WEB-SCALE BLOCKING, ITERATIVE AND PROGRESSIVE ENTITY RESOLUTION

Kostas Stefanidis
Vassilis Christophides
Vasilis Efthymiou

Kostas Stefanidis
- University of Tampere, Finland
- Kostas.Stefanidis@uta.fi

Vassilis Christophides
- INRIA-PARIS, France & University of Crete, Greece
- Vassilis.Christophides@inria.fr

Vasilis Efthymiou
- ICS-FORTH & University of Crete, Greece
- vefthym@ics.forth.gr

DESCRIBING AND LINKING ENTITIES: ENTITY-CENTRIC APPLICATIONS & KNOWLEDGE BASES

WHY ENTITIES

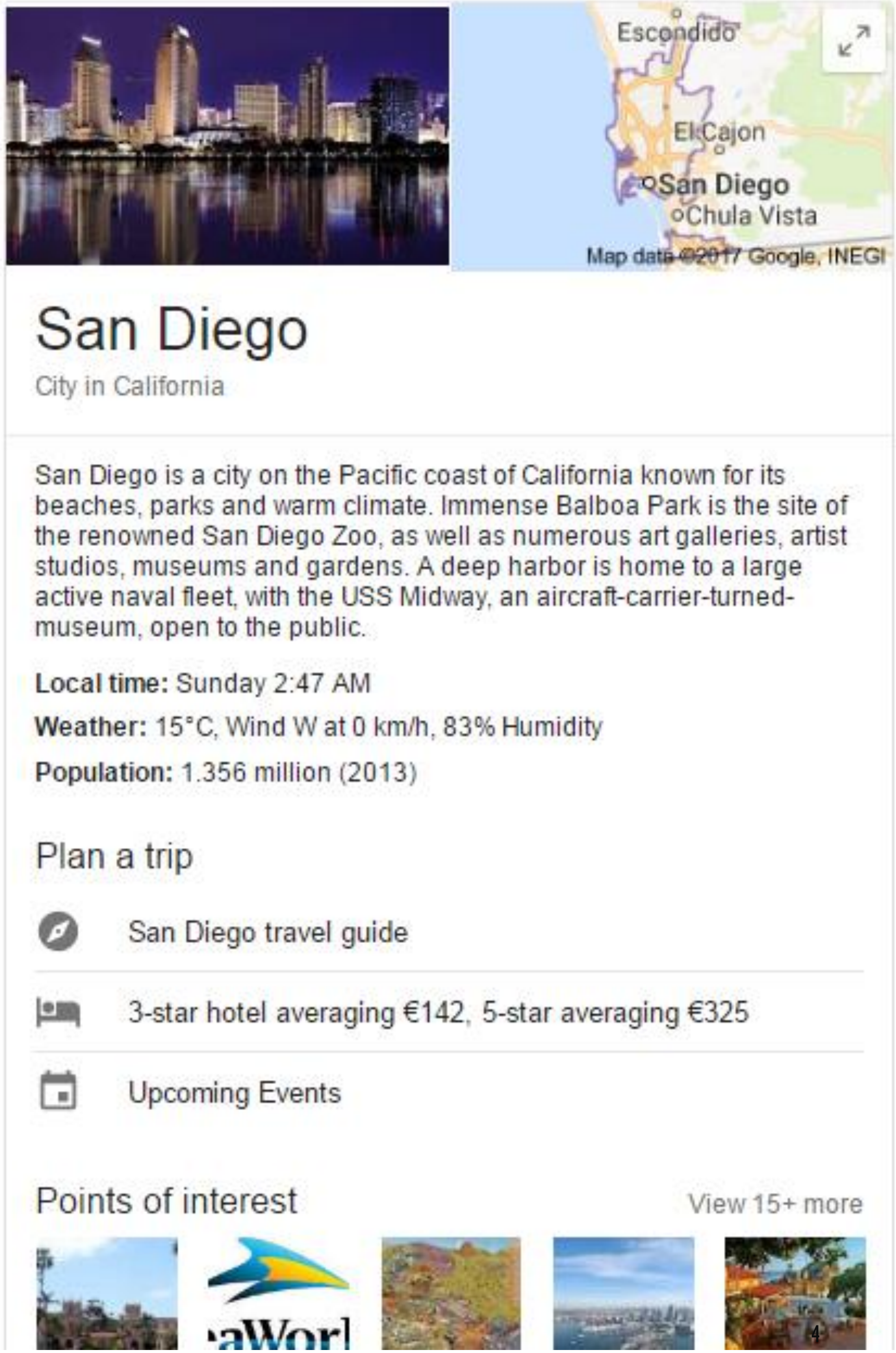
Entities is what a large part of our knowledge is about

Bing reported that people searching for entities alone account for the 10% of all their search volume

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

Photos

Map



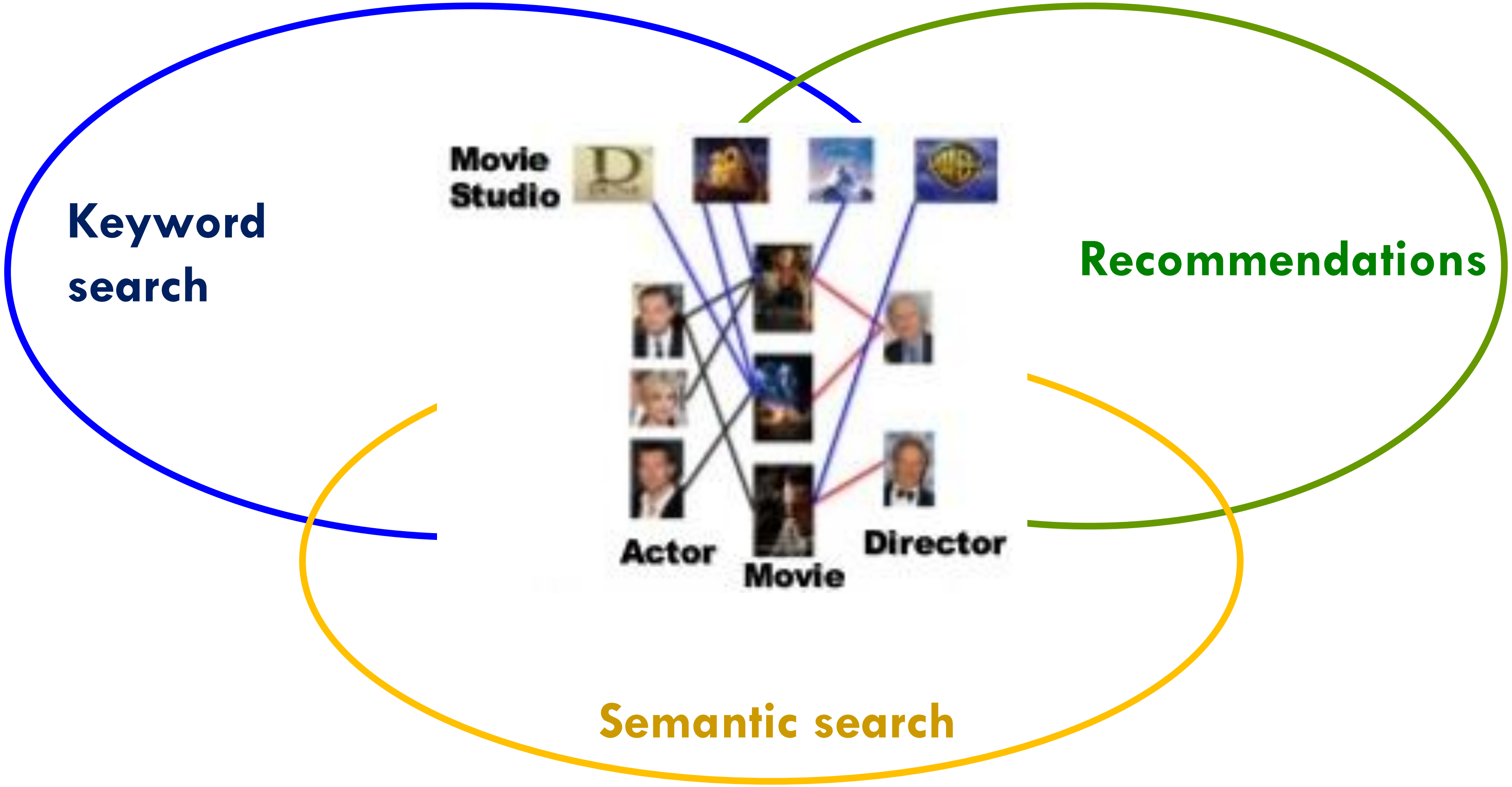
Time

Weather

Population

Places to go

PUSH/PULL TECHNIQUES FOR RETRIEVING WEB CONTENT



CORE ENTITIES

Locations

Organizations

Persons

Movies

WHAT IS A KNOWLEDGE BASE (KB)?

