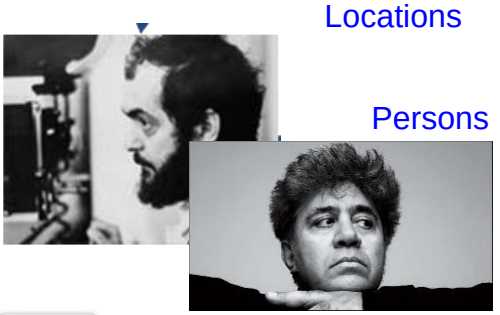


https://www.youtube.com/watch?v=L9CPOltkcJA

Core Entities

Organizations

Locations



Persons

Movies

What Is A Knowledge Base (KB)?

DBpedia

dbpedia:Stanley_Kubrick

dbo:birthPlace	**dbpedia:Manhattan**
rdf:type	foaf:Person
rdf:type	yago:AmericanFilmDirectors
rdf:type	yago:AmateurChessPlayers

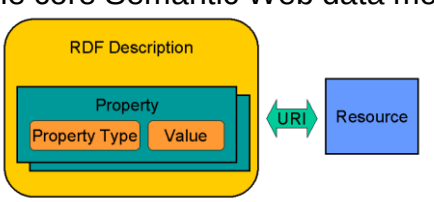
relationships

dbpedia:A_Clockwork_Orange_(film)

dbo:director	**dbpedia:Stanley_Kubrick**
dbo:Work/runtime	"136"
foaf:name	"A Clockwork Orange"

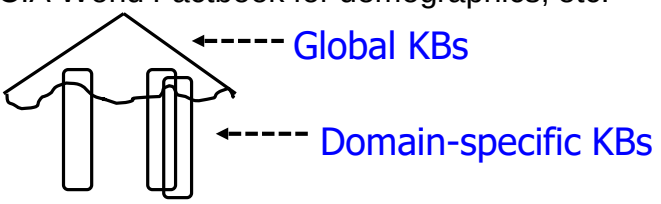
facts

- Comprehensive, machine-readable descriptions of real-world entities are hosted in knowledge bases (KB)
  - Entity names, types, attributes, relationships, provenance info
- Entities are described as instances of one or several conceptual types and may be linked through relationships
  - The core Semantic Web data model



Knowledge Bases: Scope

- Domain-specific Knowledge Base
  - Focus is on a well-defined domain:
    - IMDB for movies, Music-Brainz for music, GeoNames for geo, CIA World Factbook for demographics, etc.



- Global Knowledge Base
  - Covers a variety of knowledge across domains
    - Concept-intensive (Intensional): Cyc, WordNet
    - Entity-intensive (Extensional): DBPedia, Yago, Freebase, Knowledge Graph, Satori Bing, Knowledge Vault

Knowledge Bases: History

Cyc **WordNet**

from humans for humans

guitarist ⊂ {player,musician}
⊂ artist
algebraist
mathematician
scientist

- ∀ x: human(x) ⇒
- (∃ y: mother(x,y) ∧
- ∃ z: father(z,x))
- ∀ x,u,w: (mother(x,u) ∧
- mother(x,w)
- ⇒ u=w)

Wikipedia

4.5 Mio. English articles
20 Mio. contributors

WolframAlpha computational knowledge engine

from algorithms for machines

DBpedia

yago select knowledge

WIKIDATA

Freebase

Knowledge Graph

1985 1990 2000 2005 2010

Freebase

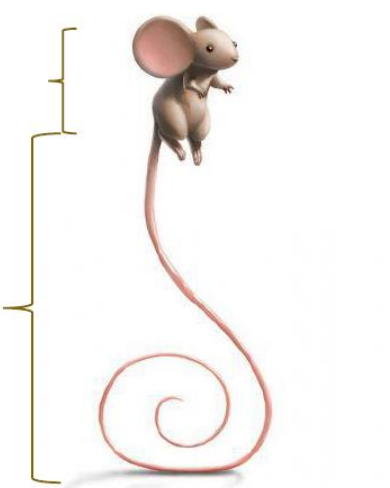
- 46.3M entities
- 1.5K entity types
- 2.67B triples
- 4.5K properties

The Long Tail of Knowledge

- Freebase is large, but still very incomplete:

Relation	% unknown in Freebase
Profession	68%
Place of birth	71%
Nationality	75%
Education	91%
Spouse	92%
Parents	94%

- We need automatic KB construction methods



http://www.flickr.com/photos/sandreli/4691045841/

Google's Knowledge Graph (GKG)

- 600M nodes (entities)
- 1.5K entity types
- 20B edges (triples)
- 35K properties

- The Knowledge Graph is used by Google to enhance its search engine's search results with semantic-search information
  - GKG is rooted in public KBs such as Freebase, Wikipedia and the CIA World Factbook
  - It's also augmented at a much larger scale focusing on a comprehensive breadth and depth of entity descriptions

Google Knowledge Vault (GKV)

Probabilistic Knowledge Fusion

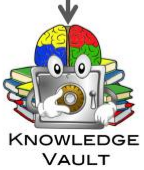
Freebase

Webmaster annotations | Tables | NL text | Page structure | Deep neural network | Path Ranking Algorithm (PRA)

Extractors | Priors

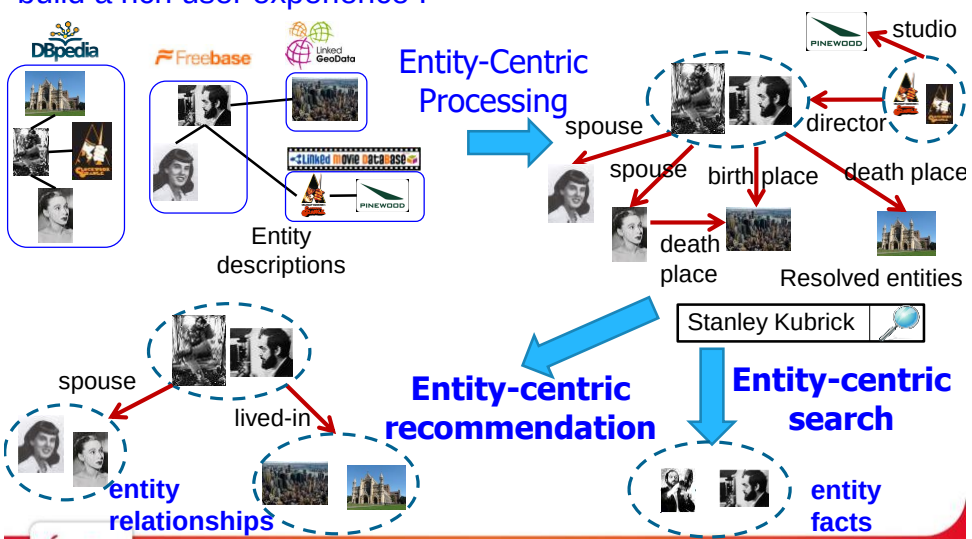
Fusion

Evaluate trustworthiness of sources by the correctness of its factual information

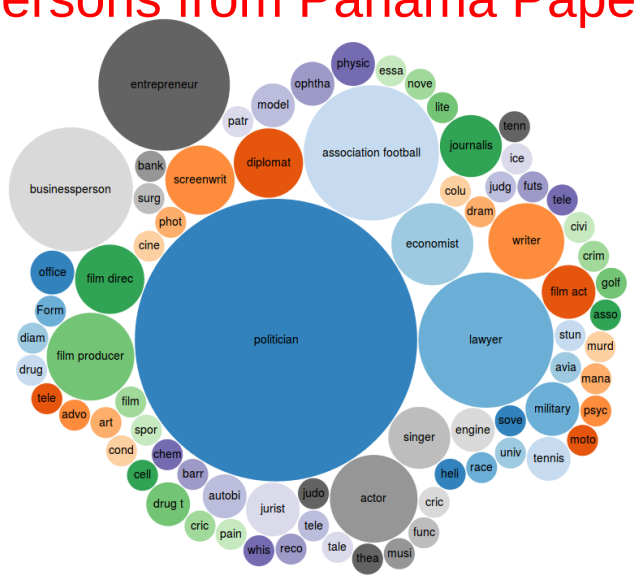


Entity-Centric Applications

Combine knowledge regarding an entity from multiple sources to build a rich user experience !



Citizen Journalism: Occupations of Persons from Panama Papers



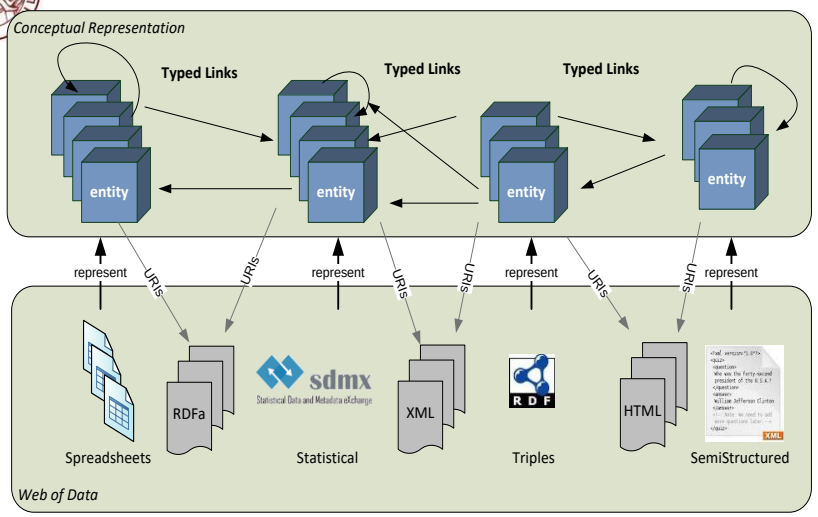
https://finnaarupnielsen.wordpress.com/tag/wikidata/

Entity-Centric Information Processing

- Automated construction of entity descriptions
  - *Information extraction*: extract new entities from web/text
  - *Link prediction*: add relationships among entities
- Entity integration and resolution
  - *Knowledge base integration*: instance & ontology mappings
  - *Entity resolution*: merging or splitting similar entities
- Entity-centric access interfaces
  - *Augmented search*: interpret the meaning of queries using entities and compute answers based on a knowledge base
  - *Entity-based matching*: recommend new entities given an entity, a user or a query
  - *Entity-centric summarization*: of textual posts in social media

Describing and Linking Entities: Linked Knowledge Bases & The Web of Data

The Web of Data



A Web of things in the world (aka **entities**), described by data on the Web

adapted from Chris Bizer, Richard Cyganiak, Tom Heath, http://linkeddata.org/guides-and-tutorials

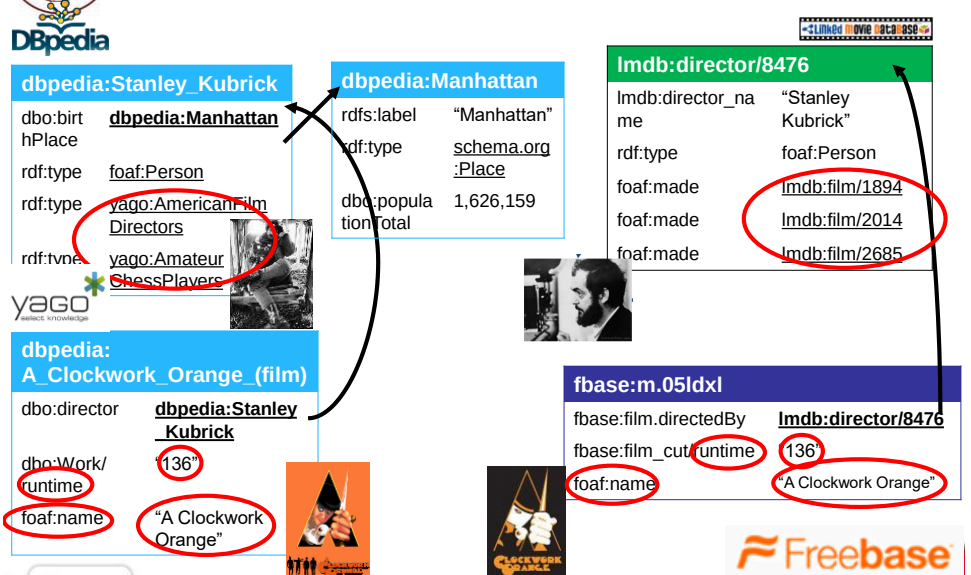
From Open to Linked Data

The Linked Data Principles

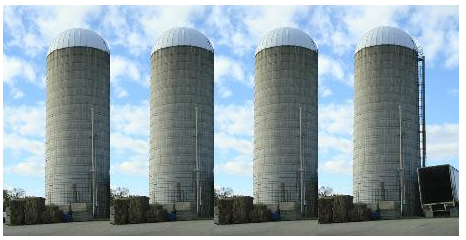
- Anyone can publish data on the Web regarding real-world entities by respecting a minimal set of syntactic conventions
  - Use URIs as names for things
  - Use HTTP URIs so that people and machines can look up those names
  - When someone looks up a URI, provide a useful description
  - Include links to other URIs, so that they can discover more things

- Data becomes self-describing
  - Applications encountering data described by an unfamiliar vocabulary, they can resolve its URIs and understand the vocabulary terms by their RDFS or OWL definitions

Linking Entities in KBs



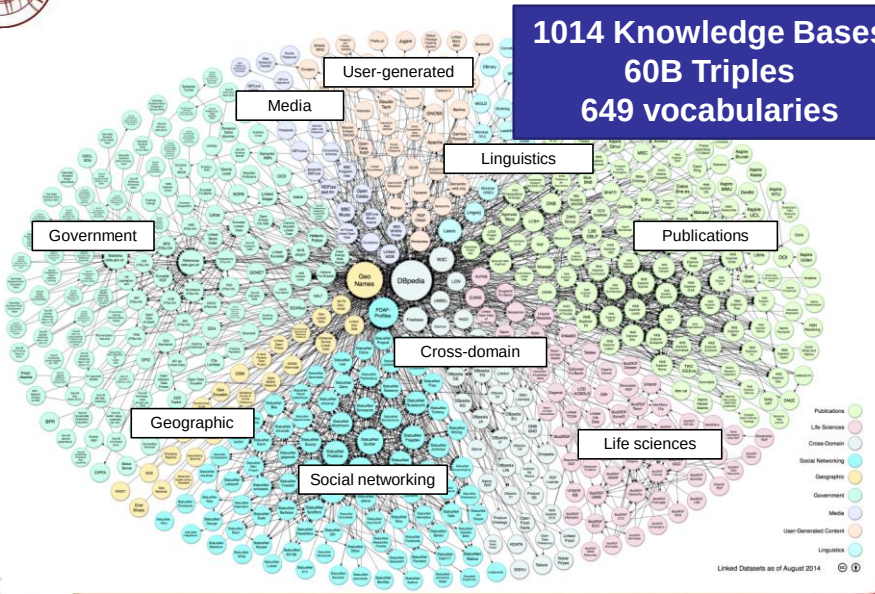
From Data Silos to the Web of Data



- Thanks to W3C standards we can support:
  - A uniform and universal access to information
  - Easy linking and re-usage of information in different contexts
  - Network effects in adding value to information
- Data Interoperability at the Web Scale
  - Connecting data from diverse sources and across domains
- Openness
  - Many common things are represented in multiple sources
  - Discover new sources at run-time by following links