DBpedia

dbpedia:Stanley_Kubrick	
dbo:birth Place	**dbpedia:Manhattan**
rdf:type	foaf:Person
rdf:type	yago:AmericanFilmDirectors
rdf:type	yago:Amateur ChessPlayers

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

- Semantic Web data model

KNOWLEDGE BASES

Domain-specific Knowledge Bases

- 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 Bases

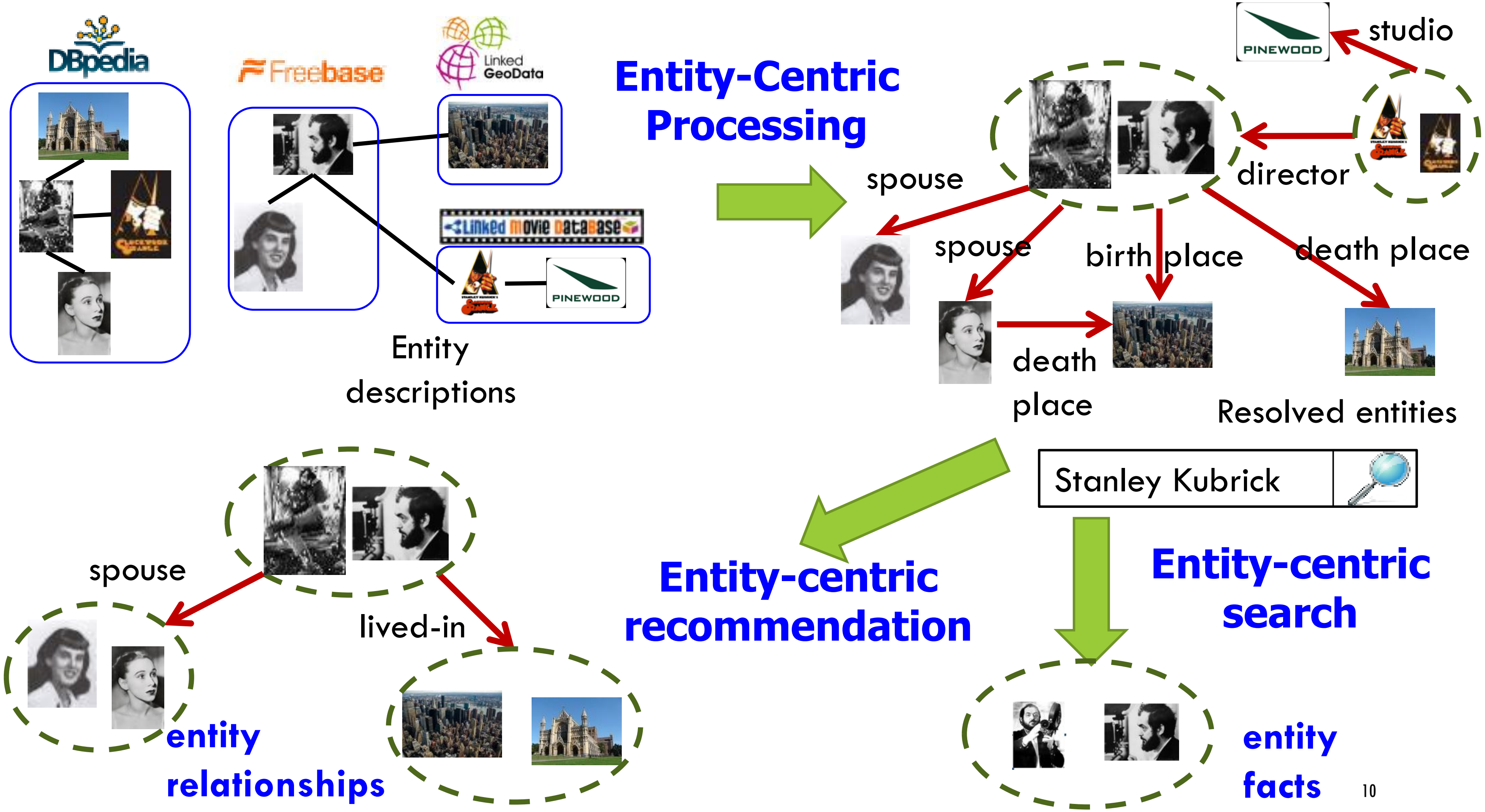
- Cover a variety of knowledge across domains
  - DBPedia, Yago, Freebase, Knowledge Graph, Satori Bing, Knowledge Vault

KNOWLEDGE BASES IN NUMBERS

KB	# Entities	# Classes	# RDF triples	# Properties
YAGO2	10M	350K	120M	100
DBpedia (en)	4.58M	685	583M	2.795K
Freebase	46.3M	1.5K	2.67B	4.5K
Knowledge Graph	600M	1.5K	20B	35K
Knowledge Vault	45M	1.1K	1.6B	4.6K

Numbers from 2014

ENTITY-CENTRIC APPLICATIONS



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

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

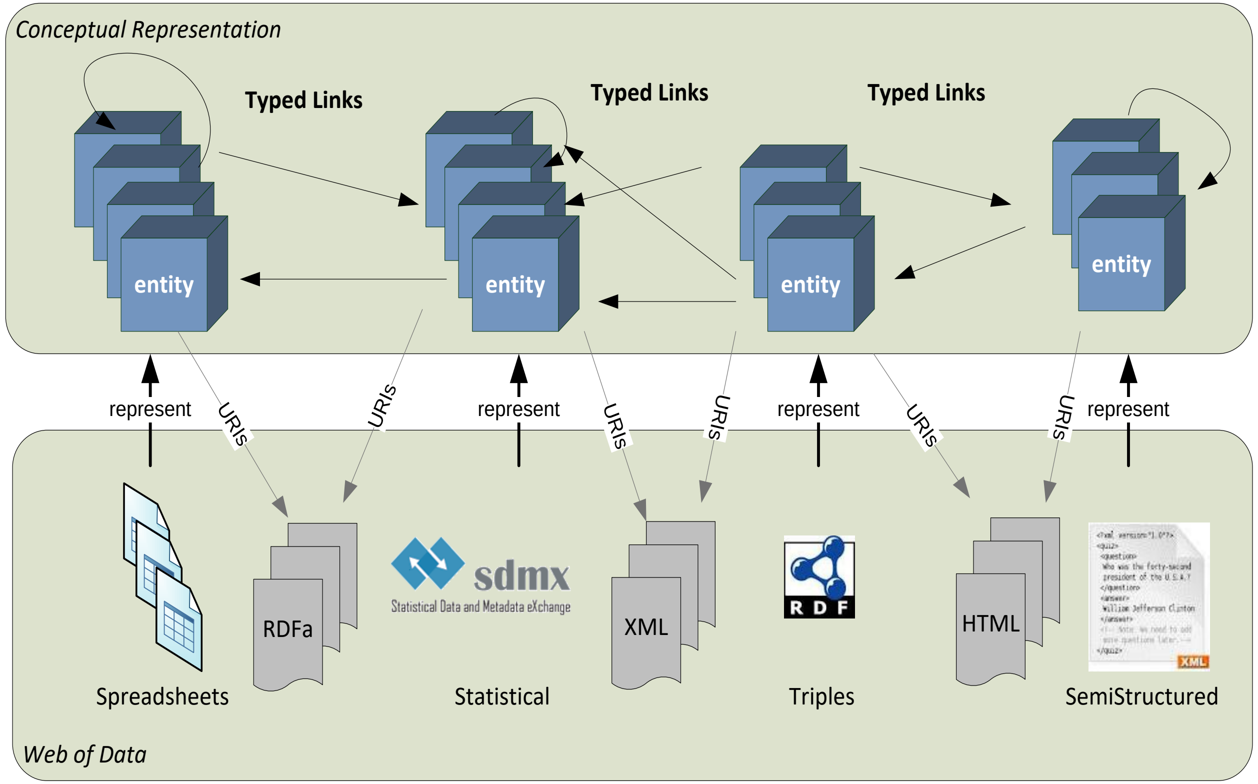
LINKED KNOWLEDGE BASES AND THE WEB OF DATA

THE WEB OF DATA

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

Global data space connecting data from diverse domains & sources

- <u>Primary objects</u>: “things” (or descriptions of “things”)
- <u>Links</u> between “things” and not “strings”



http://linkeddata.org/guides-and-tutorials

THE LINKED DATA PRINCIPLES

Linked Data is about using the Web to connect related data that wasn't previously linked, or to link data currently linked using other methods

Anyone can publish data on the Web for 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
- 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 definitions

LINKING ENTITIES IN KNOWLEDGE BASES

KB publishers are encouraged to describe and interlink real world entities using the RDF data model

dbpedia:Stanley_Kubrick	
dbo:birth Place	**dbpedia:Manhattan**
rdf:type	foaf:Person
rdf:type	yago:AmericanFilmDirectors
rdf:type	yago:Amateur ChessPlayers

dbpedia:Manhattan	
rdfs:label	"Manhattan"
rdf:type	schema.org:Place
dbo:populationTotal	1,626,159

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

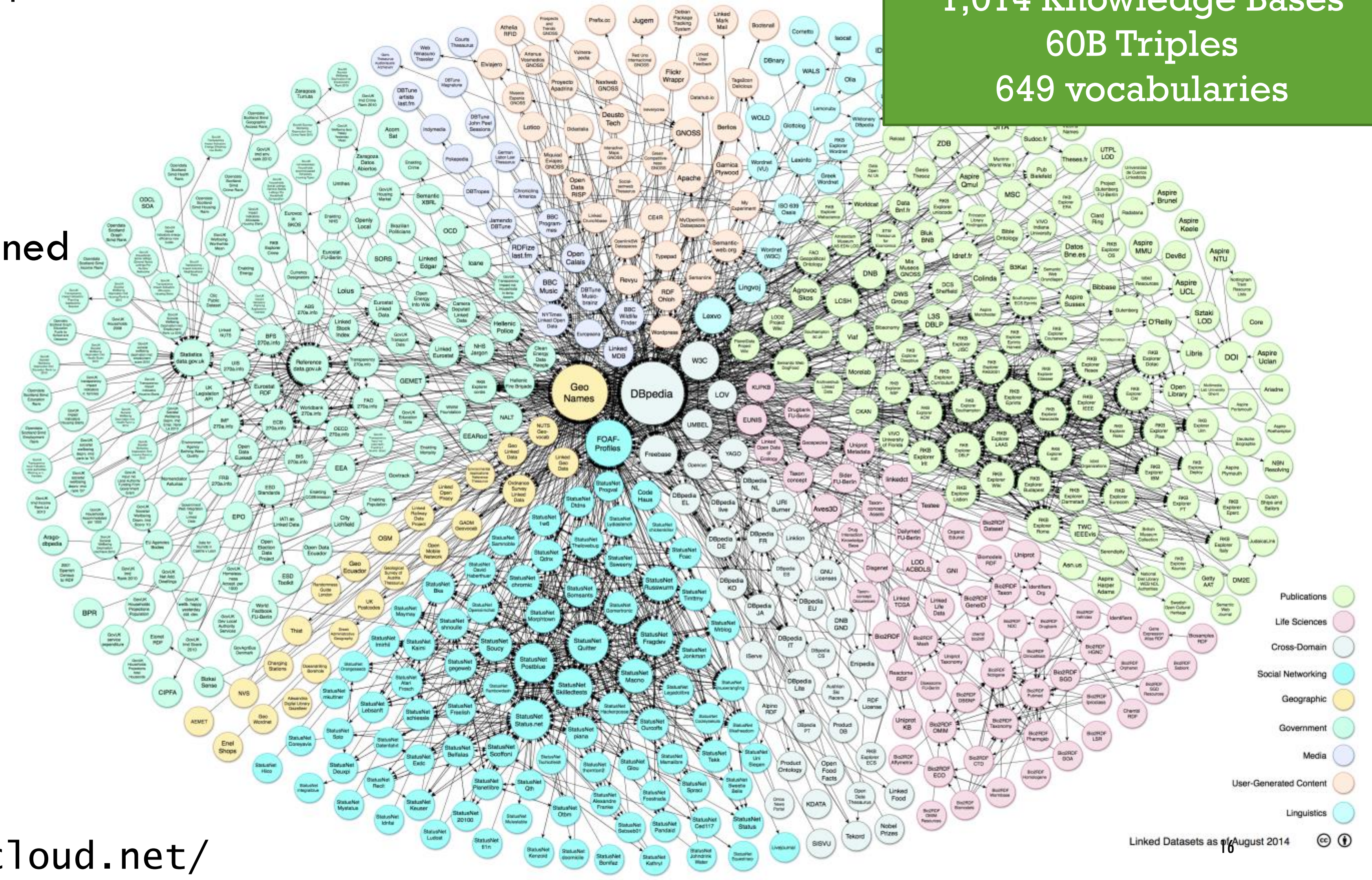
dbpedia: A_Clockwork_Orange_(film)	
dbo:director	**dbpedia:Stanley_Kubrick**
dbo:Work/runtime	'136'
foaf:name	"A Clockwork Orange"