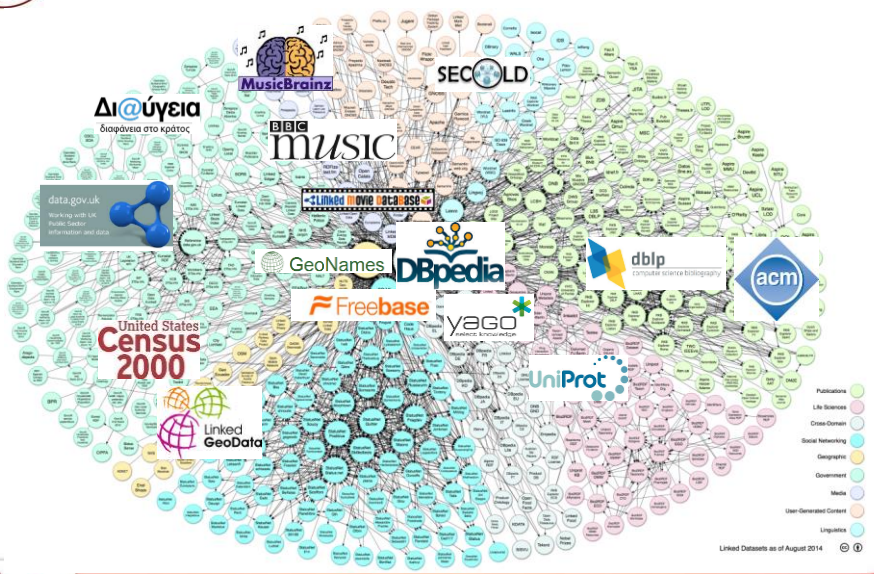
The Link Open Data (LOD) Cloud

1014 Knowledge Bases
60B Triples
649 vocabularies

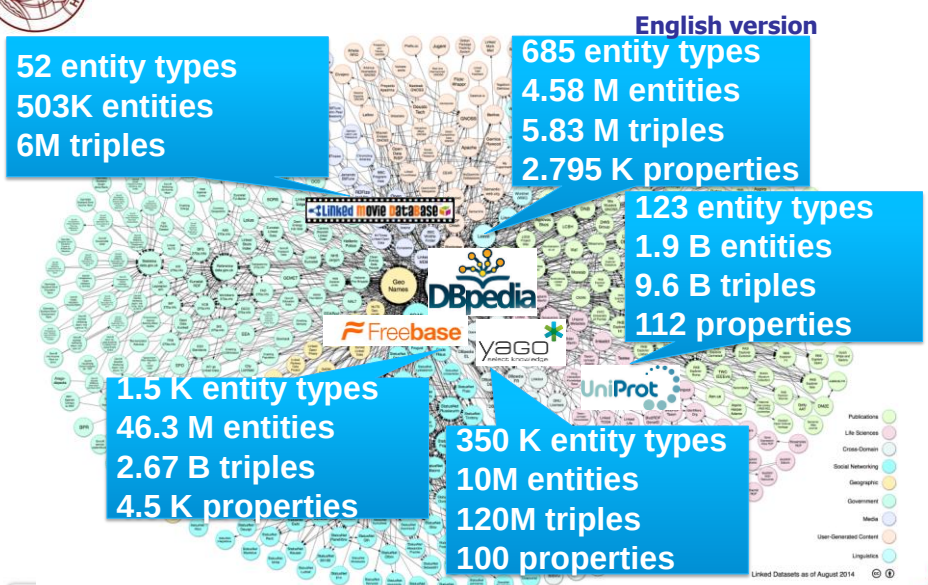


Last updated: 2014-08-30 http://lod-cloud.net/

LOD Cloud and Knowledge Bases (KB)

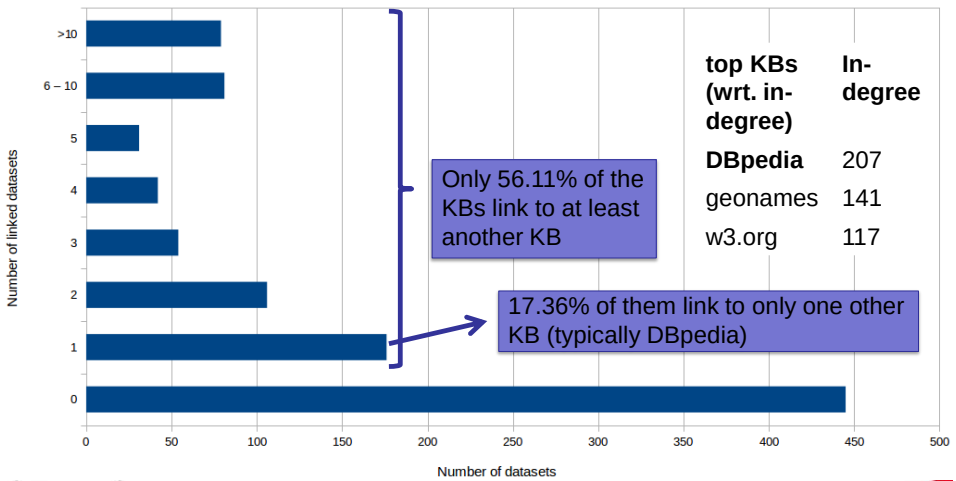


LOD KBs: Entity Statistics



LOD KBs: Entity Interlinking

The LOD cloud diagram is **sparse**



top KBs (wrt. in-degree)	In-degree
DBpedia	207
geonames	141
w3.org	117

Fall 2019

Describing and Linking Entities: Description Quality & Entity Resolution

Quality of Entity Descriptions in the Web of Data

- Given the open and decentralized nature of the Web, reliability and usability of entity descriptions need to be constantly improved
  - Incompleteness: real world entitles are only partially described in KBs
  - Redundancy: descriptions of the same real world entities usually overlap in multiple KBs
  - Inconsistency: real world entities may have conflicting descriptions across KBs
  - Incorrectness: errors can be propagated from one KB to the other due to manual copying or automated extraction/fusion techniques

Various Forms of KBs Overlapping

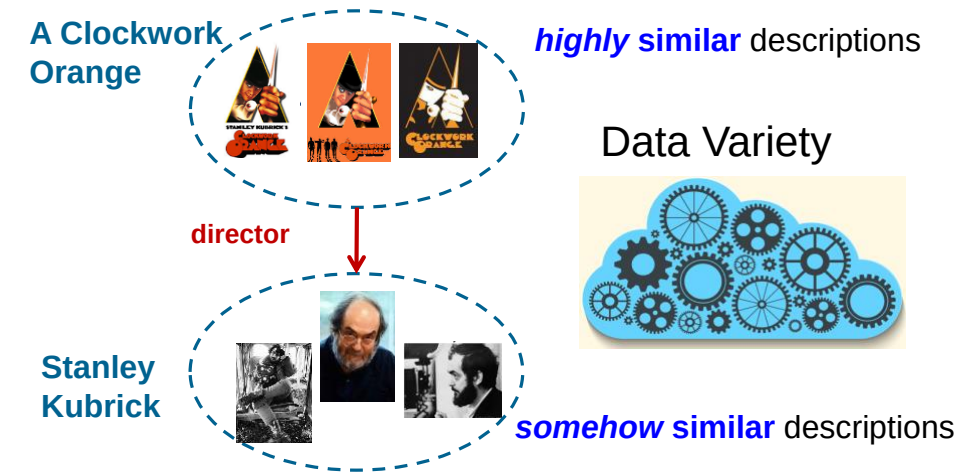
- among KBs (inter-overlapping, due to common data sources)

not identical descriptions, even if they stem from the same source

- within the same KB (intra-overlapping, due to data extraction or curation problems)
  - dbpedia:Robert_Soloway and dbpedia:Spam_King refer to the same individual, Robert Soloway, who was nicknamed the “Spam King”
  - less often than inter-overlaping

Entity Resolution (ER)

The problem of identifying descriptions of the same real-world entity



From Highly to Somehow Similar

Highly Similar

- feature many common tokens in the values of semantically related attributes
- heavily interlinked
  - mostly using *owl:sameAs* predicates
- usually provide complementary descriptions of the same entity in terms of facets, temporal evolution
  - good for fusing
- typically met in central KBs

Somehow Similar

- feature significantly fewer common tokens in attributes that are not always semantically related
- sparsely interlinked
  - using various kinds of predicates
- usually complementary descriptions of the same entity in terms of domains, aspects
  - good for linking
- typically met in peripheral KBs

Data Variety: Curse or Blessing ?

Types

Interpretation

How does ER Improves KB Quality?

- KB Completeness:
  - Linking somehow similar descriptions will increase coverage of entity facts and relationships
- KB Conciseness:
  - Merging highly similar descriptions will reduce duplicate entity facts and relationships
- KB Consistency:
  - Matching similar descriptions will enable to detect conflicting assertions
- KB Correctness:
  - Splitting complex descriptions will facilitate entity repairing

Web-Scale ER: Challenges

- The two core ER problems, how can we
  - ❶ *effectively* compute *entity similarity*
  - ❷ *efficiently* resolve *sets of entities within or across KBs*

  are challenged by the
  - important number of KBs (~ hundreds)
  - large number of entity types & properties (~ thousands)
  - massive volume of entities (~millions)
- Large-scale, progressive, multi-type, cross-domain ER
  - Big Data Volume, Velocity and Variety!

Entity Resolution – Formal Definition

- Entity resolution: The problem of identifying descriptions of the same entity within or across sources
  - $E = \{e_1, ..., e_m\}$ is a set of entity descriptions
  - $M : E \times E \rightarrow \{true, false\}$ is a match function
  - The resolution of entities in E results in a partition $P = \{p_1, ..., p_n\}$ of E, such that:
    1. $\forall e_i, e_j \in E : M(e_i, e_j) = true, \exists p_k \in P : e_i, e_j \in p_k$
    2. $\forall p_k \in P, \forall e_i, e_j \in p_k, M(e_i, e_j) = true$

each partition contains only matching descriptions

all the matching descriptions are in the same partition

Entity Resolution - Example

dbpedia:Stanley_Kubrick	
dbo:birthPlace	**dbpedia:Manhattan**
rdf:type	foaf:Person
rdf:type	yago:AmericanFilmDirectors
rdf:type	yago:AmateurChessPlayers

e1

e3

lmdb:director/8476	
lmdb:director_name	"Stanley Kubrick"
rdf:type	foaf:Person
foaf:made	lmdb:film/1894
foaf:made	lmdb:film/2014
foaf:made	lmdb:film/2685

lmdb:co0041067	
imbd:location	GeoNames:Buckinghamshire
imbd:filmography	"A Clockwork Orange"

PINEWOOD e5

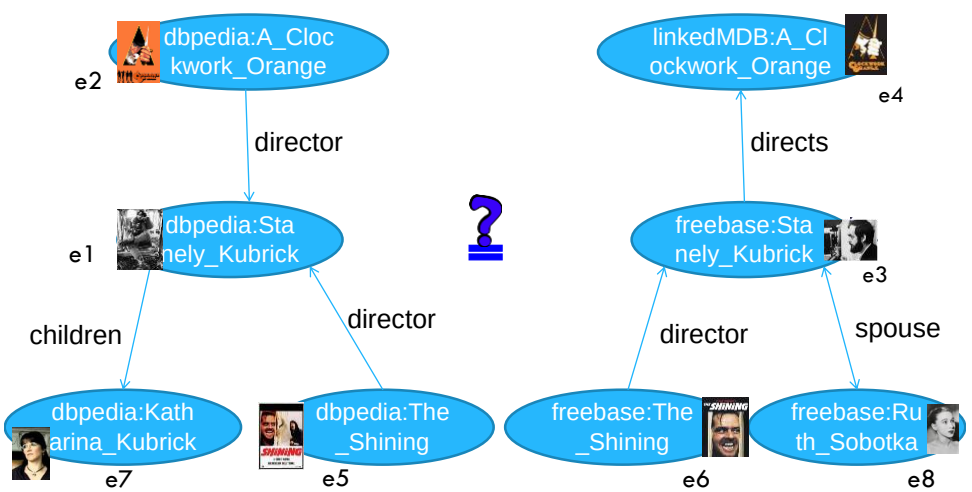
dbpedia: A_Clockwork_Orange_(film)	
dbo:director	**dbpedia:Stanley_Kubrick**
dbo:Work/runtime	"136"
foaf:name	"A Clockwork Orange"

e2

fbase:m.05ldxl	
fbase:film.directedBy	**lmdb:director/8476**
fbase:film_cut/runtime	"136"
foaf:name	"A Clockwork Orange"

e4

Matching Graph-Structured Entities



Single (Pairwise) Entity Matching Based on Content Similarity

Matching decisions are independent

e1	
birthPlace	**Manhattan**
type	Person
type	AmericanFilmDirectors
type	AmateurChessPlayers

sim_c(e1,e3) = Jaccard ({Manhattan, Person, AmericanFilmDirectors, AmateurChessPlayers}, {Stanley, Kubrick, Person, 1894, 2014, 2685}) = **0.1**

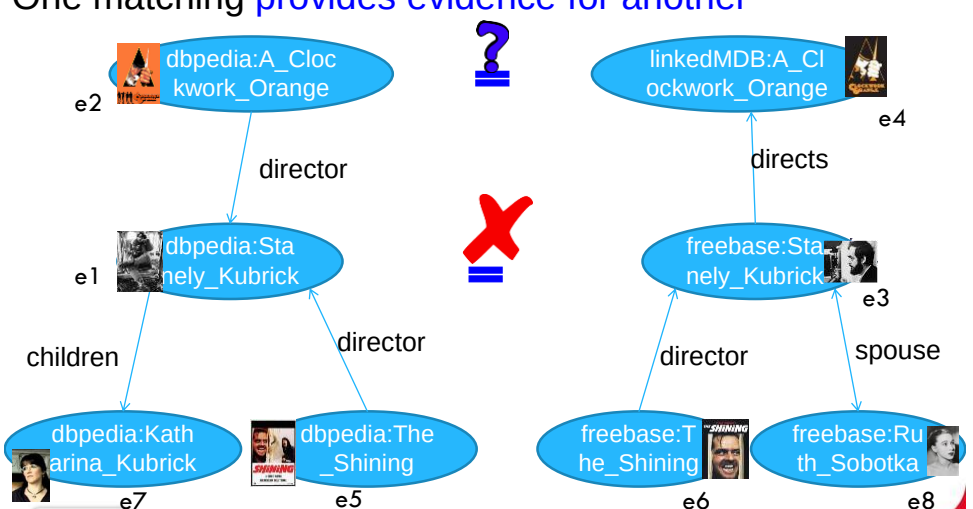
thresh = 0.5

e3	
director_name	"Stanley Kubrick"
type	Person
directs	film/1894
directs	film/2014
directs	film/2685

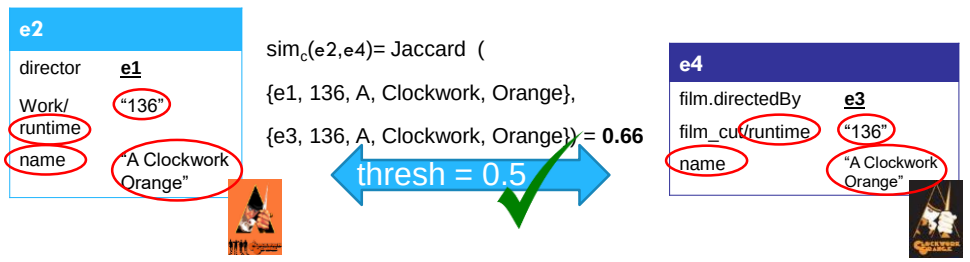
sim_c: let the content similarity of two descriptions be the Jaccard similarity of their values' token sets

Collective (Joint) Entity Matching Based on Structural Similarity

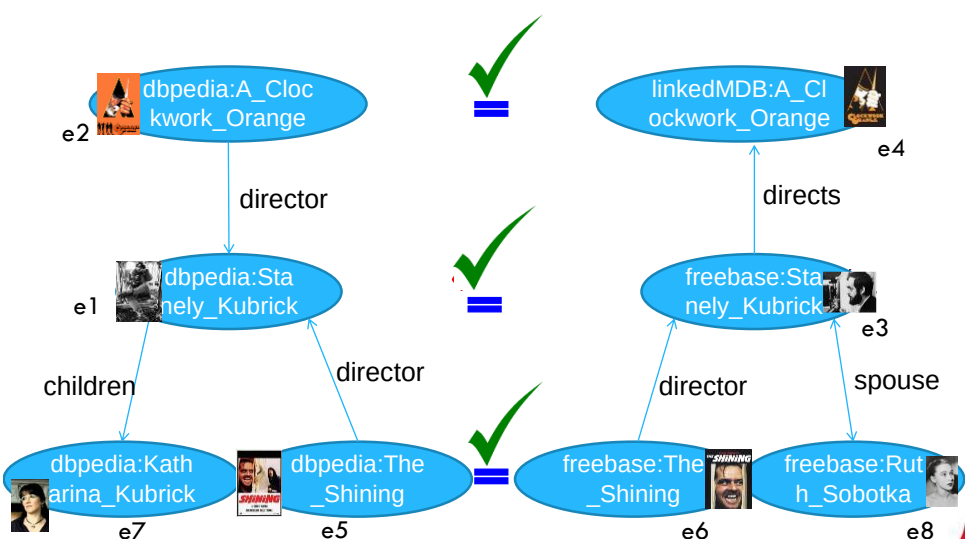
One matching provides evidence for another



Matching Neighborhood



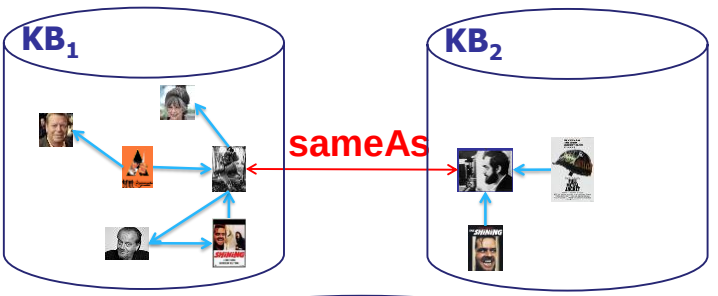
Similarity Propagation



Forms of Entity Resolution (ER)