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

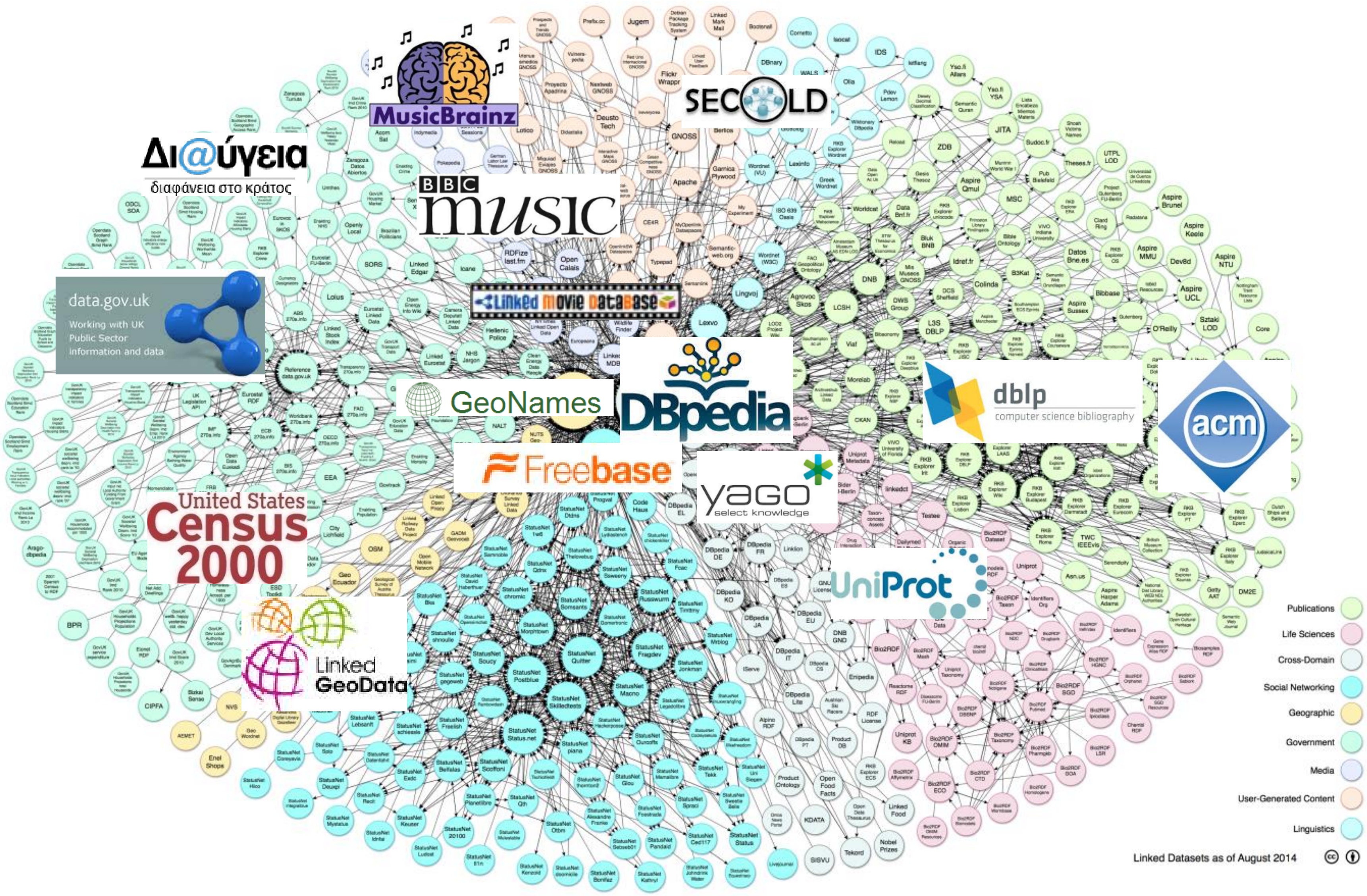
THE LINK OPEN DATA (LOD) CLOUD

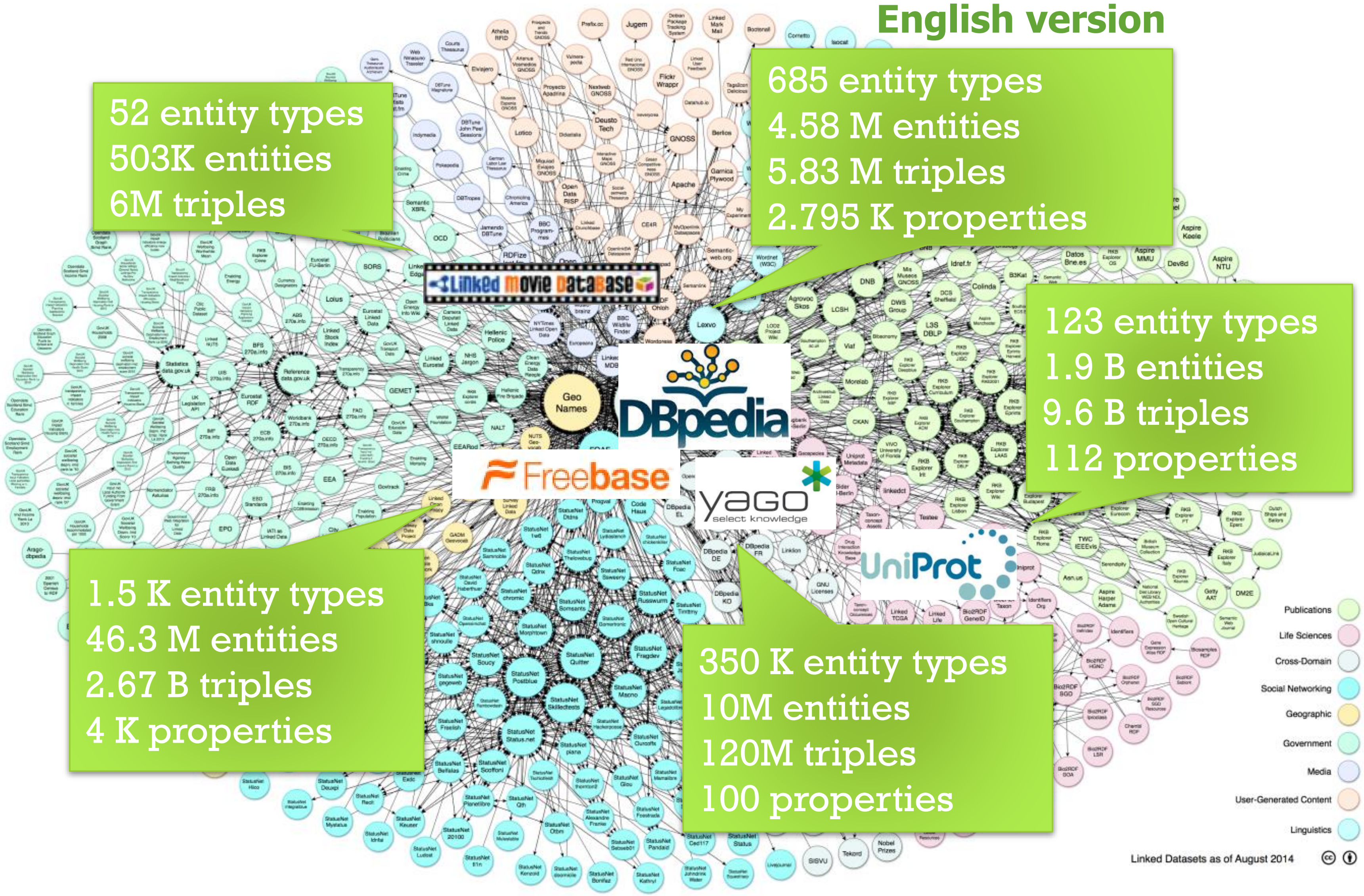
1,014 Knowledge Bases
60B Triples
649 vocabularies

- Nodes are KBs (aka RDF datasets) published, maintained or aggregated by a single provider
- Edges are links crossing KBs

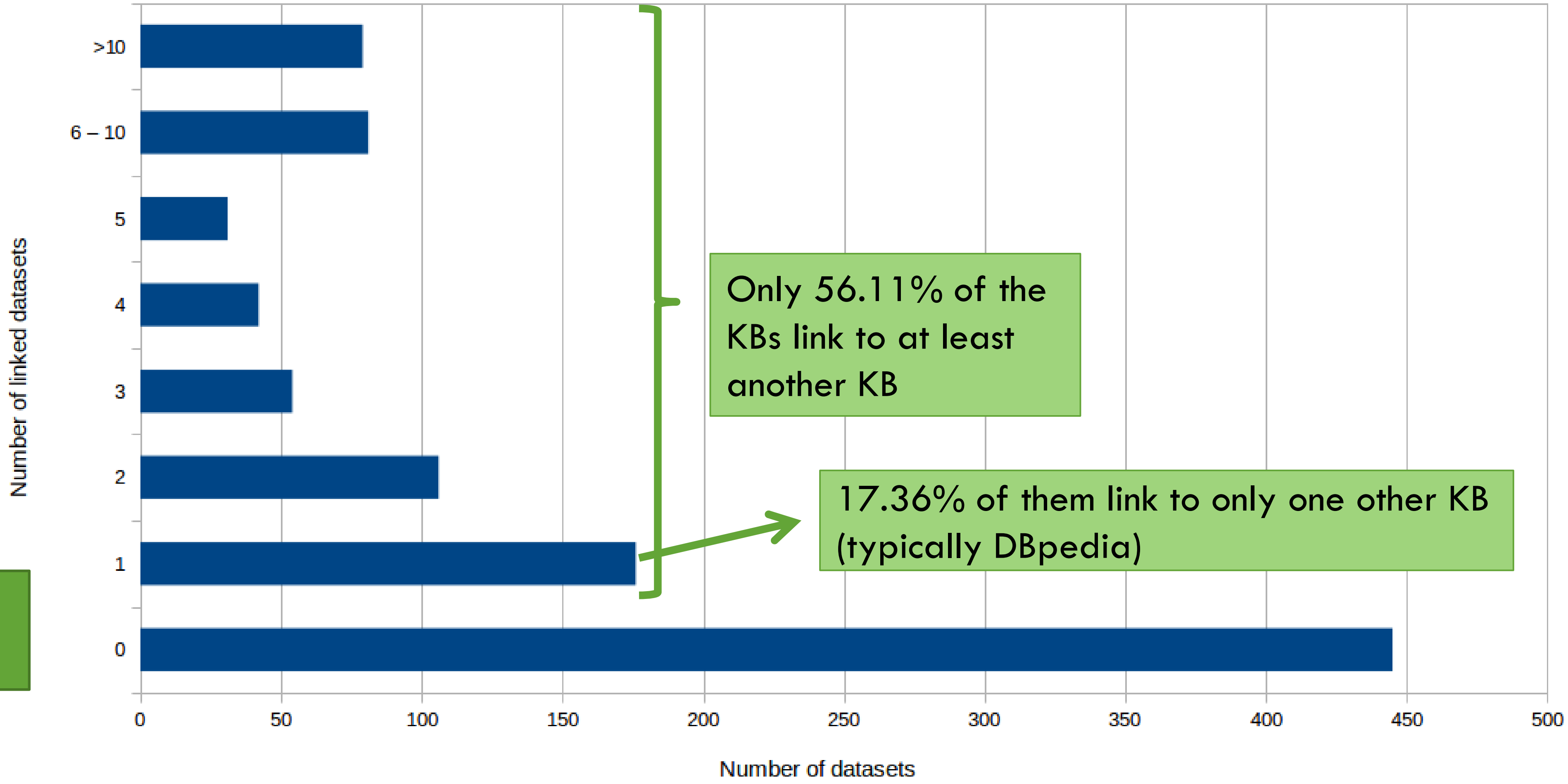


2014-08-30 http://lod-cloud.net/


MusicBrainz
SECOLD
Δι@ύγεια
διαφάνεια στο κράτος
BBC music
data.gov.uk
Working with UK Public Sector information and data
Linked Movie DataBase
GeoNames
DBpedia
dblp
computer science bibliography
acm
Freebase
yago
select knowledge
United States Census 2000
UniProt
Linked GeoData
Publications
Life Sciences
Cross-Domain
Social Networking
Geographic
Government
Media
User-Generated Content
Linguistics
Linked Datasets as of August 2014


English version
52 entity types
503K entities
6M triples
685 entity types
4.58 M entities
5.83 M triples
2.795 K properties
123 entity types
1.9 B entities
9.6 B triples
112 properties
1.5 K entity types
46.3 M entities
2.67 B triples
4 K properties
350 K entity types
10M entities
120M triples
100 properties
Linked Movie Database
DBpedia
Freebase
yago select knowledge
UniProt
Geo Names
Publications
Life Sciences
Cross-Domain
Social Networking
Geographic
Government
Media
User-Generated Content
Linguistics
Linked Datasets as of August 2014

ENTITY INTERLINKING IN LOD



The LOD cloud diagram is **sparse**

DESCRIPTIONS QUALITY AND 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

FORMS OF OVERLAPPING

Among KBs (inter-duplicates, due to common data sources)

Not identical descriptions, even if they have the same source

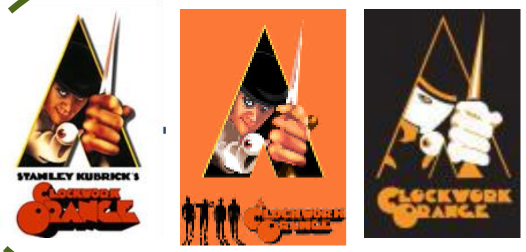
Within the same KB (intra-duplicates, due to wrong integration or bad curation)

- dbpedia:Dichopogon_strictus and dbpedia:Chocolate_lily refer to the same flower
- Less often than inter-duplicates

ENTITY RESOLUTION (ER)

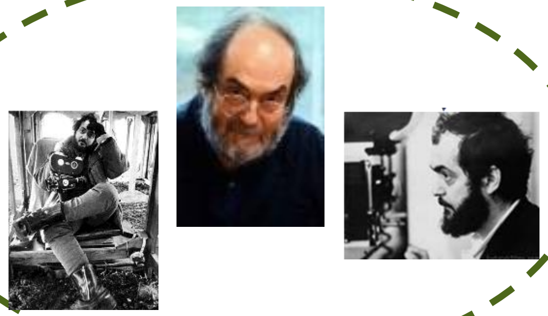
The problem of identifying descriptions of the same real-world entity

A Clockwork Orange



director

Stanley Kubrick



Images are descriptions, not real-world entities

- Descriptions are partial, incomplete

***highly* similar** descriptions

***somehow* similar** descriptions

HIGHLY & SOMEHOW SIMILAR DESCRIPTIONS

Highly Similar

- Feature many common tokens in the values of semantically related attributes
- Heavily interlinked
  - Mostly using *owl:sameAs* predicates
- Good for fusing
- Typically met in central KBs
  - Extracted from common sources

Somehow Similar

- Feature significantly fewer common tokens in attributes that are not always semantically related
- Sparsely interlinked
  - Using various kinds of predicates
- Good for linking
- Typically met in peripheral KBs
  - Extracted from various sources

HOW DOES ER IMPROVE 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

CHALLENGES OF WEB-SCALE ER

ER has been studied for many years in different cs communities, but it still remains active!

The problem has enjoyed a renaissance recently, due to the many descriptions of entities provided on the Web by government, scientific, corporate or even user-crafted KBs

How can we: *i) effectively* compute the entity similarity, *ii) efficiently* resolve *single or sets of entities*

are challenged by the:

important number of KBs (~ hundreds)

large number of entity types & properties (~ thousands)

massive volume of entities (~millions)

Large-scale, multi-type, cross-domain ER: Big Data Volume, Variety, Veracity

ER DEFINITION

Entity resolution: The problem of identifying descriptions of the same entity within or across sources

- $E = \{e_1, \ldots, e_m\}$ 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 $P = \{p_1, \ldots, p_n\}$ 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