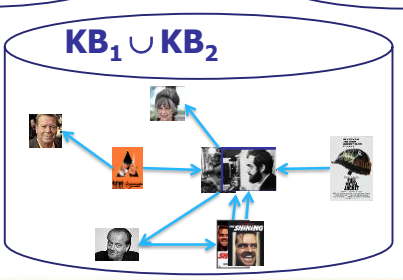
Record Linkage: ER without results merging

exploits exclusivity of matches

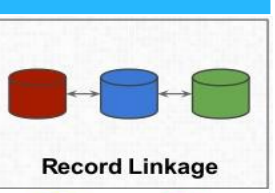
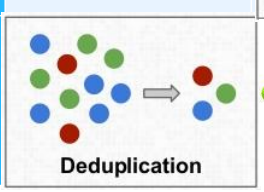


Record Deduplication: ER with results merging

exploits transitivity of matches



ER Forms & Similarity

	High similarity in structure	Low similarity in structure
High similarity in content		 Record Linkage
Low similarity in content	 Deduplication	

set sim. in the values of specific atts from **two** relations

string sim. in the values of specific atts from **one** relation

att & value sim. in a network of relations

ER Settings

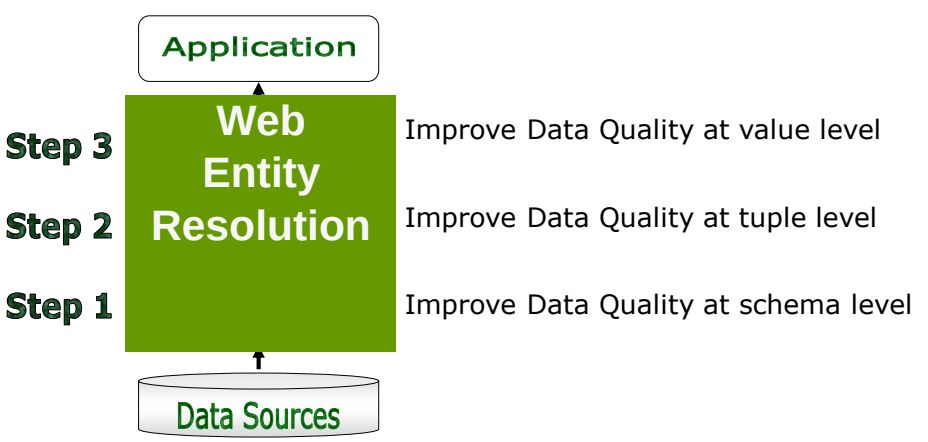
Two kinds of entity collections as input:

- Clean: redundant-free
- Dirty: contains redundant entity descriptions

An Entity Resolution (ER) task with input two entity collections can be:

- Clean-Clean ER: Given two clean, but overlapping entity collections, identify the common entity descriptions
  - a.k.a. the *record linkage* in databases
- Dirty-Dirty ER: Identify unique entity descriptions contained in the union of two dirty input entity collections
  - aka the *deduplication* problem in databases
- Dirty-Clean Entity Resolution

Comparison with Data Integration Tasks



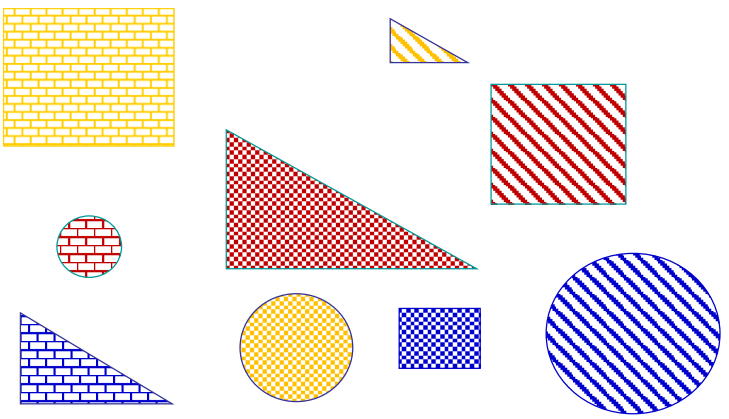
F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving inconsistencies at schema-,tuple-,and value-level. IEEE Data Engineering Bulletin, 29(2):21-31, 2006

Scope of the Seminar

- Describing and Linking Entities
  - Entity-Centric Applications & Knowledge Bases
  - Linked Knowledge Bases & The Web of Data
  - Description Quality & Entity Resolution
- Matching and Resolving Entities (I)
  - Entity Similarity (Content & Context)
  - Blocking Techniques (Token, Attribute & URI)
  - Block Post-Processing
- Matching and Resolving Entities (II)
  - Iterative Resolution Techniques (Merging & Matching)
  - Progressive Resolution Techniques
  - Conclusions & Open Issues

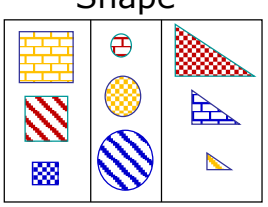
Matching and Resolving Entities (I): Entity Similarity

Similar or Dissimilar?

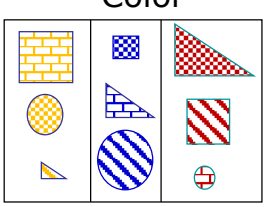


Similarity is Domain Specific

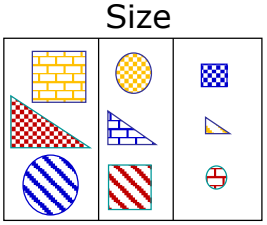
Shape



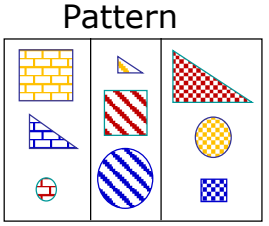
Color



Size



Pattern



Entity Resolution - Match

- Matches: Sets of entity descriptions that refer to the same real-world entity, intuitively:
  - Matching descriptions are placed in the same partition
  - All the descriptions of the same partition match

- A match function `M()` maps each pair of entity descriptions (e_i, e_j) to {true, false}
  - $M(e_i, e_j)$ = true => e_i, e_j are matches
  - $M(e_i, e_j)$ = false => e_i, e_j are non-matches

Match Function: Formal Properties

- The match function `M()` introduce an equivalence relation (`owl:sameAs`) among entity descriptions
  - Reflexivity: $\forall e_i \in E$ if $M(e_i, e_i)$ = true
  - Symmetry: $\forall e_i, e_j \in E$ $M(e_i, e_j) = M(e_j, e_i)$
  - Transitivity: $\forall e_i, e_j, e_k \in E$ if $M(e_i, e_j)$ = true and $M(e_j, e_k)$ = true then $M(e_i, e_k)$ = true

- Additional properties are useful when linking & repairing entities
  - Exclusivity (1-1 Assumption): $\forall e_i, e_k \in E$ and $\forall e_j \in E'$ if $M(e_i, e_j)$ = true then $M(e_k, e_j)$ = false
  - Functional Dependency: $\forall e_i, e_j, e_k, e_l \in E$ where $e_i \Rightarrow e_k$ and $e_j \Rightarrow e_l$ if $M(e_i, e_j)$ = true then $M(e_k, e_l)$ = true

Entity Resolution - Similarity

- In practice, the match function is defined via a similarity function **sim()**, measuring how similar two entity descriptions are to each other, according to certain comparison criteria
- Given a similarity threshold ϑ:
  - $M(e_i, e_j)$ = true, if $sim(e_i, e_j) \geq \vartheta$
  - $M(e_i, e_j)$ = false, otherwise
- ML techniques for automatically learning similarity measures are challenged by a Web-scale entity resolution [Köpcke et al. 2010]
  - adaptive learning techniques require training data for each domain [Tejada et al. 2002] [Bilenko et al. 2003]
  - active learning techniques (threshold-based Boolean functions or linear classifiers) work well with highly similar descriptions [Arasu et al. 2010]

Entity Similarity: Example

dbpedia: A_Clockwork_Orange_(film)	
dbo:director	dbpedia:Stanley_Kubrick
dbo:Work/runtime	"136"
foaf:name	"A Clockwork Orange"

e2

e4

fbase:m.05ldxl	
fbase:film.directedBy	lmdb:director/8476
fbase:film_cut.runtime	"136"
foaf:name	"A Clockwork Orange"

although not identical e_2 and e_4 are highly similar

dbpedia:Stanley_Kubrick	
dbo:birthPlace	dbpedia:Manhattan
rdf:type	foaf:Person
rdf:type	yago:AmericanFilmDirectors
rdf:type	yago:AmateurChessPlayers

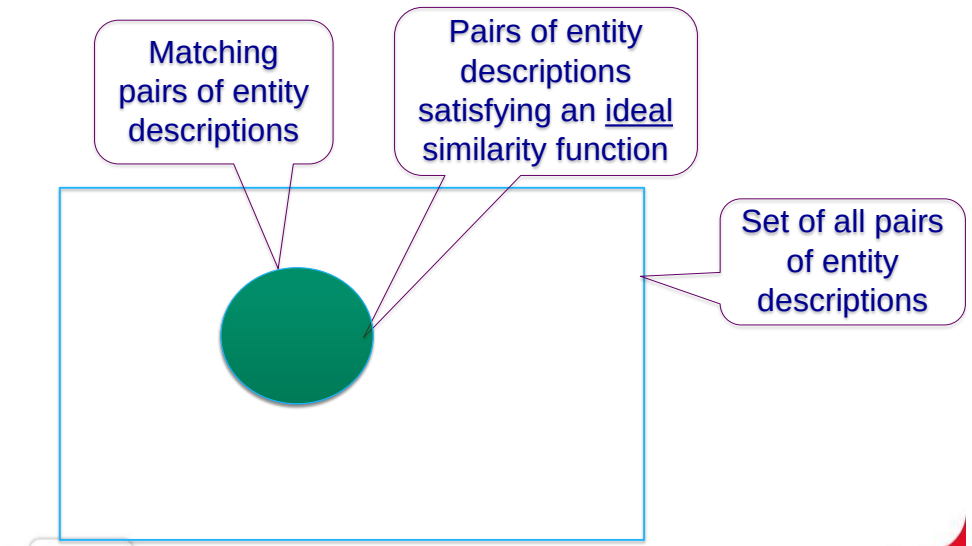
e1

e3

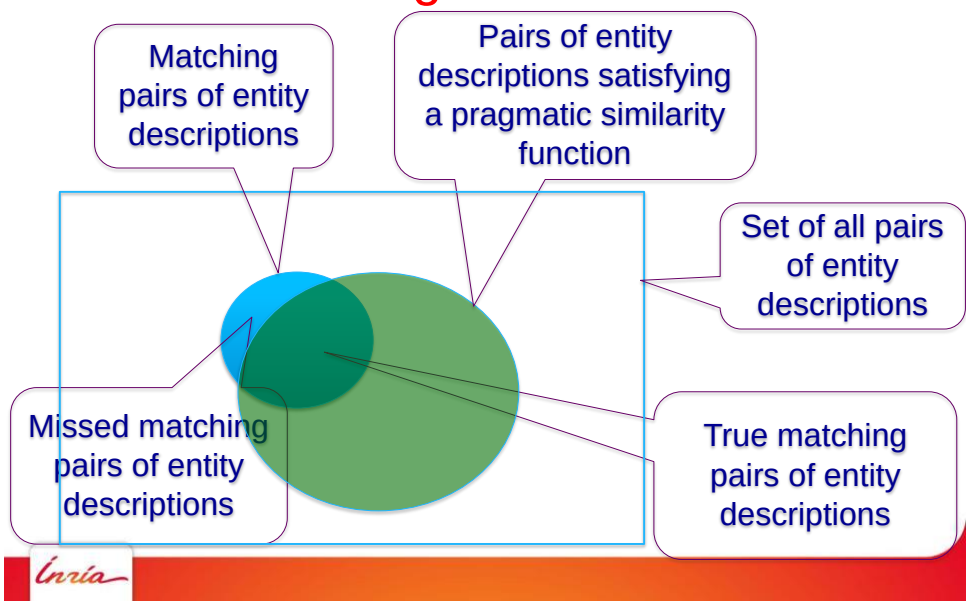
lmdb:director/8476	
lmdb:director_name	"Stanley Kubrick"
rdf:type	foaf:Person
foaf:made	lmdb:film/1894
foaf:made	lmdb:film/2014
foaf:made	lmdb:film/2685

e_1 and e_3 are at best somehow similar

The Role of Similarity Functions: Ideal Case



The Role of Similarity Functions: A Pragmatic Case

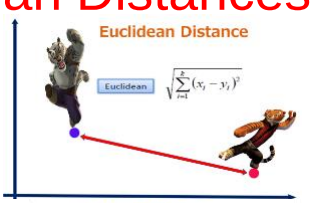


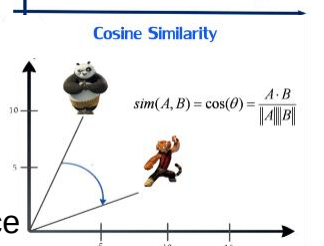
Entity Similarity: Formal Description

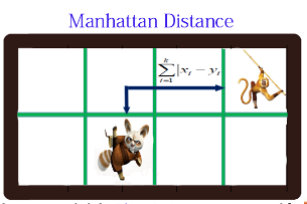
- A similarity function $sim(): E \times E \rightarrow R$ is a metric, if for any $e_i, e_j, e_l \in E$ for a given set E, it satisfies the following conditions:
  - $sim(e_i, e_i) \geq 0$ (*positive*),
  - $sim(e_i, e_i) \geq sim(e_i, e_j)$,
  - $sim(e_i, e_j) = sim(e_j, e_i)$ (*reflexive*),
  - $sim(e_i, e_i) = sim(e_j, e_j) = sim(e_j, e_i) \Rightarrow e_j = e_i$
  - $sim(e_i, e_j) + sim(e_j, e_l) \leq sim(e_i, e_l) + sim(e_j, e_j)$ (triangle inequality)
- Normalized similarity: $sim(e_i, e_i) \in [0,1]$
  - $sim(e_i, e_i) = 1$ for exact match
  - $sim(e_i, e_j) = 0$ for "completely" different e_i and e_i
  - $0 < sim(e_i, e_j) < 1$ for some approximate similarity
- Any similarity metric can be transformed to a distance metric and vice versa
  - $sim(e_i, e_j) = 1 - dist(e_i, e_j)$

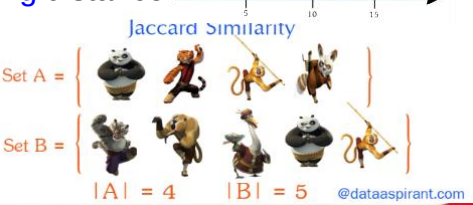
Euclidian vs Non-Euclidian Distances

- A Euclidean space: has some number of real-valued dimensions and dense points
  - There is notion of average of two points
  - A Euclidian distance is based on the locations of points in such space (L_2-norm, L_1-norm)
- A Non-Euclidian distance is based on properties of points, but not on their "location" in space
  - Jaccard, Cosine, Edit, Hamming distance









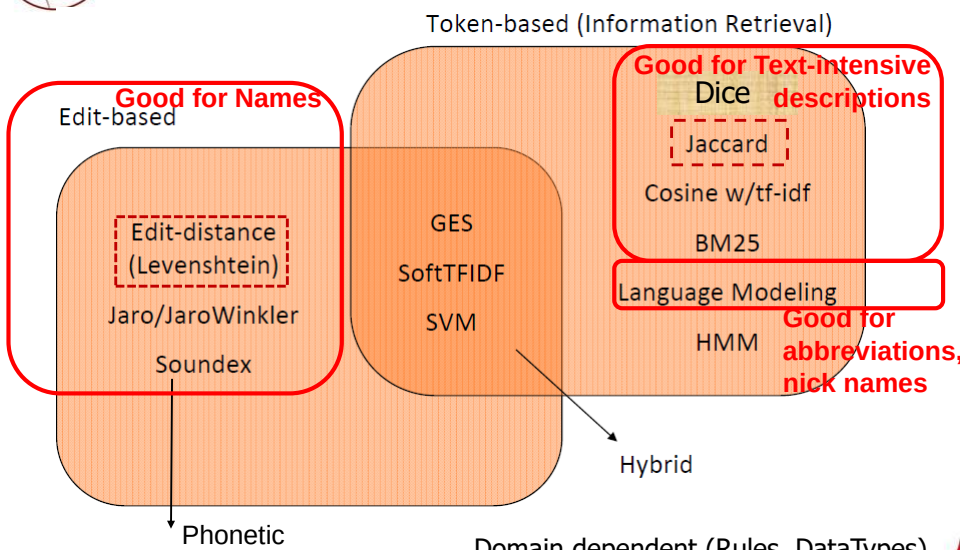
http://dataconomy.com/implementing-the-five-most-popular-similarity-measures-in-python/

Matching and Resolving Entities (I): Content-based Entity Similarity

In Search of Entity Similarity Measures

- Defining similarity functions that satisfy the formal properties of metric spaces is, in practice, too restrictive for non-geometric models
- Two main families of similarity measures for resolving entity descriptions in the Web of data
  - Content-based: mostly for measuring string similarity of attribute values in pairs of entity descriptions
    - character-based, token-based
  - Context-based: exploit similarity of neighbor descriptions via different entity relationships
    - tree-based, graph-based