ER EXAMPLE

dbpedia:Stanley_Kubrick	
dbo:birth Place	**dbpedia:Manhattan**
rdf:type	foaf:Person
rdf:type	yago:AmericanFilmDirectors
rdf:type	yago:Amateur ChessPlayers

e1

e3

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

Imdb: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

e4

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

ER EXAMPLE

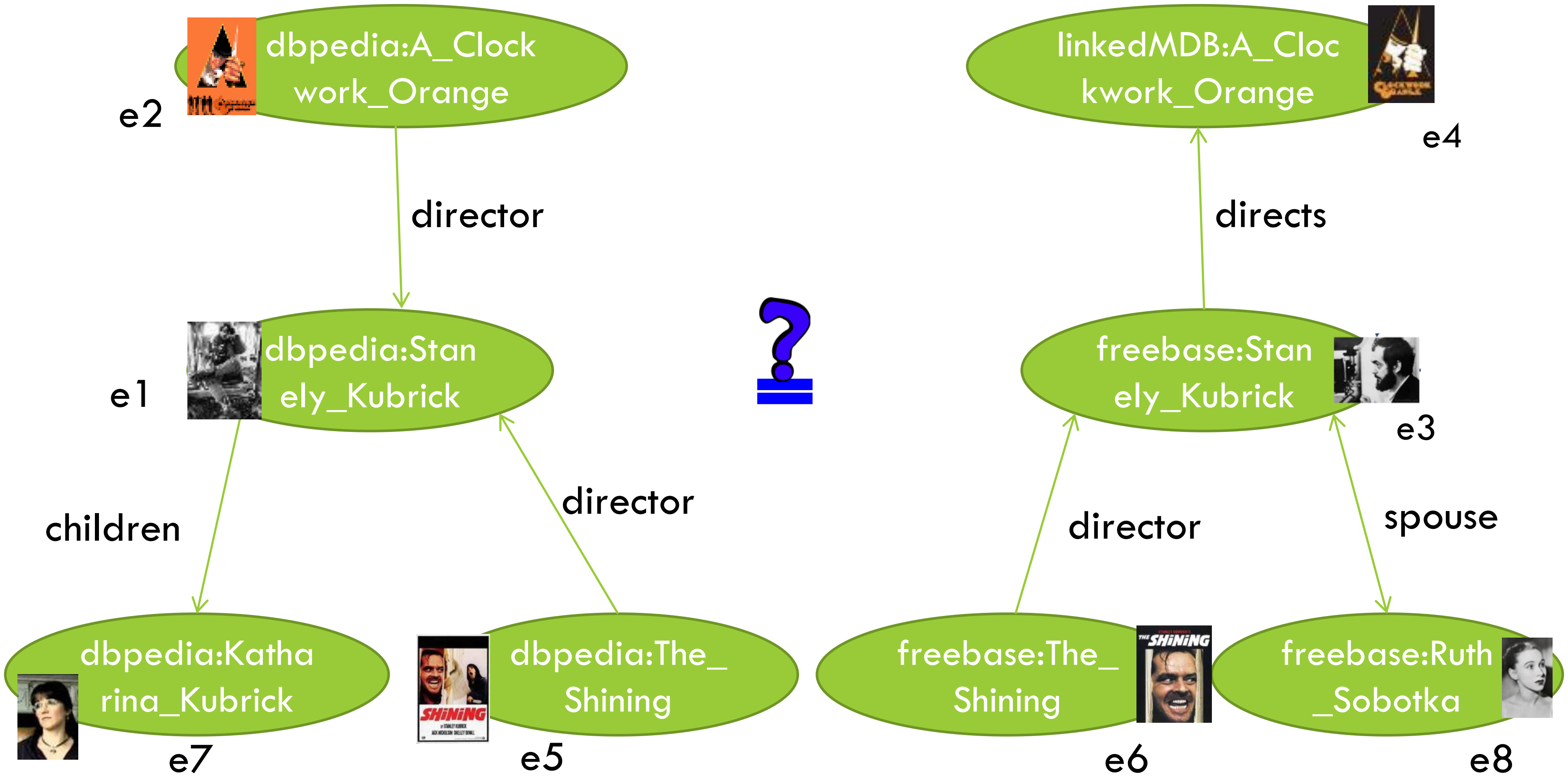
Assume as input of entity resolution, the set $E = \{e_1, e_2, e_3, e_4, e_5\}$

A possible output $P = \{\{e_1, e_3\}, \{e_2, e_4\}, \{e_5\}\}$ indicates that:

- e_1, e_3 refer to the same real-world person, the director Stanley Kubrick
- e_2, e_3 represent a different entity, the movie A Clockwork Orange
- e_5 represents a third thing, the movie studio PineWood



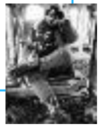

MATCHING GRAPH-STRUCTURED ENTITIES



SINGLE (PAIRWISE) ENTITY MATCHING BASED ON CONTENT

Matching decisions are independent

e1	
birthPlace	**Manhattan**
type	Person
type	AmericanFilmDirectors
type	Amateur ChessPlayers



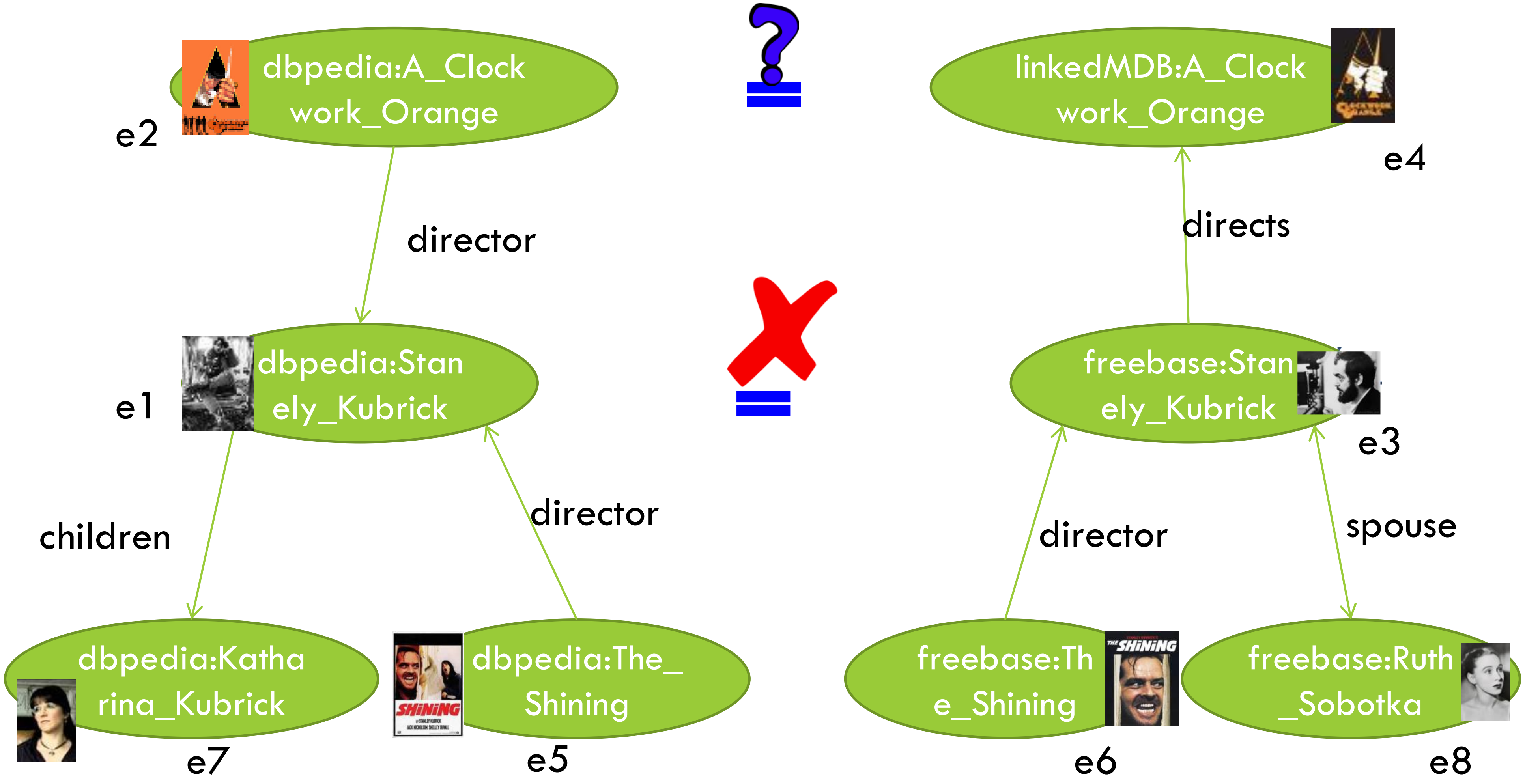
sim_c(e1,e3) = Jaccard ({Manhattan, Person, AmericanFilmDirectors, AmateurChessPlayers}, {Stanley, Kubrick, Person, 1894, 2014, 2685}) = **0.1**