String Similarity Measures



Tokenizing Strings: Separators

- Forming words from sequence of characters:
  - Surprisingly complex in English, can be harder in other languages
- General idea: Separate string into tokens using some separator
  - Space, hyphen, punctuation, special characters
  - Usually also convert to lower-case
- Problems:
  - Sometimes hyphens should be considered either as part of the word or a word separator: e.g., spanish-speaking
  - Apostrophes can be a part of a word, a part of a possessive, or just a mistake: e.g., master's degree
  - Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations: e.g., F.M.I

Tokenizing Strings: n-grams

- Split string into short substrings of length n
- Sliding window over string
  - n=2: Bigrams
  - n=3: Trigrams
- Variation: Pad with n – 1 special characters
  - Emphasizes beginning and end of string
- Variation: Include positional information to weight similarities
  - Number of n-grams = |x| – n + 1
  - Count how many n-grams are common in both strings

String	Bigrams	Padded bigrams	Positional bigrams	Trigrams
gail	ga, ai, il	⊙g, ga, ai, il, l⊗	(ga,1), (ai,2), (il,3)	gai, ail
gayle	ga, ay, yl, le	⊙g, ga, ay, yl, le, e⊗	(ga,1), (ay,2), (yl,3), (le,4)	gay, ayl, yle
peter	pe, et, te, er	⊙p, pe, et, te, er, r⊗	(pe,1), (et,2), (te,3), (er,4)	pet, ete, ter
pedro	pe, ed, dr, ro	⊙p, pe, ed, dr, ro, o⊗	(pe,1), (ed,2), (dr,3), (ro,4)	ped, edr, dro

Token-based Entity Similarity

name	Eiffel Tower
architect	Sauvestre
year	1889
location	Paris

e1

name	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e2

about	Lady liberty
architect	Eiffel
location	NY

e3

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e4

name	White Tower
location	Thessaloniki
year-constructed	1450

e5

$$Jaccard(tokens(e_i), tokens(e_j)) = \frac{|tokens(e_i) \cap tokens(e_j)|}{|tokens(e_i) \cup tokens(e_j)|}$$

$Jaccard(e_1,e_3) = 1/8$
$Jaccard(e_1,e_4) = 1$
$Jaccard(e_1,e_5) = 1/8$

$Jaccard(e_2,e_3) = 3/7$
$Jaccard(e_2,e_4) = 1/11$
$Jaccard(e_2,e_5) = 0/11$

Matching and Resolving Entities (I): Context-based Entity Similarity

Similarity of Hierarchical Data

- Relational star / snowflake schema [Ananthakrishna et al. 2002]

Casting

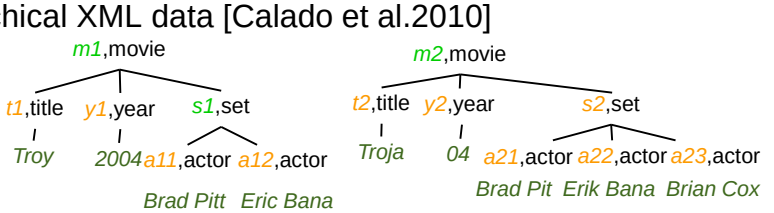
ID	Actor	Film
S1	Al Pacino	F1
S2	Al Pacino	F2
S3	Marlon Brando	F2

Movie

ID	Name	Year	Rating
F1	The Godfather	1972	9.2
F2	Gottvatter, The	72	

single branch

- Hierarchical XML data [Calado et al.2010]



multiple branch

DELPHI Containment Metric [Ananthakrishna et al. 2002]

- Hybrid measure considering both similarity of attribute values (*tcm*) and similarity of children sets reached by foreign keys (*fkcm*)
- Similarity of attribute values
  - Divide tuples into tokens → token sets *TS*
  - Compute the edit distance between the tokens of different sets
  - Determine weight of each token via inverse document frequency
    - Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low
  - The token similarity metric *tcm* measures which fraction of one tuple *T* is covered by the other tuple *T'* in a relation of interest R_i

$$tcm(T,T') = \frac{\sum idf(TS(T) \cap TS(T'))}{\sum idf(TS(T))} \qquad \mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}.$$

DELPHI Containment Metric [Ananthakrishna et al. 2002]

- Similarity of children sets
  - The children set ***CS*** of a tuple ***T*** includes all tuples of a relation R_j referencing ***T*** from a relation R_i by means of a foreign key
  - Foreign-key containment metric (*fkcm*) measures at what extent the children set of a tuple *T* is covered by the children set of a tuple *T'*

$$fkcm(T,T') = \frac{|CS(T) \cap CS(T')|}{|CS(T)|}$$

DELPHI Containment Metric [Ananthakrishna et al. 2002]

- Combining ***tcm*** and ***fkcm***
  - Both ***tcm*** and ***fkcm*** are assigned an IDF weight
  - Use of a classification function:

```
pos(x) = 1   if x > 0,
         -1  otherwise
```

  - Threshold for ***tcm***: *s1*
  - *Threshold for **fkcm**: s2*
  - Classification of pairwise comparison between ***T*** and ***T'*** using

$$pos(IDF(TS) * pos(tcm(T,T') - s1) + IDF(CS) * pos(fkcm(T,T') - s2))$$

    - If final result equals 1, then match, otherwise non-match

DELPHI Containment Metric: Example

ID	Actor	Film
S1	Al Pacino	F1
S2	Al Pacino	F2
S3	Marlon Brando	F2

ID	Name	Year	Rating
F1	The Godfather	1972	9.2
F2	Gottvatter, The	72	

1. Token sets:

TS(F1) = {The, Godfather, 1972, 9.2}

TS(F2) = {Gottvatter, The, 72}

2. Attribute value similarities

The = The, Godfather = Gottvatter, 1972 = 72.

3. Weights

For simplification, we assume all tokens have equal weight

4. Token containment metric

tcm(F1,F2) = ¾, tcm(F2,F1) = 1

5. Children co-occurrence

fkcm(F1,F2) = 1, *fkcm(F2,F1)* = ½

6. **Combination of both metrics**
(*s1* = *s2* = 0.5, weights = 1)

pos(pos(3/4 - 0.5) + pos(1 - 0.5) = 1

→ F1 and F2 match

Graph-Context Similarity

- Assign a score $s_{x\,y}$ to each pair of entities x and y
  - two descriptions are similar if their structural neighborhood is similar (hide attribute values)
- Local similarity Indices
  - Common neighbors (CN)
    - $s_{xy}^{CN} = |\Gamma(x) \cap \Gamma(y)|$ where $\Gamma(x)$ is the set of neighbors
  - Jaccard: Normalized common neighbors (NCN)
    - $s_{xy}^{NCN} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$ where $\Gamma(x)$ is the set of neighbors
  - Adamic-Adar (AA):
    - $s_{xy}^{AA} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log k_z}$ where k_z is the degree of node z and $\frac{1}{\log k_z}$ counts the uniqueness of z, which is inversely proportional to its number of neighbors

Content & Context Similarity: Linda [Böhm et al. 2012]

- Works on an entity graph constructed from RDF triples having URIs as subject, predicate and object
  - Literals are stored for each entity e as L(e)
- Matches are identified using a hybrid similarity:
  - String similarity (token-based) of their literal values L(e)
  - Contextual similarity (based on in and out neighbors in the entity graph)
- The context $C(n)$ of e is a set of tuples (p_i, e_i, w_i), where
  - e_i is a neighboring node of e
  - p_i is the label of the relationship between e and e_i
  - w_i is a numeric weight selected to be higher for less frequent and thus the most discriminative context information

Contextual Similarity

The contextual similarity of nodes n and m is:

context_sim(n, m) =

- $\sum_{(p_i,z_i,w_i)\in C(n)} \max_{(p_j,z_j,w_j)\in C(m)} w_i \cdot x_{z_i,z_j} \cdot sim(p_i,p_j), if\ |C(n)| \leq |C(m)|$
- $\sum_{(p_j,z_j,w_j)\in C(m)} \max_{(p_i,z_i,w_i)\in C(n)} w_j \cdot x_{z_i,z_j} \cdot sim(p_i,p_j), else$

where

$x_{n,m}$ is 1, if n, m are identified as matches, and 0 else and $sim(p_i, p_j)$ is the string similarity of the predicates of n, m (edit-distance based)

- It counts the number of common or matching neighbors of two descriptions, which are linked to them in a similar way, i.e., using a relationship with a similar name

LINDA Hybrid Similarity

- The similarity score for descriptions e and e' is:

$sim^{LINDA}(e,e') = content\_sim(e,e') + \beta * context\_sim(C(e),C(e')) - \theta$

where

β controls the contextual influence

θ is used for re-normalization to values around 0

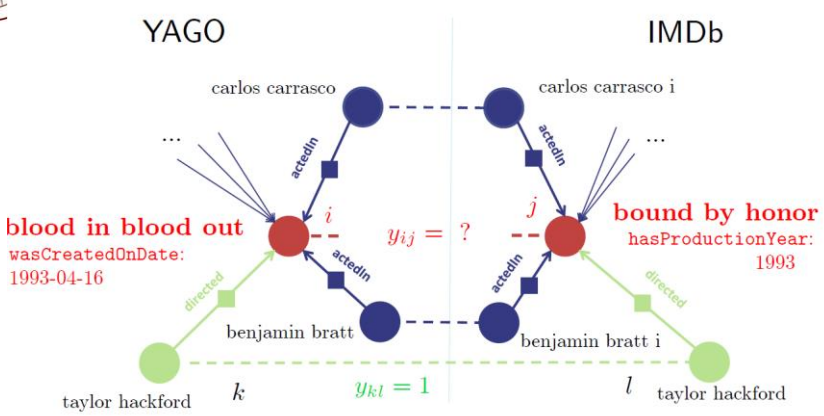
content_$sim_0(n,m) = \frac{|N_n \cap N_m|}{\min(|N_n|,|N_m|) + \ln(||N_n| - |N_m|| + 1)}$

- sim^{LINDA} is not a normalized measure as it serves to rank pairs of descriptions based on the evidence that they are matching
  - positive scores reflect likely mappings
  - negative scores imply dissimilarities

Content & Structure Similarity: SiGMa [Lacoste-Julien et all 2013]

- Even though entities `i` and `j` have no tokens in common, the fact that several of their respective neighbors are matched together is a strong evidence that `i` and `j` should be matched together
  - Use neighbors for scoring and suggesting candidate pairs

SiGMa Neighbors Similarity



- Compatible-neighbors N_{ij}: a neighbor `k` of `i` being matched to a *compatible* neighbor `l` of `j` should encourage `i` to be matched to `j`
  - N_{ij} = {(k, l): (i, r, k)∈KB1 and (j, s, l)∈KB2 and relationship r is matched to s}

SiGMa Similarity Measures

- Content similarity: *static* score of both the string representation of entities (`rdfs:label`) and their other property values

$$s_{ij} = (1-\beta)\texttt{string}(i,j) + \beta\texttt{prop}(i,j) \quad \beta \in [0,1]$$

- Context-dependent similarity: *dynamic* score where the weight $w_{ij,kl}$ is the contribution of a neighboring matched pair `(k,l)` to the score of the candidate pair `(i,j)`

$$\delta g_{ij}(y) \doteq \sum_{(k,l)\in\mathcal{N}_{ij}} y_{kl}\,(w_{ij,kl} + w_{kl,ij})$$

  - count the number of compatible neighbors currently matched together for a pair of candidates

$$g_{ij}(y) = \sum_{(k,l)\in\mathcal{N}_{ij}} y_{kl}(\gamma_i w_{ik} + \gamma_j w_{jl})$$

- Global Score: $\texttt{score}(i,j;y) = (1-\alpha)s_{ij} + \alpha\,\delta g_{ij}(y)$

In Search of Entity Similarity Measures

- For highly similar entities content similarity (i.e., their attribute values) is sufficient
- For somehow similar entities we need to also consider the similarity of the structured context of entities in an iterative way
  - Identifying *most discriminating* attributes and relationships is helpful
- An orthogonal issue is the schematic discrepancy of attributes and relationships employed in the entity descriptions whose hybrid similarity is assessed
  - Simple: either ontological commitments exist or schematic mappings are provided by the users
  - Complex: assess similarity of attributes and relationships based on the similarity of their names or values

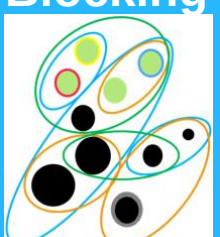
Matching and Resolving Entities (I): Blocking Techniques

Typical ER Workflow

Reduce comparisons leading to non-matching entities

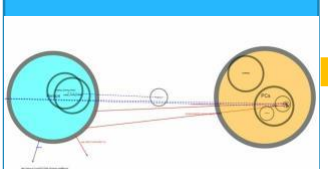
O(n)
if hash-based

Blocking



O(n²)
in blocks of descriptions (avg)

Matching



Gr… of
en… …criptions

Preliminary experiment over 9M entity descriptions in a cluster of 15 VMs:

ER workflow without blocking: >200 hrs

ER workflow with blocking: 11 hrs

Token Blocking [Papadakis et al. 2011]

- Assume two clean sets KB_1, KB_2 of entity descriptions free of intra-overlapping (*Clean-Clean ER)*
- Each distinct token t_i of values of entity descriptions in $KB_1 \cup KB_2$ corresponds to a block
  - Each block contains all entity descriptions sharing the corresponding token
  - Pairs originating from the same (clean) KB are not compared
- Token blocking offers a brute-force method for comparing descriptions even if they are highly heterogeneous
  - The same pair of descriptions is contained in many blocks (redundant comparisons)
  - Many dissimilar pairs are put in the same block (unnecessary comparisons)

Token Blocking Example

name	Eiffel Tower
architect	Sauvestre
year	1889
location	Paris e1

name	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY e2

about	Lady liberty
architect	Eiffel
location	NY e3

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris e4

name	White Tower
location	Thessaloniki
year-constructed	1450 e5

Actually, an **inverted index**

Generated Blocks

Eiffel	Tower	Statue	Liberty	White	1889	Bartholdi
e_1, e_2, e_3, e_4	e_1, e_4, e_5	e_2	e_2, e_3	e_5	e_1, e_4	e_2

NY	Paris	1886	1450	Lady	Sauvestre	Thessaloniki
e_2, e_3	e_1, e_4	e_2	e_5	e_3	e_1, e_4	e_5

Redundant comparisons for (e_1, e_4) contained in 5 different blocks!

Token Blocking Example

name	Eiffel Tower
architect	Sauvestre
year	1889
location	Paris

e1

name	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e2

about	Lady liberty
architect	Eiffel
location	NY

e3

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e4

name	White Tower
location	Thessaloniki
year-constructed	1450

e5

Actually, an **inverted index**

Generated Blocks

Eiffel	Tower	Liberty	1889
e_1, e_2, e_3, e_4	e_1, e_4, e_5	e_2, e_3	e_1, e_4

NY	Paris	Sauvestre
e_2, e_3	e_1, e_4	e_1, e_4

Unnecessary comparisons between (e_1, e_3), (e_2, e_4), (e_1, e_5)

Attribute Clustering Blocking [Papadakis et al. 2013]

- Token blocking totally ignores the semantics of attributes
  - When attribute mappings are not known, attribute clustering considers similarity of attributes
  - Attribute similarity is computed w.r.t. the string similarities of their values
- Two main steps:
  - Similar attributes are placed together in non-overlapping clusters
  - Token blocking is performed on the descriptions of each cluster

Attribute Clustering Blocking: Example

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e11

work	Lady Liberty
artist	Bartholdi
location	NY

e15

about	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e12

about	Auguste Bartholdi
born	1834

e13

work	Eiffel Tower
year-constructed	1889
location	Paris

e16

about	Joan Tower
born	1938

e14

work	Bartholdi Fountain
year-constructed	1876
location	Washington D.C.

e17

Finding the attribute of KB_2 that is the most similar to "about" of KB_1:
values {Eiffel, Tower, Statue, Liberty, Auguste, Bartholdi, Joan}
compared to (with Jaccard similarity) :
values of **work**: {Lady,Liberty,Eiffel,Tower,Bartholdi,Fountain} → **Jaccard = 4/9**
values of artist: {Bartholdi} → Jaccard = 1/8
values of location: {NY, Paris, Washington, D.C.} → Jaccard = 0
values of year-constructed: {1889, 1876} → Jaccard = 0

Attribute Clustering Blocking: Example

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e11

work	Lady Liberty
artist	Bartholdi
location	NY

e15

about	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e12

about	Auguste Bartholdi
born	1834

e13

work	Eiffel Tower
year-constructed	1889
location	Paris

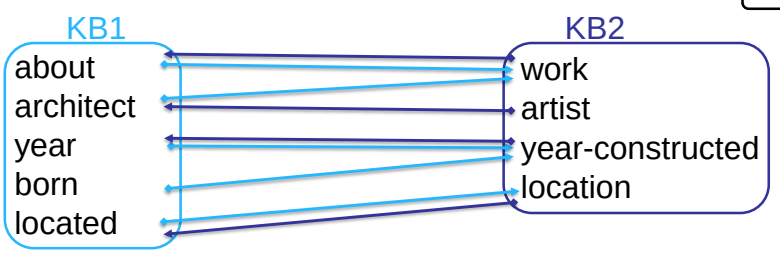
e16

about	Joan Tower
born	1938

e14

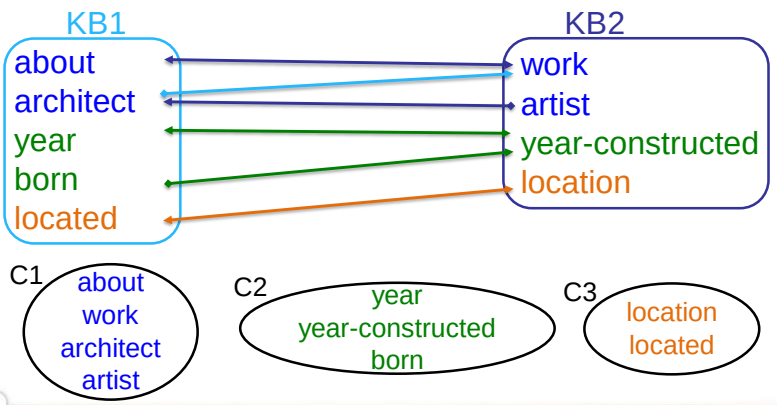
work	Bartholdi Fountain
year-constructed	1876
location	Washington D.C.

e17



Attribute Clustering Blocking: Example

- Compute the transitive closure of the generated attribute pairs
  - Connected attributes form clusters
- Example: Pairs (about, work), (work, about), (artist, architect), (architect, work)



Token Blocking in Each Attribute Cluster

about	Eiffel Tower
architect	Sauvestre
year	1889
located	Paris

e11

about	Statue of Liberty
architect	Bartholdi Eiffel
year	1886
located	NY

e12

about	Auguste Bartholdi
born	1834

e13

about	Joan Tower
born	1938

e14

work	Lady Liberty
artist	Bartholdi
location	NY

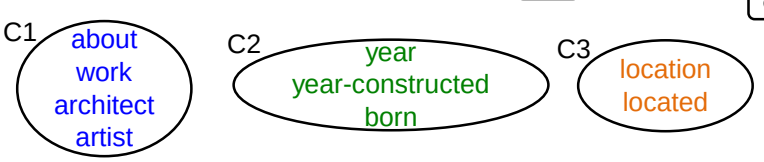
e15

work	Eiffel Tower
year-constructed	1889
location	Paris

e16

work	Bartholdi Fountain
year-constructed	1876
location	Washington D.C.

e17



C3.NY	C1.Tower	C1.Bartholdi
e_{12}, e_{15}	e_{11}, e_{14}, e_{16}	e_{12}, e_{13}, e_{15}, e_{17}

→ *compare Lady Liberty to Auguste Bartholdi*

Prefix-Infix(-Suffix) [Papadakis et al.12]

- Not only the content but also the identity (URI) of descriptions reveal valuable semantics regarding similarity of entities
  - E.g., 66% of the 182 million URIs of BTC09 follow the scheme: Prefix-Infix(-Suffix)
- Example: http://en.wikipedia.org/wiki/Linked_data#Principles.html
  - Prefix describes the source, i.e. domain, of the URI
  - Infix is a local identifier or a named anchor
  - The optional Suffix contains details about the format, e.g. .html, .rdf .nt, etc.
- Infix blocking
  - The blocking key is the URI infix of the entity description
- Infix profile blocking
  - The blocking keys are the URI infixes of attribute values in entity descriptions

Infix Blocking

yago:Statue_of_Liberty

skos:prefLabel	Statue of Liberty
yago:isLocatedIn	yago:Liberty_Island

e1

dbpedia:Statue_of_Liberty

rdfs:label	Statue of Liberty
dbprop:location	dbpedia:Liberty_Island

e2

fb:m.072p8

fb:official_name	Statue of Liberty
fb:contained_by	fb:m.026kp2
ex:location	ex:Liberty_Island

e3

geonames:5139572

geonames:name	Statue of Liberty
geonames:nearby	geonames:5124330

e4

yago:Tina_Brown

skos:prefLabel	Tina Brown
yago:linksTo	yago:Liberty_Island

e5

Statue_of_Liberty	m.072p8	5139572	Tina_Brown
e_1, e_2	e_3	e_4	e_5

Infix Profile Blocking

yago:Statue_of_Liberty

skos:preFLabel	Statue of Liberty
yago:isLocatedIn	yago:Liberty_Island

e1

dbpedia:Statue_of_Liberty

rdfs:label	Statue of Liberty
dbprop:location	dbpedia:Liberty_Island

e2

fb:m.072p8

fb:official_name	Statue of Liberty
fb:contained_by	fb:m.026kp2
ex:location	ex:Liberty_Island

e3

geonames:5139572

geonames:name	Statue of Liberty
geonames:nearby	geonames:5124330

e4

yago:Tina_Brown

skos:prefLabel	Tina Brown
yago:linksTo	yago:Liberty_Island

e5

(e_1, e_3) true positive
(e_1, e_5) false positive

Liberty_Island	m.026kp2	5124330
e_1, e_2, e_3, e_5	e_3	e_4

- The effectiveness of these approaches relies on the good naming practices of the KBs publishing entity descriptions

Other Blocking Techniques

- Similarity Join algorithms: relies on an inverted index from the most discriminating (non-frequent) tokens of the descriptions and the prefix filtering principle [Chaudhuri et al., 2006]

$$Jaccard(x, y) \geq t \Rightarrow |x \cap y| \geq \frac{t}{1+t} \cdot (|x| + |y|)$$

  - prefix filtering is effective only when the similarity threshold is extremely high
- Frequent itemsets blocking: build blocks for sets of tokens that frequently co-occur in entity descriptions
  - may significantly reduce the number of candidate pairs
  - may also significantly increase missed matches, especially between descriptions with few common tokens
- Multidimensional blocking: constructs a collection of blocks for each similarity function used to resolve entities and aggregates them into a single multidimensional collection, by taking into account the similarities of descriptions that share blocks

Placing Entities in the Same Block

Method	Criterion
Token Blocking [Papadakis et al., 2011]	The descriptions have a common token in their values
Attribute Clustering Blocking [Papadakis et al., 2013]	The descriptions have a common token in the values of attributes that have similar values in overall
Prefix-Infix(-Suffix) [Papadakis et al., 2012]	The descriptions have a common token in their literal values, or a common URI infix
ppjoin+ [Vernica et al., 2010]	The descriptions have a common token in their p first tokens (sorted in ascending frequency order)
Frequent itemsets [Kenig and Gal, 2013]	The descriptions have frequently co-occurring tokens in their values

Blocking Algorithms Comparison

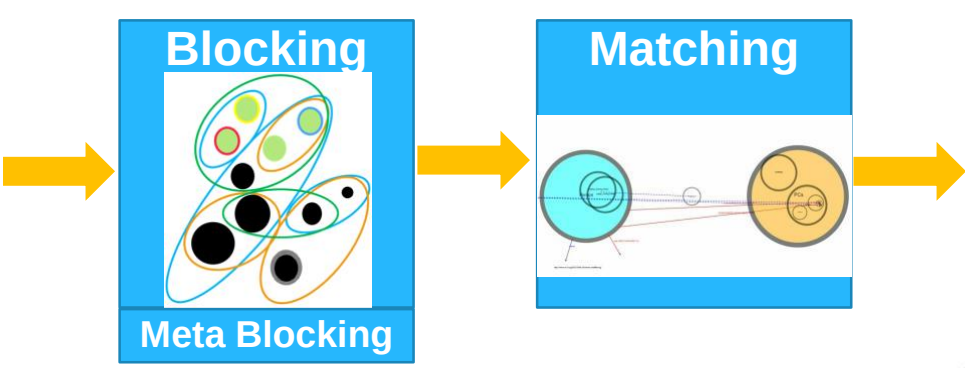
Method (suffix)	Partitioning	Overlapping			Algorithm	
		positive	negative	neutral	Hash-based	Sort-Based
Canopy Clustering[McCallum et al. 00]			•			•
Standard blocking [Fellegi& Sunter 69, Kolb et al.,12b]	•				•	
Q-grams [Gravano et al., 01]		•			•	
Suffixes [Aizawa & Oyama, 05]		•			•	
Sorted neighborhood [Hernàndez & Stolfo 95, Kolb et al.,12a]				•		•
Adaptive sorted neighborhood [Yan et al., 07]	•					•
Token blocking [Papadakis et al.,11]		•			•	
Attribute clustering [Papadakis et al.,13]		•			•	
Prefix-infix (suffix) [Papadakis et al. 12]		•			•	
ppjoin+ [Vernica et al., 10]		•			•	
Frequent itemsets [Kenig&Gal,13]		•			•	

Schema Agnostic

Matching and Resolving Entities (I): Block Post-Processing

Meta-blocking: Improving the Efficiency of Blocking

Eliminate redundant comparisons & reduce unnecessary comparisons



Meta-blocking – The Blocking Graph

Blocks (ToB):

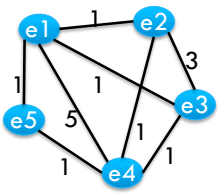
Eiffel	Tower	Liberty
e_1, e_4, e_2, e_3	e_1, e_4, e_5	e_2, e_3

NY	1889	Paris
e_2, e_3	e_1, e_4	e_1, e_4

Sauvestre
e_1, e_4

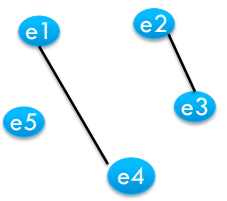
14 comparisons to identify 2 matches e_1-e_4 and e_2-e_3

Blocking graph (Nodes: entity descriptions, Edges: common block)



edge weights = #common blocks

Pruned blocking graph (discard edges with weight below avg.: 1.75)



2 comparisons to identify 2 matches

Prune edges to discard unnecessary comparisons between non-matches based on positive overlapping evidence

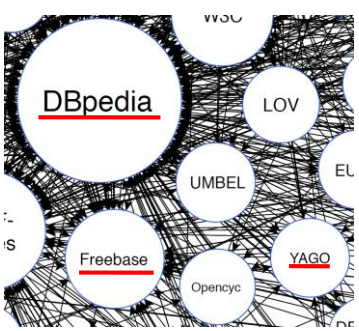
Edge Weighting and Pruning

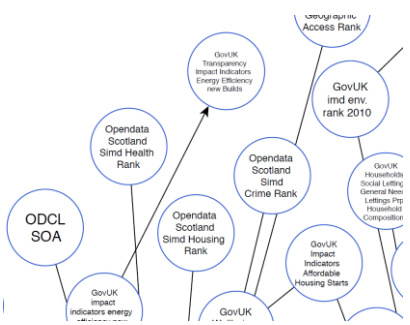
- Weighting Schemes (how to weight the edges)
  - **Common Blocks (CBS)**: $w_{i,j} = |B_{i,j}|$
  - **Jaccard (JS)**: $w_{i,j} = |B_{i,j}|/(|B_i|+|B_j|-|B_{i,j}|)$
  - **Enhanced CBS (ECBS)**: $w_{i,j}$ = CBS • log(|B|/|Bi|) • log (|B|/|Bj|)

- Pruning Methods (which edges to prune)
  - **WEP**: Keep edges with weight above average
  - **CEP**: Keep top-K edges overall
  - **WNP**: Keep, for each node, the edges with weight above a local average
  - **CNP**: Keep, for each node, its top-K edges

Matching and Resolving Entities (II): Iterative Resolution Techniques

Central vs. Peripheral KBs



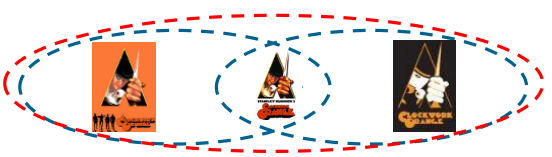


In Search of Similarity Evidence in KBs [Efthymiou et al. 2016, 2015]

- Attribute-based comparisons
  - unique attributes (e.g., `rdfs:label`) provide strong evidence
    - >90% of matching pairs have >80% overlap similarity in the values of `rdfs:label`
- Content-based comparisons
  - central KBs: 3-4 common tokens in entity values
  - peripheral KBs: 1-2 common tokens in entity values
    - blocking algorithms miss up to 30% matches in peripheral KBs
- Relationships-based comparisons
  - matching neighbors provide positive evidence
    - >92% of pairs with at least one matching neighbor, are matches in most KBs
  - some types of relationships provide strong negative evidence
    - dissimilar values for `wasBornIn` indicate a non-matching pair

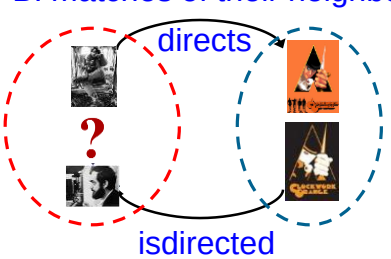
Types of Missed Matches (FNs)

- Type A: a third, matching description (transitivity)



applicable to identify matches within a KB

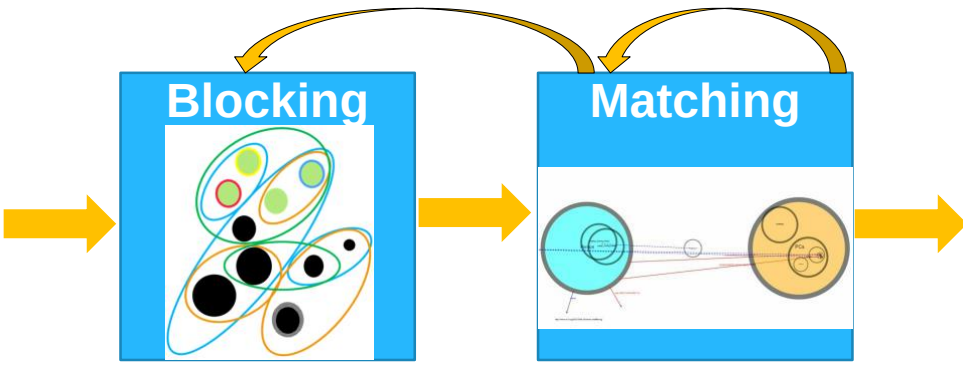
- Type B: matches of their neighbors



can identify matches both within a KB and across different KBs

Iterative ER

Increase the number of matching entities



Iterative ER: identify new matches based on partial results either of matches or of merges

- Good for high Variety

Iterative ER Approaches

- Merging-based: new matches can be found by exploiting merged (more complete) descriptions of previously identified matches
  - Idea: ER resembles a database self-join operation (of the initial set of descriptions with itself)
    - But: No knowledge about which descriptions may match, so all pairs of descriptions need to be compared

- Matching-based: If descriptions related to entity e_i are matching to descriptions related to e_j, then e_i and e_j are likely to match
  - Idea: ER resembles to a graph traversal problem in which similarity is propagated until a fixed point is reached
    - Use positive or negative evidence for prioritize similarity re-computation

Matching and Resolving Entities (II): Merging-based Iterative Resolution

Merging-based ER– Formal Definition

- Let E = {e_1, ..., e_m} a set of entity descriptions and
  - M : E ×E → {true, false} is a match function
  - μ : E ×E → E is a partial merge function
- The merging-based (generic) resolution of entities in E is a set of descriptions E', such that:
  1. $\forall e_i, e_j \in E$: M(e_i, e_j) = true, $\exists e_k \in E'$: μ (e_i, e_j) ≤ e_k
  2. $\forall e_k \in E'$, $\forall e_l \in E$, $e_l \leq e_k$
  3. no strict subset of E' satisfies conditions 1 and 2

  where $e_l \leq e_k$ means that e_k is at least as informative than e_l, regarding the same real-world entity
- Note: $E' \subseteq \hat{E}$ (the merge closure)
  - Condition 1: E' cannot produce more than $\hat{E}$
  - Condition 2: E' produce at least all information of $\hat{E}$
  - Condition 3: Minimal solution

Match and Merge Functions: ICAR Properties [Benjelloun et al., 2009]

- Idempotence: a description always matches itself, and merging it with itself still yields the same description
  - $\forall e_i \in E$ if $M(e_i,e_i)$ = true then $\mu(e_i,e_i) = e_i$
- Commutativity: direction of match and merge is irrelevant
  - $\forall e_i,e_j \in E$ if $M(e_i,e_j) = M(e_j,e_i)$ = true then $\mu(e_i,e_j) = \mu(e_j,e_i)$
- Associativity: order of merge is irrelevant
  - $\forall e_i,e_j,e_k \in E$ if $\mu(\mu(e_i,e_j),e_k)$ and $\mu(e_i,\mu(e_j,e_k)$ exist then $\mu(\mu(e_i,e_j),e_k) = \mu(e_i,\mu(e_j,e_k)$
- Representativity: merging does not lose matches; no "negative evidence"
  - if $e_k=\mu(e_i,e_j)$ then $\forall e_l \in E$ such that $M(e_i,e_l)$ = true also $M(e_k,e_l)$ = true
- Transitivity is not assumed!