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 STRUCTURE

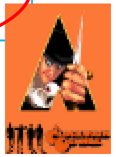
One matching provides evidence for another



MATCHING NEIGHBORHOOD

e2

director	**<u>e1</u>**
Work/ runtime	"136"
name	"A Clockwork Orange"



sim$_c$(e2,e4)= Jaccard (
{e1, 136, A, Clockwork, Orange},
{e3, 136, A, Clockwork, Orange}) = **0.66**

thresh = 0.5 ✓

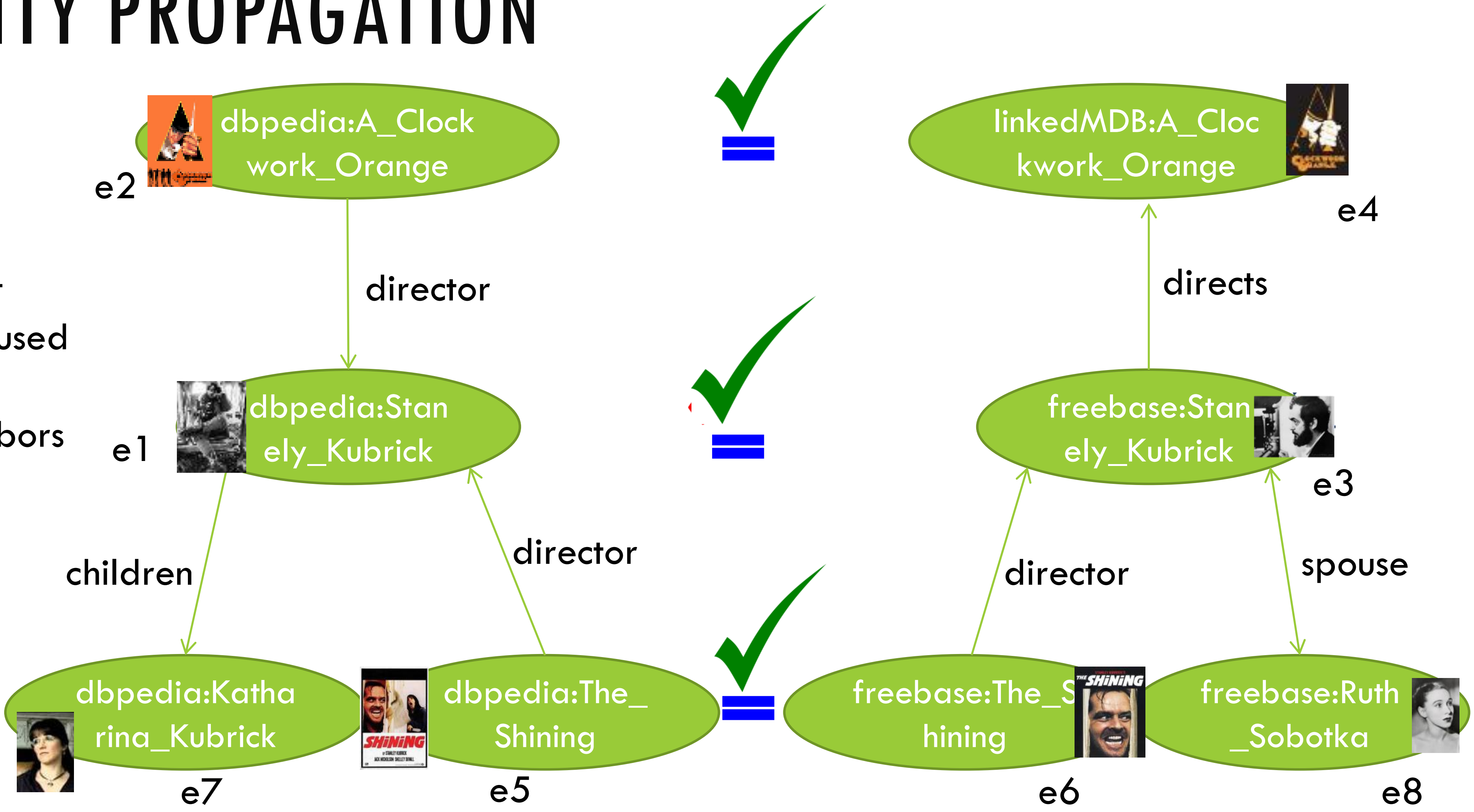
e4

film.directedBy	**<u>e3</u>**
film_cut/runtime	"136"
name	"A Clockwork Orange"