Merge Domination & Monotonicity

- When the match and merge functions satisfy the ICAR properties, there is a natural domination (partial) order of descriptions
- Given two descriptions, e_1 and e_2, we say that e_1 is merge dominated by e_2, denoted $e_1 \leq e_2$, if $M(e_1,e_2)$ = true and $\mu(e_1,e_2) = e_2$
  - e_1 does not add information
- Merged descriptions always dominates the ones they have derived from
  - $\forall e_1, e_2 \in E$ such that $M(e_1,e_2)$ = true, it holds that $e_1 \leq \mu(e_1,e_2)$ and $e_2 \leq \mu(e_1,e_2)$
- Match function is monotonic
  - $\forall e_1, e_2, e_3 \in E$ If $e_1 \leq e_2$ and $M(e_1,e_3)$=true, then $M(e_2,e_3)$=true
- Merge function is monotonic
  - $\forall e_1, e_2, e_3 \in E$ If $e_1 \leq e_2$ and $M(e_1,e_3)$, then $\mu(e_1,e_3) \leq \mu(e_2,e_3)$
- $\forall e_1, e_2, e_3 \in E$ If $e_1 \leq e_3$, $e_2 \leq e_3$ and $M(e_1,e_2)$=true, then $\mu(e_1,e_2) \leq e_3$

What is Best Sequence of Match, Merge Calls that Give us Right Answer?

KB1:SKBRK	
KB1:name	"S. Kubrick"
KB1:birthPlace	KB1:Manhattan
KB1:deathPlace	KB1:UnitedKingdom
KB1:diedIn	1999

e_1

KB1:Stanley_Kubrick	
KB1:birthPlace	KB1:Manhattan
KB1:bornIn	1928-7-26
KB1:parents	KB1:Gertrude Kubrick
KB1:parents	KB1:Jacques Leonard Kubrick
rdf:type	yago:AmericanFilmDirectors

e_2

KB2:SKubrick	
foaf:name	"Stanley Kubrick"
KB2:place_of_birth	KB2:MNHT
rdf:type	foaf:Person
KB2:activeYearsEndYear	7/3/1999
KB2:directed	KB2:A_Clockwork_Orange

e_3

$M(e_2,e_3)$=true

$e_{2,3}=\mu(e_2,e_3)$

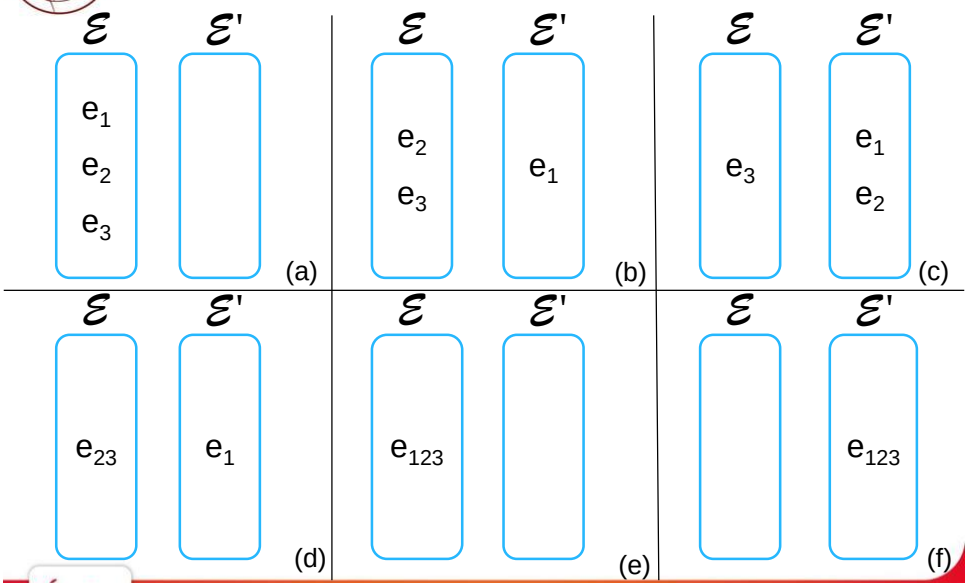
KB12:Stanley_Kubrick	
birthPlace	Manhattan
bornIn	1928-7-26
parents	Gertrude Kubrick
parents	Jacques Leonard Kubrick
rdf:type	AmericanFilmDirectors
activeYearsEndYear	7/3/1999
directed	A_Clockwork_Orange

$M(e_1,e_{2,3})$=true

R-Swoosh [Benjelloun et al., 2009]

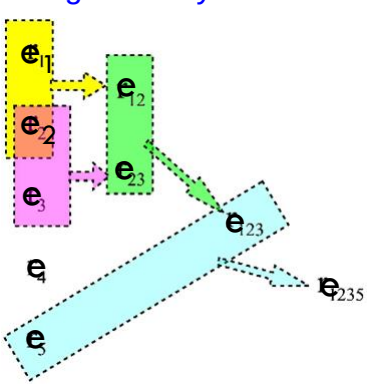
- Assumes ICAR and merge domination
- Idea 1: if $M(e_1,e_2)$ = true we can remove e_1 and e_2
  - Whatever would match e_1 or e_2 now also matches $\mu(e_1,e_2)$
  - Representativity and associativity
- Idea 2: Removal of dominated descriptions is not necessary as a last step in the algorithm
  - Assume e_1 and e_2 appear in final answer where $e_1 \leq e_2$
    - Then $M(e_1,e_2)$ = true and $\mu(e_1,e_2) = e_2$
  - Thus comparison of e_1 and e_2 should have generated merged description e_2, and e_1 should have been eliminated

R-Swoosh Example



ER with ICAR properties

- General ER process is guaranteed to be finite
- Entity descriptions can be matched and merged in any order
- Dominated entity descriptions can be discarded anytime
- Union class of match and merge
  - Union-merge: All values are kept in merged descriptions
  - Union-match: At least one of the values in common



- Commutativity of match and merge functions for highly heterogeneous descriptions does not always hold

Matching and Resolving Entities (II): Matching-based Iterative Resolution

Positive and Negative Match Evidence

Evaluated on **individual entities**

Evaluated on **interrelated entities**

	Hard Constraints	Soft Constraints
Positive Evidence	**FD**: **if** M(e_i, e_j) = true **then** M(e_k, e_l) = true **Example**: if two movies match then their director also match	**FD**: **if** M(e_i, e_j) = true **then most likely** M(e_k, e_l) = true **Example**: if two movies match then their actors are most likely to match
Negative Evidence	**FD**: **if** M(e_i, e_j) = false **then** M(e_k, e_l) = false **Example**: if two directors don't match then movies directed by them don't also match	**FD**: **if** M(e_i, e_j) = false **then most likely** M(e_k, e_l) = false **Example**: if two movies don't match then their actors are less likely to match

Negative constraints are usually stated by domain experts

Constrains can be **recursive**

Additional Constraints [Shen et al, 2005]

Type	Example
Aggregate	No director has produced more than four movies per year
Subsumption	If a movie X from Yago matches a movie Y from IMBD then each studio cited by in Y matches some studio cited by in X
Neighborhood	If actors X and Y share similar names and some co-actors in a movie, they are likely to match
Incompatible	No director has filmed a movie in studios both in Africa and Japan
Layout	If two movies with similar names are mentioned by the same review they are likely to match
Key/Uniqueness	Actors of the same movie must refer to distinct persons
Ordering	If two paper citations match, then their authors will be matched in order
Individual	'Victorina Mérida Rojas' matches with 'Victoria Abril'

Matching-based Iterative ER

- Pair-wise ER: matching decisions are made independently
  - Deduplication, Linkage
- Collective ER: matching decisions depend on other matching decisions according to positive and negative evidence (general constraints)
  - Similarity propagation approaches (more scalable)
    - Dependency graphs of matching decisions
    - Collective relational clustering
  - Probabilistic approaches (scalability is an open issue)
    - Generative Models (for acyclic dependencies between match decisions)
    - Undirected Models based on Markov Networks (sometimes with a first-order logic syntax)
  - Hybrids of constraints and probabilistic models

Similarity Propagation Approaches

- A graph structure for encoding the similarity between entity descriptions and matching decisions, and iteratively assess matching of entities by propagating similarity values
  - Details of how the graph is constructed and traversed and how (content and context) similarity is computed vary
- Similarity-propagation ER: the match function is re-computed at each iteration step by considering previous matching decisions:
  - $M^n(e_i,e_j)$ = true, if $sim^{n-1}(e_i,e_j) \geq \vartheta$
  - $M^n(e_i,e_j)$ = false, if $sim^{n-1}(e_i,e_j) \leq \vartheta'$
  - $M^n(e_i,e_j)$= undecided, otherwise
- Total similarity:

$sim(e_i,e_j)= a*sim_{nbr}(e_i,e_j)+(1-a)*sim_{nbr}(nbr(e_i),nbr(e_j))$

where $nbr(e)$ denotes the neighborhood (in, out) nodes of e

Maintaining the Order of Comparisons

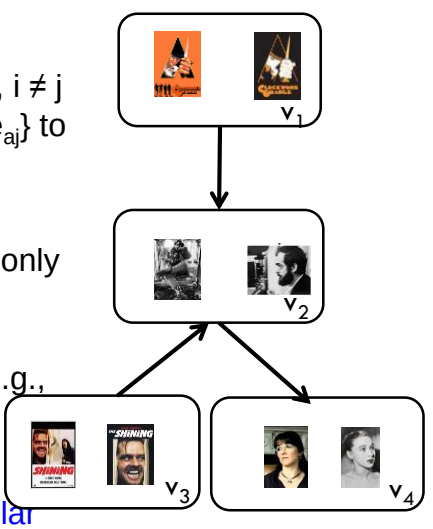
- In similarity-propagation approaches the order of comparisons is dynamic
- Graph traversal usually supported by a priority queue (PQ) on the similarity score of nodes
  - As entities are resolved, the PQ is updated for maximizing effectiveness & reducing re-comparisons
- Different strategies of order maintenance:
  - Based on heuristics
    - type of nodes and edge direction [Dong et al. 2005]
    - degree of nodes [Weis & Naumann 2006]
    - edge weights [Kalashnikov & Mehrotra 2006]
  - Triggered by recent matches [Böhm et al. 2012, Lacoste-Julien et al. 2013]
  - Lazy maintenance [Herschel et al. 2012, Altowim et al. 2014]

Dependency Graph [Dong et al., 2005]

- Works on an entity graph constructed from the relational records
  - nodes represent similarity comparisons between pairs of records and/or their attribute values (real-valued)
  - edges represent match decisions based on the matching of associated nodes (boolean-valued)
- A matching decision is taken when the real-valued similarity score (between 0 and 1) of a node is above a threshold θ
  - If it exceeds the threshold, it is marked as match, otherwise as undecided,
  - if no more neighbors are undecided, it is marked as non-match
- Idea 1: consider richer matching evidence
- Idea 2: propagate similarity between matching decisions
- Idea 3: Gradually enrich references by merging attribute values (Swoosh-style)

Dependency Graph: Example

- Let E be a set of entity descriptions
  - A node $v = \{e_i, e_j\}$, where $e_i, e_j \in E$, $i \neq j$
  - An edge $e = (v_a, v_b)$ from $v_a = \{e_{ai}, e_{aj}\}$ to $v_b = \{e_{bi}, e_{bj}\}$ implies $e_{bi}, e_{bj} \in values(e_{ai}) \cup values(e_{aj})$
- Directed edge when dependency is only in one direction
  - can be stated by domain experts
  - inferred by the data semantics (e.g., keys/forein keys, rdf properties)
- Include only nodes whose two entities have the potential to be similar



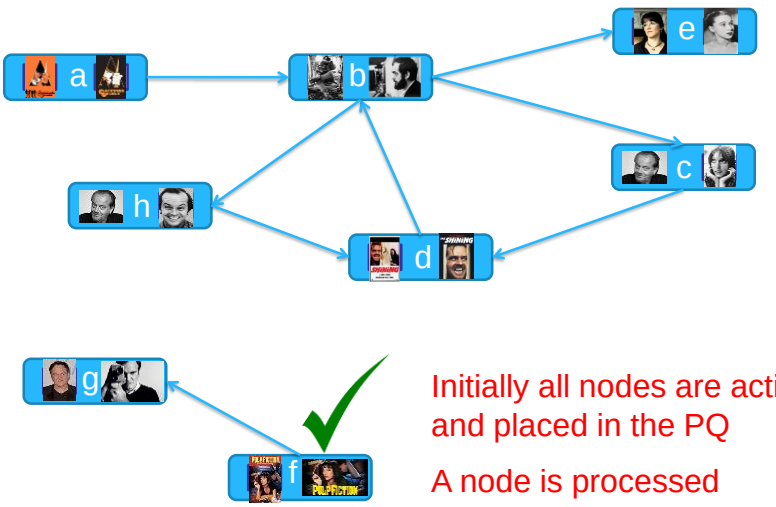
Consider Richer Matching Evidence [Dong et al., 2005]

- Positive evidence (i.e., constraints for match nodes) is captured by the Boolean similarity of neighborhood nodes
  - Strong-boolean: Resolution implies resolution of neighbour
    - E.g., if two movies are matched then director must also be matched
  - Weak-boolean: No direct implication
    - E.g., similarity of two movies increases as their `rdf:labels` are highly similar
- Negative evidence (i.e., constraints for non-match nodes) is verified after similarity propagation is performed, and inconsistencies are fixed

Similarity Propagation [Dong et al., 2005]

- Similarity function for node u: $sim(u) = S_{rv} + S_{sb} + S_{wb}$
  - S_{rv}: from real-valued neighbors (decision-tree shape)
  - S_{sb}: from strong-boolean-valued neighbors
  - S_{wb}: from weak-boolean-valued neighbors
- When a node is matched, the similarity score of its neighbors is re-computed
- Process converges if
  - Similarity score is monotone in the similarity values of neighbors
  - Compute neighbor similarities only if similarity increase is not too small

Traversing the ER Graph

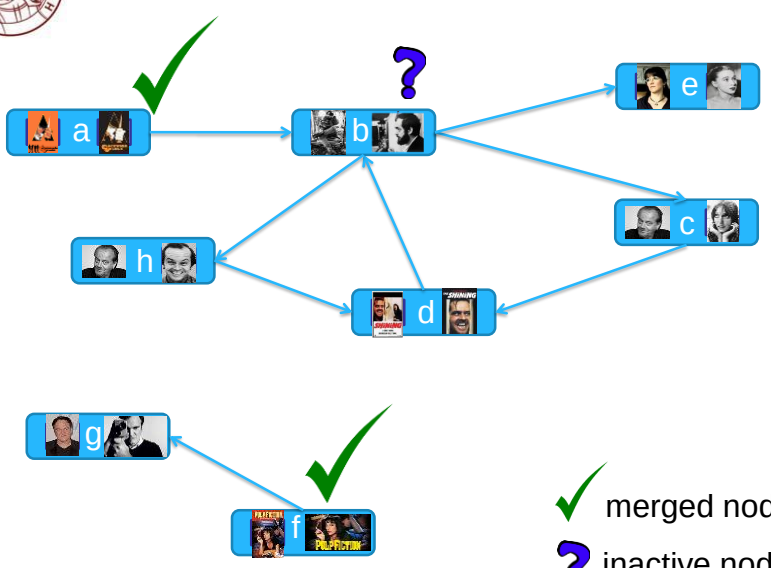


PQ
f
a
b
c
h
d
e
g

Initially all nodes are active and placed in the PQ

A node is processed before its out-neigbors

Traversing the ER Graph

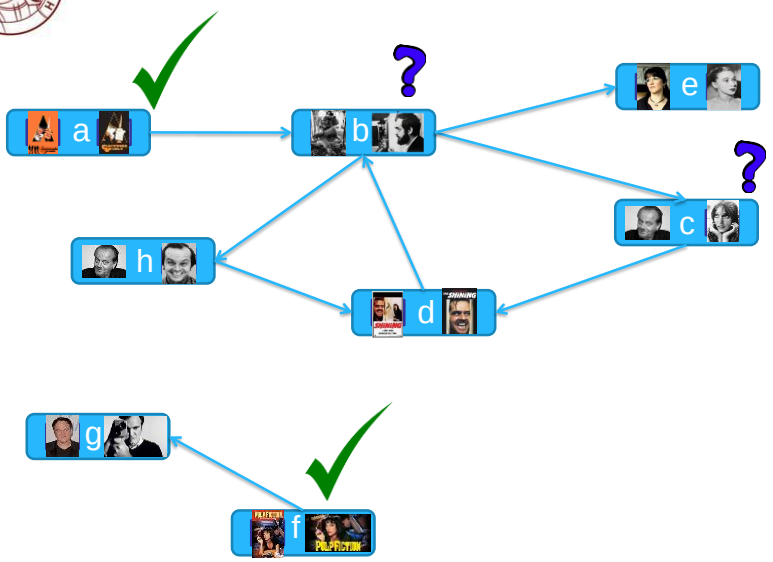


PQ
b
c
h
d
e
g

✓ merged node

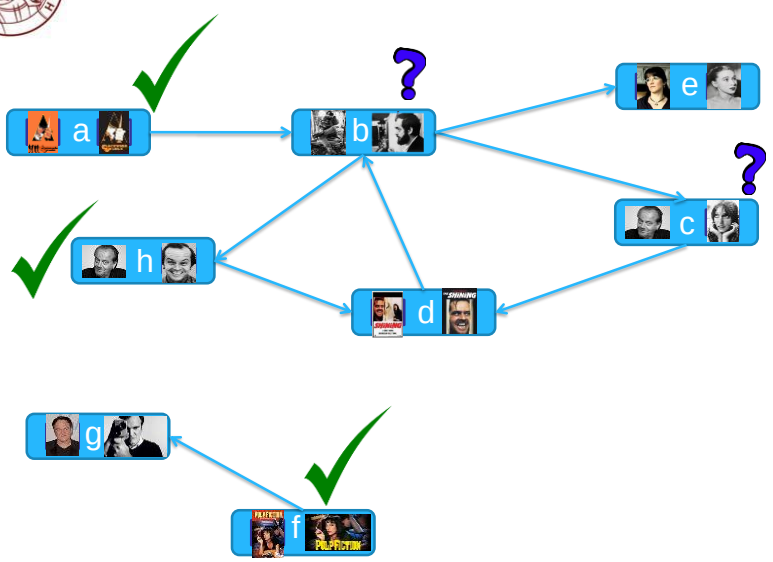
? inactive node

Traversing the ER Graph



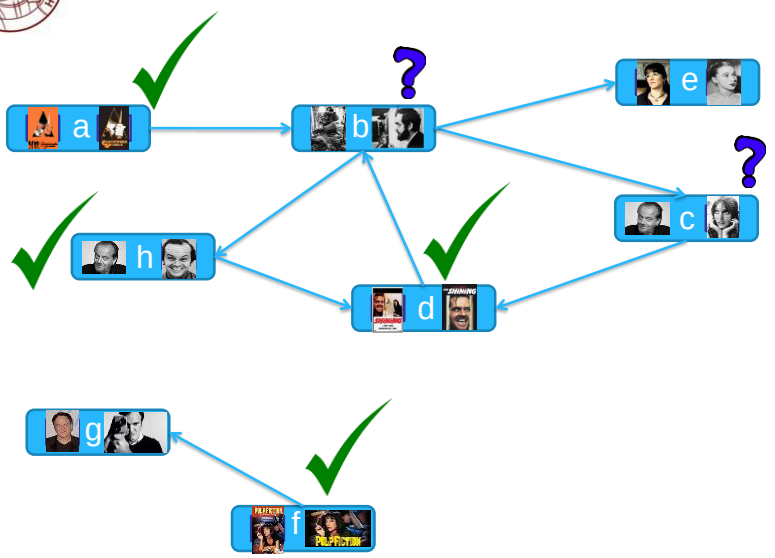
PQ
c
h
d
e
g

Traversing the ER Graph



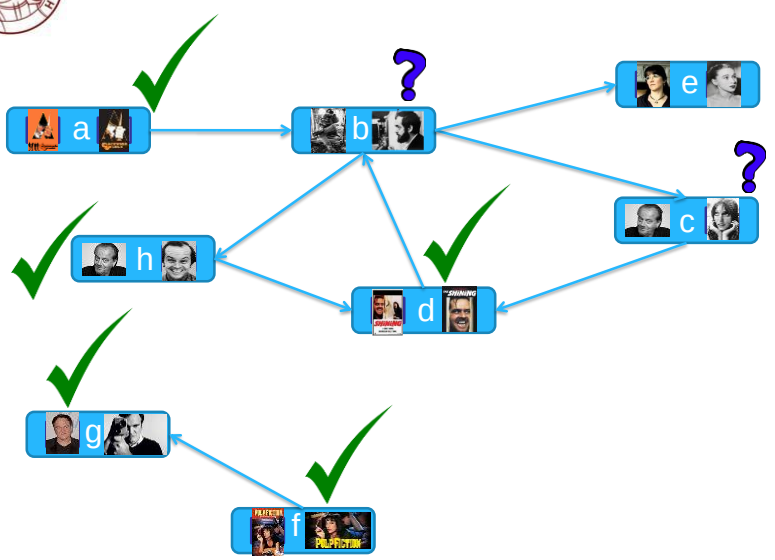
PQ
h
d
e
g

Traversing the ER Graph



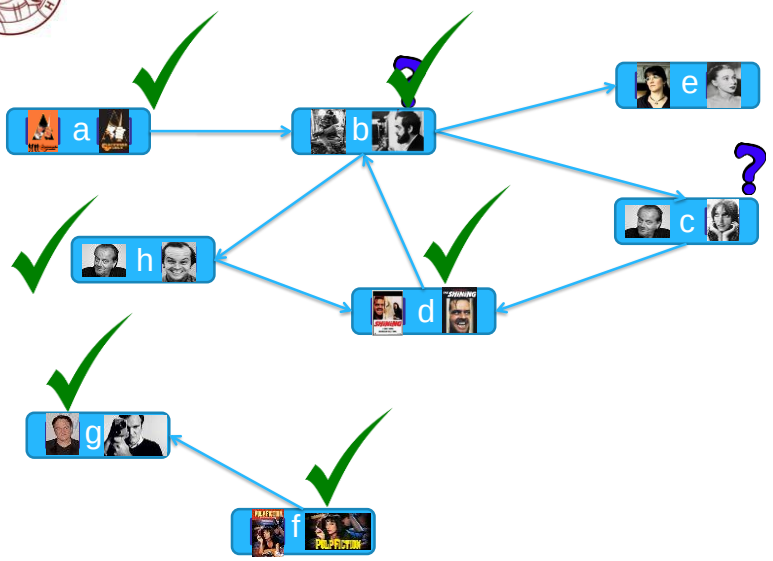
PQ
d
e
g

Traversing the ER Graph



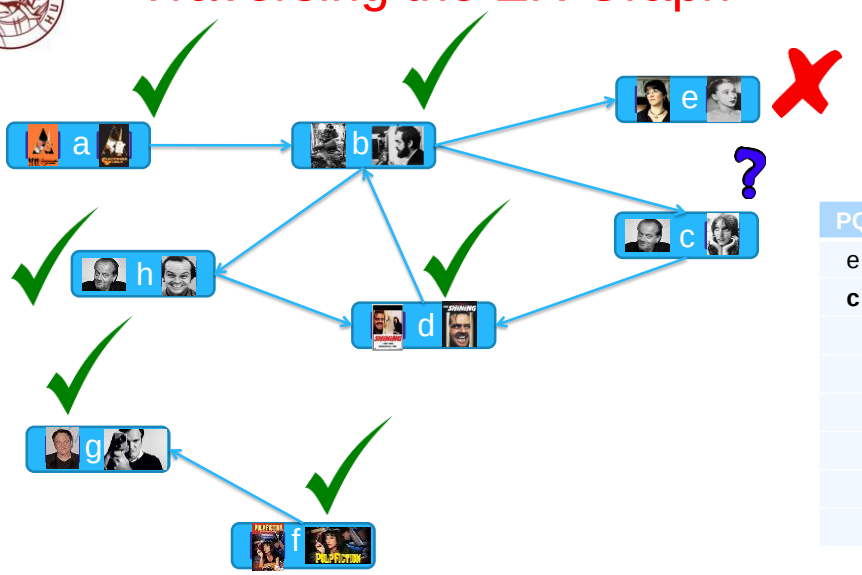
PQ
g
b
e

Traversing the ER Graph



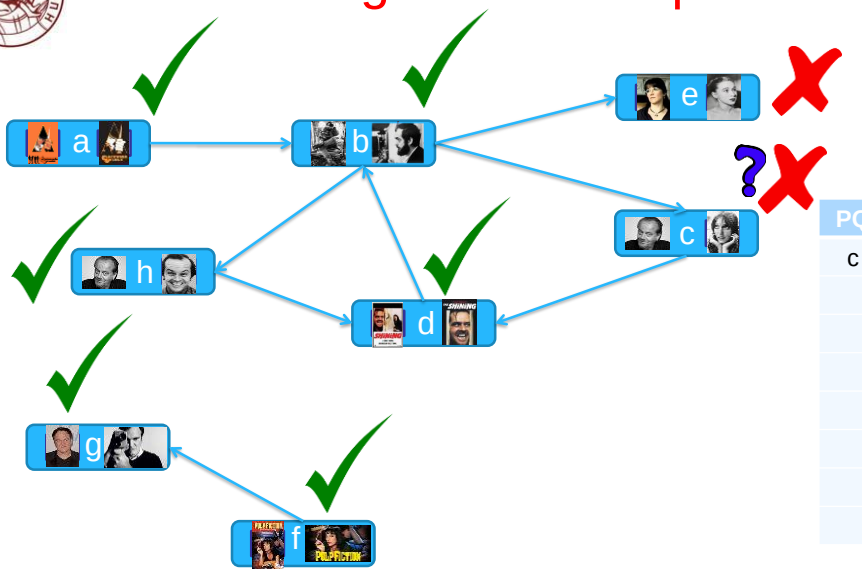
PQ
b
e

Traversing the ER Graph



PQ
e
c

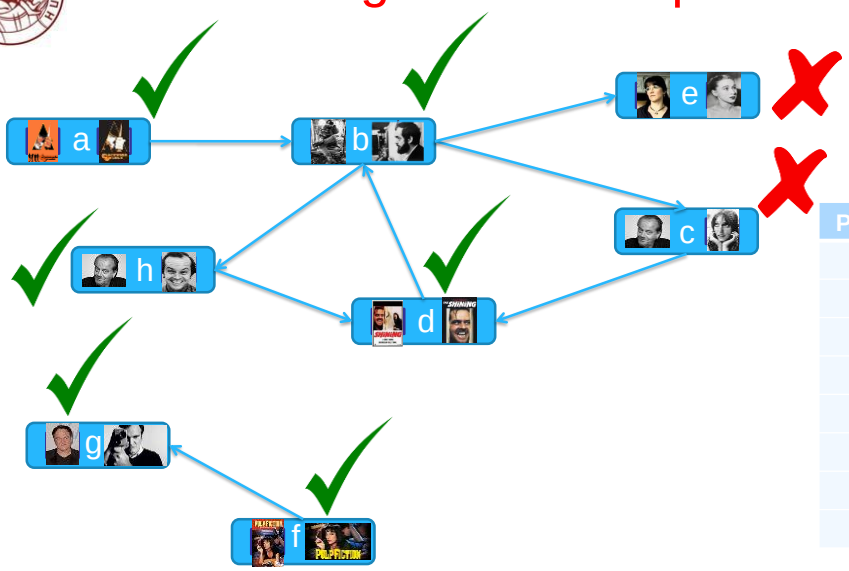
Traversing the ER Graph



PQ

C

Traversing the ER Graph



PQ

Linda [Böhm et al. 2012]

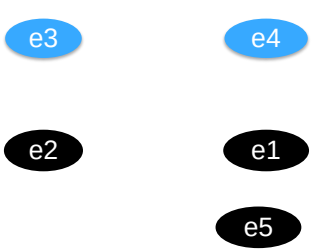
- Works on an entity graph constructed from the RDF descriptions
  - exploits the unique mapping constraint between two KBs
- Key Idea: the more matching neighbors via similar relationships two descriptions have, the more likely it is that they match
  - String similarity of the literal values of entities: checked once
  - Contextual similarity of the graph neighbors:checked iteratively
- Two square matrices (|E| × |E|) are used:
  - X captures the identified matches (binary values)
  - Y captures the pair-wise similarities (real values)
    - Initialization: common neighbors & string similarity of literals
    - Updates: Use the new identified matches of X
- Until the priority queue (extracted from Y) becomes empty:
  - Get the pair (e_i, e_j) with the highest similarity: match by default!
    - Update X: matches of e_i are also matches of e_j
  - Update the similarity of nodes influenced by the new matches

Linda Algorithm Example

Matches	e1	e2	e3	e4	e5
e1	1	0	0	0	0
e2		1	0	0	0
e3			1	0	0
e4				1	0
e5					1

PQ
e1 – e4
e2 – e4
e1 – e3
e5 – e3
e2 – e3
…

A priority queue, derived by an initial similarity computation between **all pairs**, based on their attribute values

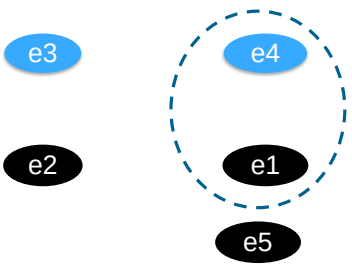


Linda Algorithm Example

Matches	e1	e2	e3	e4	e5
e1	1	0	0	1	0
e2		1	0	0	0
e3			1	0	0
e4				1	0
e5					1

PQ
e1 – e4
e2 – e4
e1 – e3
e5 – e3
e2 – e3
…

the head of PQ is a match by default



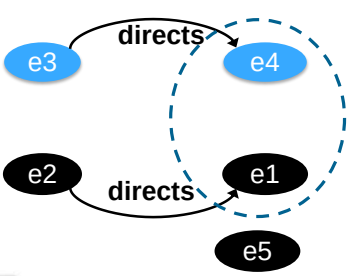
Linda Algorithm Example

Matches	e1	e2	e3	e4	e5
e1	1	0	0	1	0
e2		1	0	0	0
e3			1	0	0
e4				1	0
e5					1

PQ
~~e2 – e4~~
~~e1 – e3~~
e2 – e3 ↑
e5 – e3 ↓
…

unique mapping constraint (1-1 Assumption)

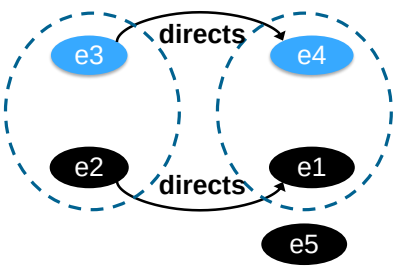
similarity re-computation, based on the matching neighbors and the names of the links to them



Linda Algorithm Example

Matches	e1	e2	e3	e4	e5
e1	1	0	0	1	0
e2		1	1	0	0
e3			1	0	0
e4				1	0
e5					1

PQ
e2 – e3
e5 – e3
…

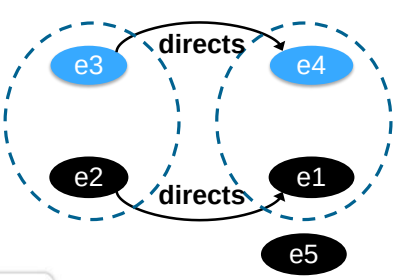


Linda Algorithm Example

Matches	e1	e2	e3	e4	e5
e1	1	0	0	1	0
e2		1	1	0	0
e3			1	0	0
e4				1	0
e5					1

PQ
~~e5 – e3~~
…

unique mapping constraint (1-1 Assumption)



stops when PQ is empty

Linda Distributed Version

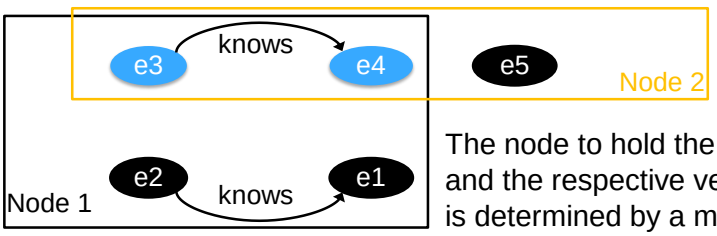
Matches	e1	e2	e3	e4	e5
e1	1	0	0	0	0
e2		1	0	0	0
e3			1	0	0
e4				1	0
e5					1

PQ
e1 – e4
e2 – e4
e1 – e3
e2 – e3
…

Node 1

PQ
e5 – e3
e5 – e4
…

Node 2

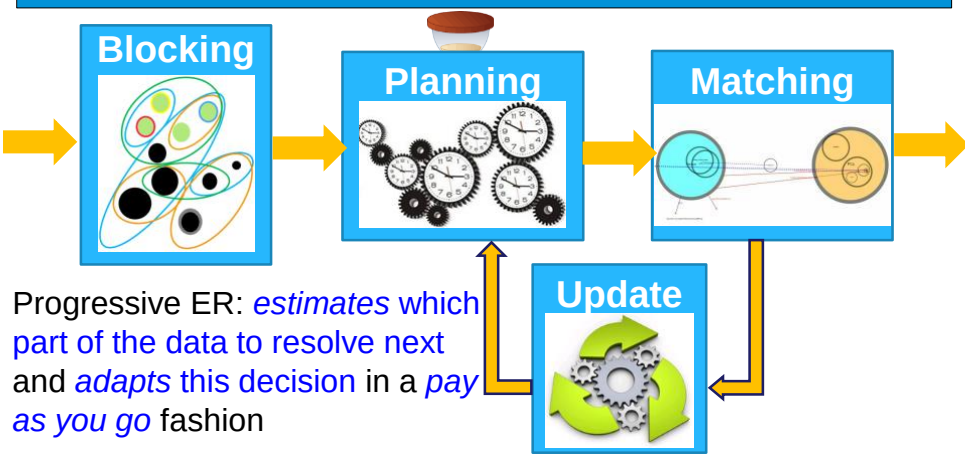


The node to hold the PQ entry (a,b) and the respective vertex neighbors is determined by a modulo operation of the first component (a)

Matching and Resolving Entities (II): Progressive Resolution Techniques

Progressive ER

Optimization: maximize **benefit** (number or type of matches) for a given **cost** (number of comparisons, disk/cloud access)

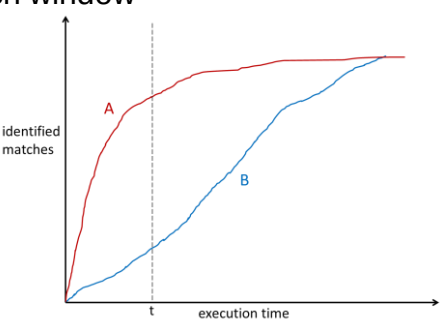


Progressive ER: *estimates* which part of the data to resolve next and *adapts* this decision in a *pay as you go* fashion

- Good for high Velocity

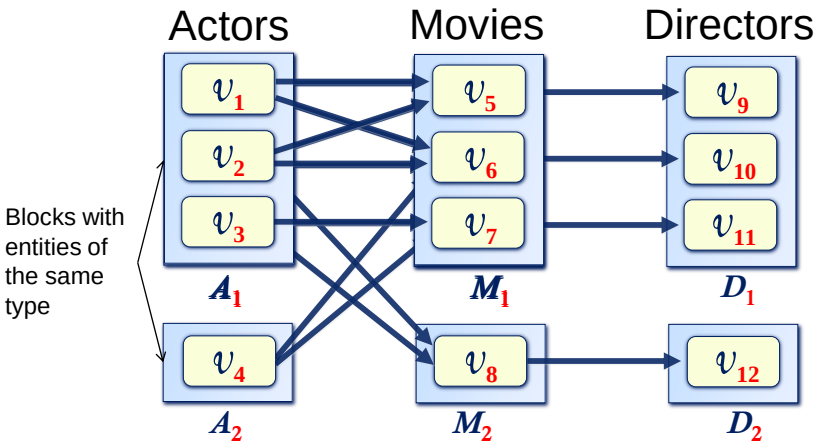
Progressive Approach to Relational Entity Resolution [Altowim et al. 2014]

- Key Idea: divides the ER process into several windows and generates a resolution plan for each window
  - specifies which blocks and entity pairs within these blocks will be resolved during the plan execution phase of a window
  - associates with each identified pair the order in which to apply the similarity functions on the attributes of the two entities



- Lazy resolution strategy to resolve pairs with the smallest cost
  - Unlike single entity type resolution a block based prioritization is significantly more important when resolving multiple types

Relational Progressive ER: Example

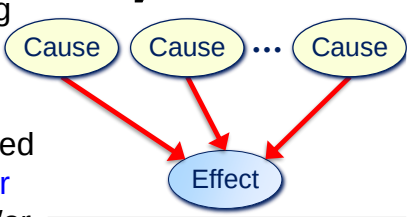


Matching Probability Estimation

- The nodes in the ER graph influencing a pair of entities are interpreted as common causes to the same effect
- The probability that a node is processed at each round depends on the number of cause nodes that are matches and/or on the percentage of matching nodes in the block of the effect node

$$\mathcal{P}(v_i) = 1 - \prod_{x_j \in \mathbf{X}} 1 - \mathcal{P}(v_i|x_j)$$

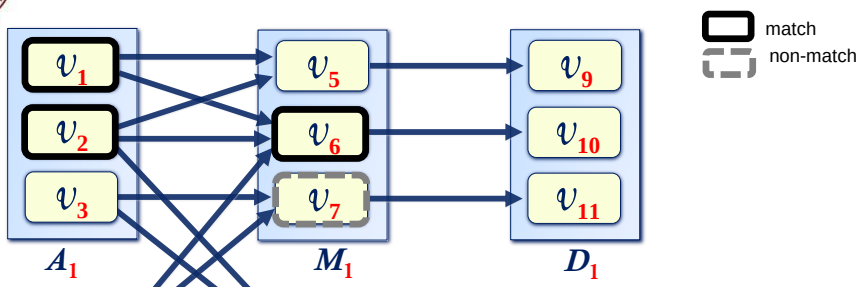
Noisy-OR Model



not required to know all the causes

Effect: Node considered for matching v_i
Causes: (a) Influencing matching nodes x_j
(b) Fraction of matching pairs in the block of v_i

Heuristics for Estimating Node Benefit

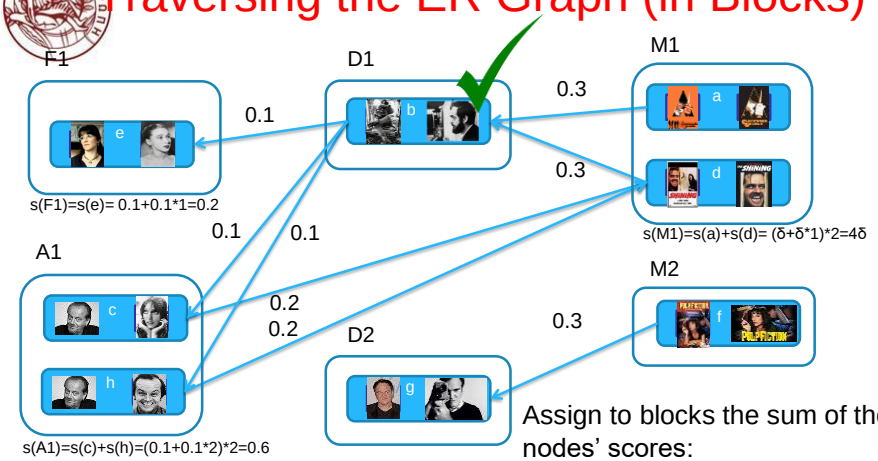


$$Benefit(v_i) = \mathcal{P}(v_i|\mathcal{S}) + \mathcal{P}(v_i|\mathcal{S}) * Imp(v_i)$$

Direct Indirect

The Impact is estimated for k nodes of the same entity type, as the average number of their direct dependent unresolved nodes

Traversing the ER Graph (in Blocks)

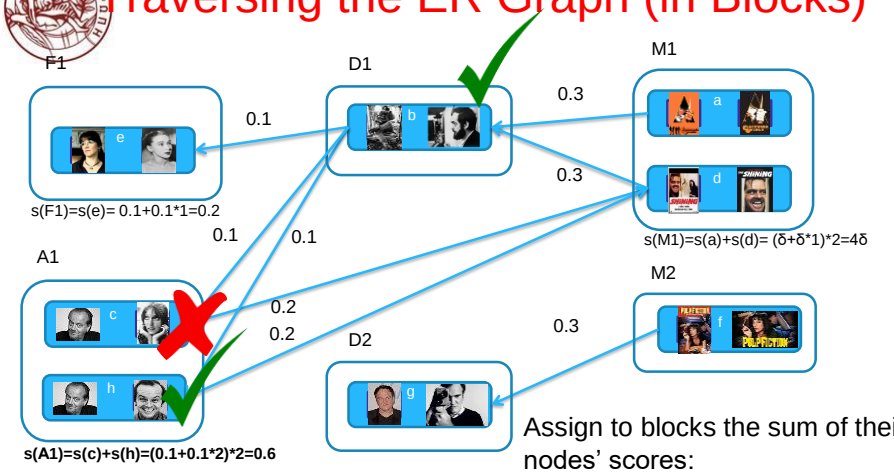


Assign to blocks the sum of their nodes' scores:
$s(B) = \sum_{v_i \in B} P(vi) + P(vi) * Impact(vi)$

Impact(directors) = 3,
Impact(actors) = 2,
Impact(movies) = 1
Impact(family) = 1
Initial $P(v_i)$ = δ

Initially, a block of the entity type with the highest out-degree (impact) is selected (in random), since $P(v_i)$ is the same for all nodes (of the same type)

Traversing the ER Graph (in Blocks)



Assign to blocks the sum of their nodes' scores:

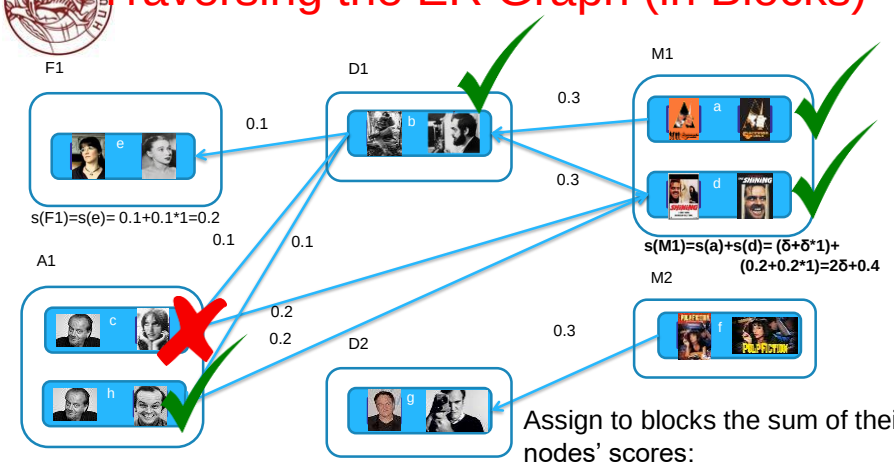
$s(B) = \sum_{v_i \in B} P(vi) + P(vi) * Impact(vi)$

Impact(directors) = 3,
Impact(actors) = 2,
Impact(movies) = 1
Impact(family) = 1
Initial $P(v_i) = \delta$

Then, update the scores, load the block with the highest score and resolve its nodes

Fall 2019

Traversing the ER Graph (in Blocks)



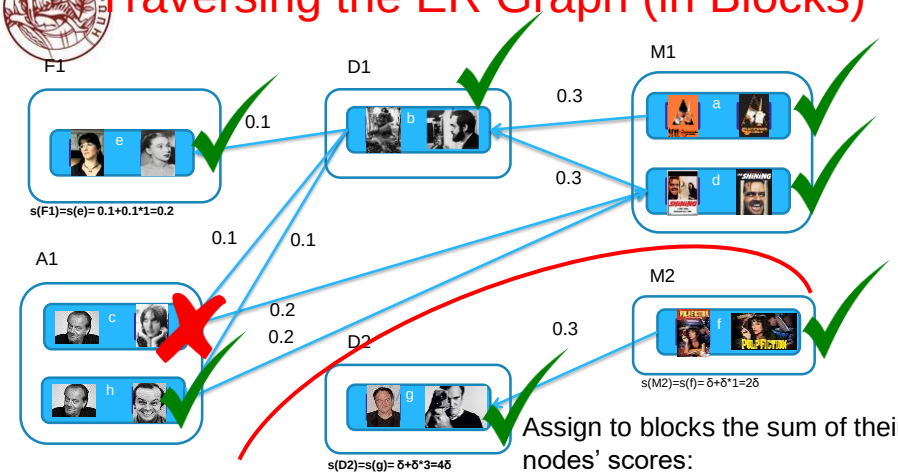
Assign to blocks the sum of their nodes' scores:

$s(B) = \sum_{v_i \in B} P(vi) + P(vi) * Impact(vi)$

Impact(directors) = 3,
Impact(actors) = 2,
Impact(movies) = 1
Impact(family) = 1
Initial $P(v_i) = \delta$

Then, update the scores, load the block with the highest score and resolve its nodes

Traversing the ER Graph (in Blocks)



Assign to blocks the sum of their nodes' scores:
$s(B) = \sum_{v_i \in B} P(vi) + P(vi) * Impact(vi)$

Impact(directors) = 3,
Impact(actors) = 2,
Impact(movies) = 1
Impact(family) = 1
Initial $P(v_i) = \delta$

Then, update the scores, load the block with the highest score and resolve its nodes

Comparison of ER Graph Traversals

Property	[Dong et al. 2005]	[Böhm et al. 2012]	[Altowim et al. 2014]
Seed node selection	any node with no in-edges	the node with the highest content_sim	any node of the entity type with the highest out-degree
Matching condition	similarity is above a threshold value	the highest score (head of PQ)	a (black-box) resolve function returns true
Similarity/Evidence Update (PQ)	weighted sum of in-neighbors similarities	weighted sum of matching in-/out-neighbors similarities	leaky noisy-or of matching in-neighbors
Backtracking	✓	✗	✗
Traversal policy	topological sort	advance in the PQ the neighbors (with most similar relationships) of the recent match	sort PQ based on #in-matches and #out-neighbors

Web-scale Progressive ER: Ongoing Work

- Relational progressive ER algorithms assume that the more entity pairs are correctly identified, the higher the quality of the result is expected to be
- We are interested in characterizing the quality of resolved pairs
  - # of real-world entities resolved ***(shallow strategy)***
    - entity-centric search (*entity coverage*)
  - # of real-world entity graphs resolved ***(deep strategy)***
    - entity-centric recommendation (*relationship completeness*)
  - # descriptions resolved for the same real-world entity ***(tail strategy)***
    - web-scale knowledge curation (*attribute completeness*)

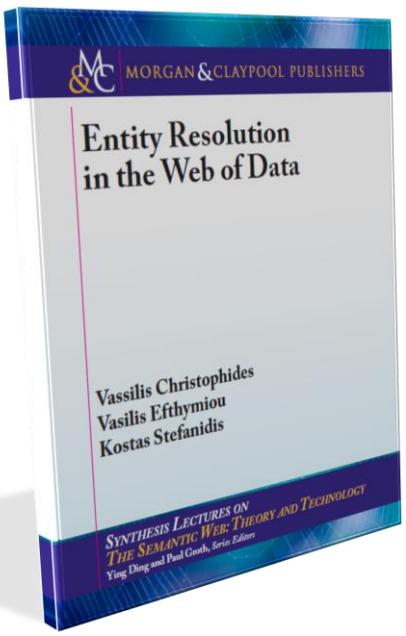
Matching and Resolving Entities (II): Challenges and Open Issues

Challenges and Open Issues

- Tight coupling of Blocking with Iterative Matching/Merging
  - Better control of block characteristics w.r.t. the entity similarity subsequently used (see [J. Fisher et al. 2015]
- Progressive ER with Quality Guarantees
  - guarantees (e.g., coverage) regarding the quality of matches/ merges w.r.t. subsequent entity-centric services and data analysis tasks
- ER for Big Data
  - Algorithms for high Velocity (see [D. Firmani et al. 2016]), Variety, and Volume entity descriptions (see [Q. Wang et al. 2015] [L. Kolb et al. 2012])
- Large-Scale ER Testbeds
  - Real-world ground truth datasets for different match types and open source ER platforms (see [Efthymiou et al. 2015, 2016])

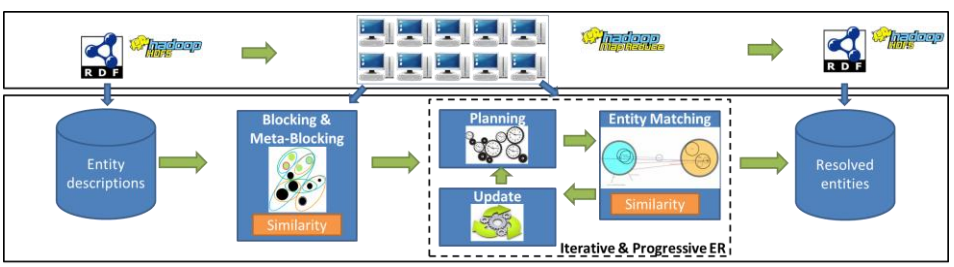
Challenges and Open Issues

- Crowdsourced ER:
  - reduce the crowdsourcing cost for obtaining ground truth (see [Chai et al. 2016] [Gokhale et al. 2014] [Wang et al. 2012])
- Temporal ER:
  - resolve evolving entity descriptions and analyse the history of descriptions (see [Dong & Tan 2015])
- Uncertain ER:
  - consider confidence scores when resolving certain & uncertain entity descriptions (see [Gal 2014] [Demartini et al. 2013])
- Privacy-aware ER:
  - Trade-off between entity obfuscation techniques and ER results quality (see [Whang & Garcia-Molina 2013])



http://www.morganclaypoolpublishers.com/catalog_Orig/product_info.php?products_id=823

The Minoan ER Framework



http://csd.uoc.gr/~vefthym/minoanER

Acknowledgements

- EU FP7-PEOPLE-2013-IRSES SemData
- Big Geospatial Data Quality and User Privacy Défi Mastodons CNRS 2016

License

Questions?