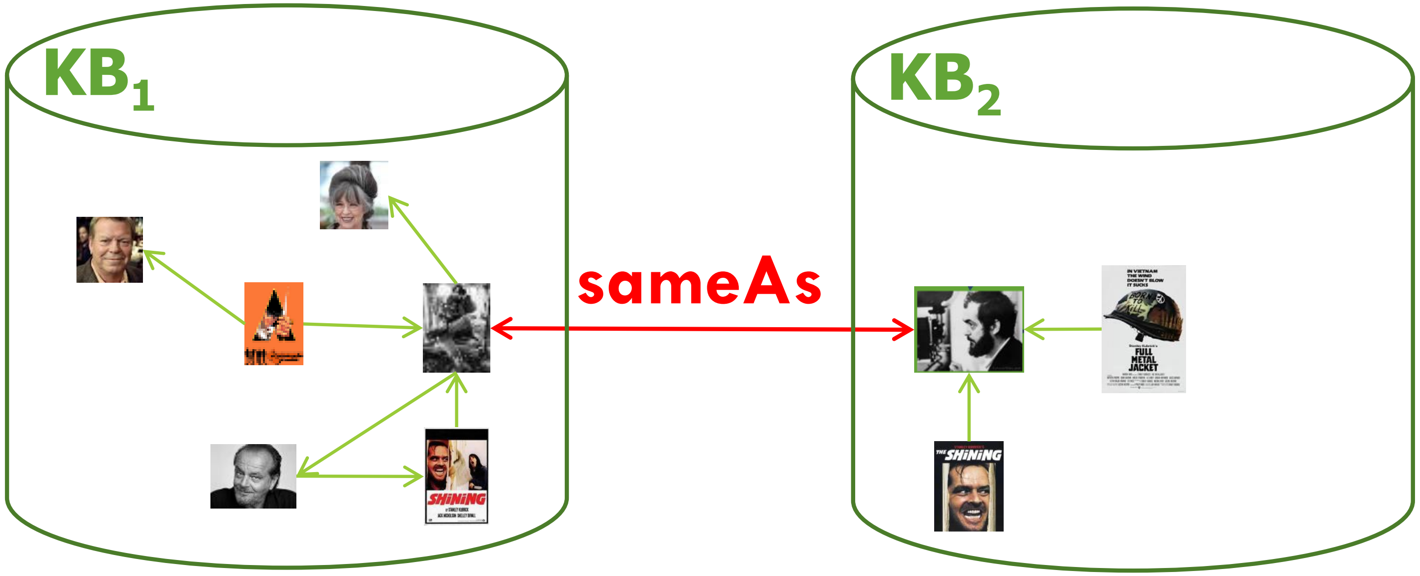
SIMILARITY PROPAGATION

A weighted sum of content and structural similarity is used

Infrequent matching neighbors contribute more to the similarity score

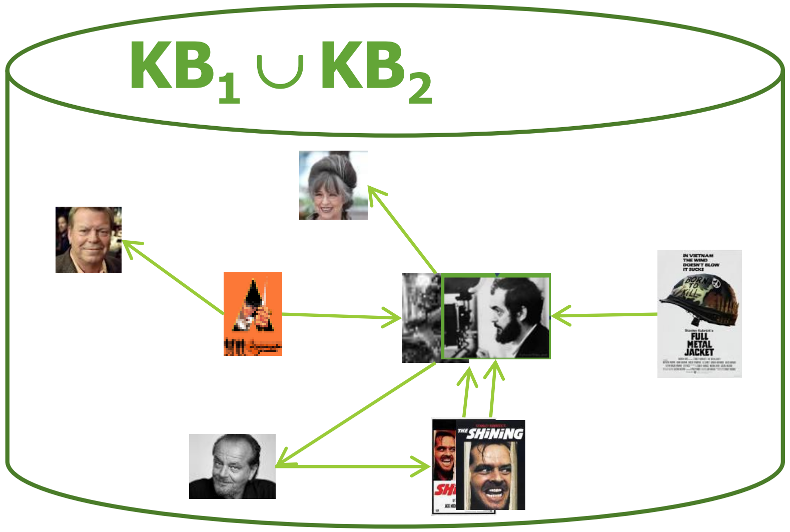


FORMS OF ER



[Record Linkage]: ER without results merging

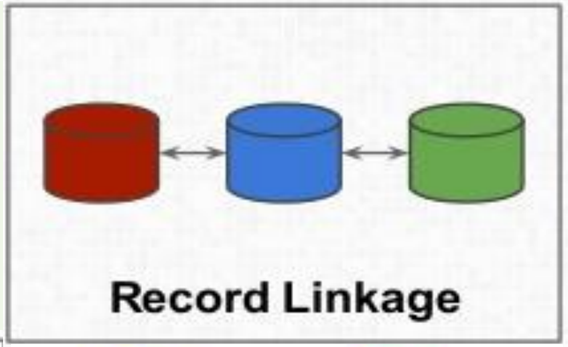
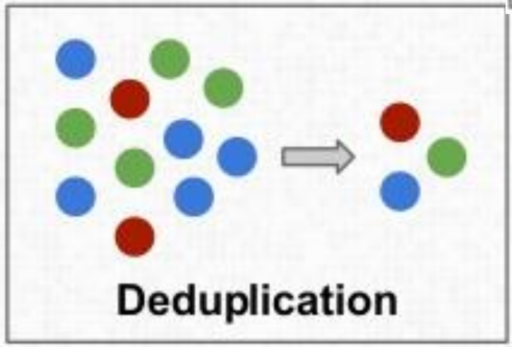
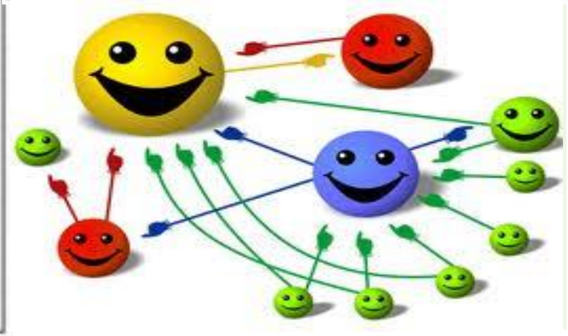
- Exploit exclusivity of matches



Record Deduplication: ER with results merging

- Exploit transitivity of matches

FORMS OF ER & SIMILARITY

	High similarity in structure	Low similarity in structure
High similarity in content		
Low similarity in content		

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

The definition of what is similar is domain-dependent

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

att & value sim. in a network of relations

SCOPE OF THE TUTORIAL

Describing and Linking Entities

- Knowledge Bases, The Web of Data, Entity Resolution

Matching and Resolving Entities

- Entity Similarity (Content & Context)
- Blocking Techniques (Token, Attribute, URI)
- Block Post-Processing
- Iterative Resolution Techniques
- Progressive Resolution Techniques
- Conclusions & Open Issues

ENTITY SIMILARITY

ENTITY SIMILARITY - MATCH

Matches: Sets of entity descriptions that refer to the same real-world entity:

- Matching descriptions are placed in the same partition
- All the descriptions of the same partition match

Finding matches vs non-matches is a classification problem

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

Imbalanced: typically, O(E) matches, O(E^2) non-matches

MATCH FUNCTION: FORMAL PROPERTIES

The match function M() introduces an equivalence relation (owl:sameAs) among entity descriptions:

- Reflexivity: $\forall e_i \in E$, $M(e_i, e_i) = true$
- 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 $M(e_i, e_k) = true$

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 [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

although not identical e_2 and e_4 are highly similar

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	Imdb:director/8476
fbase:film_cut/runtime	'136'
foaf:name	"A Clockwork Orange"

e_1 and e_3 are at best somehow similar

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

e1

e3

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

Entity Matching: Relies on a similarity function, the higher the similarity of two descriptions, the more likely it is that they match

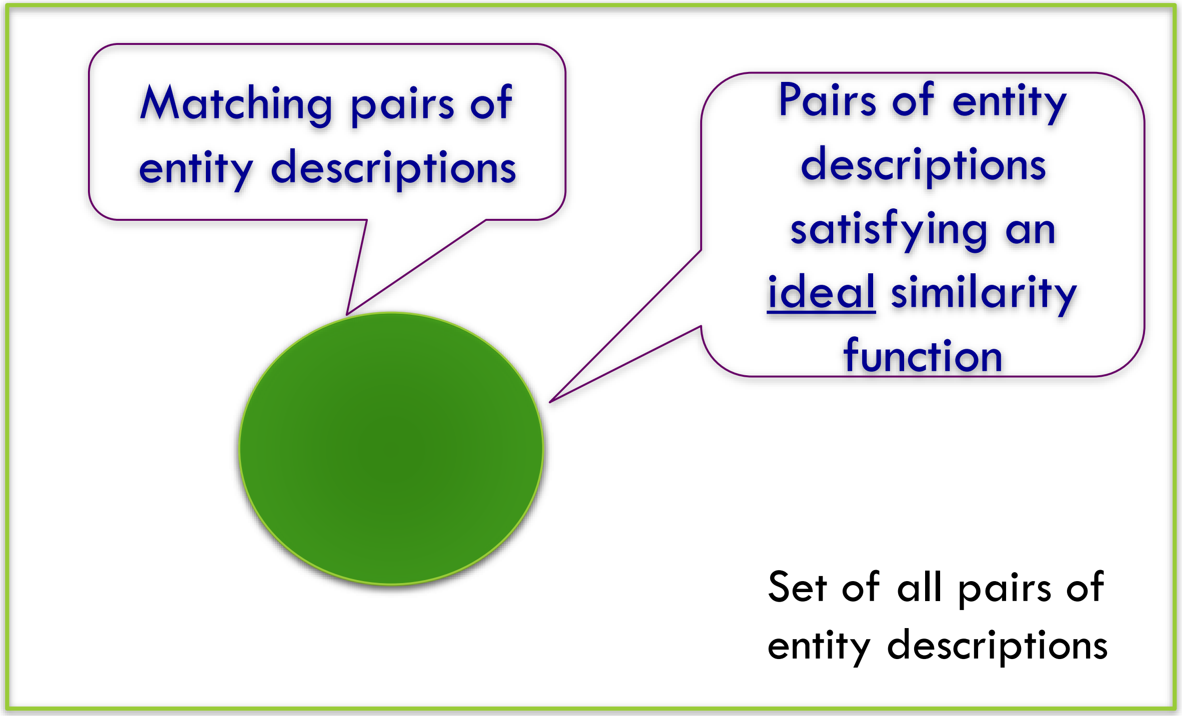
- Content : standalone comparisons between entities based on the values of their attributes
- Context: graph-based comparisons between entities based on their relationships

THE ROLE OF SIMILARITY FUNCTIONS: THE IDEAL CASE

Intuitively, the higher the similarity of two descriptions, the more likely it is that they match

- The similarity of two descriptions is used as a hint for their matching

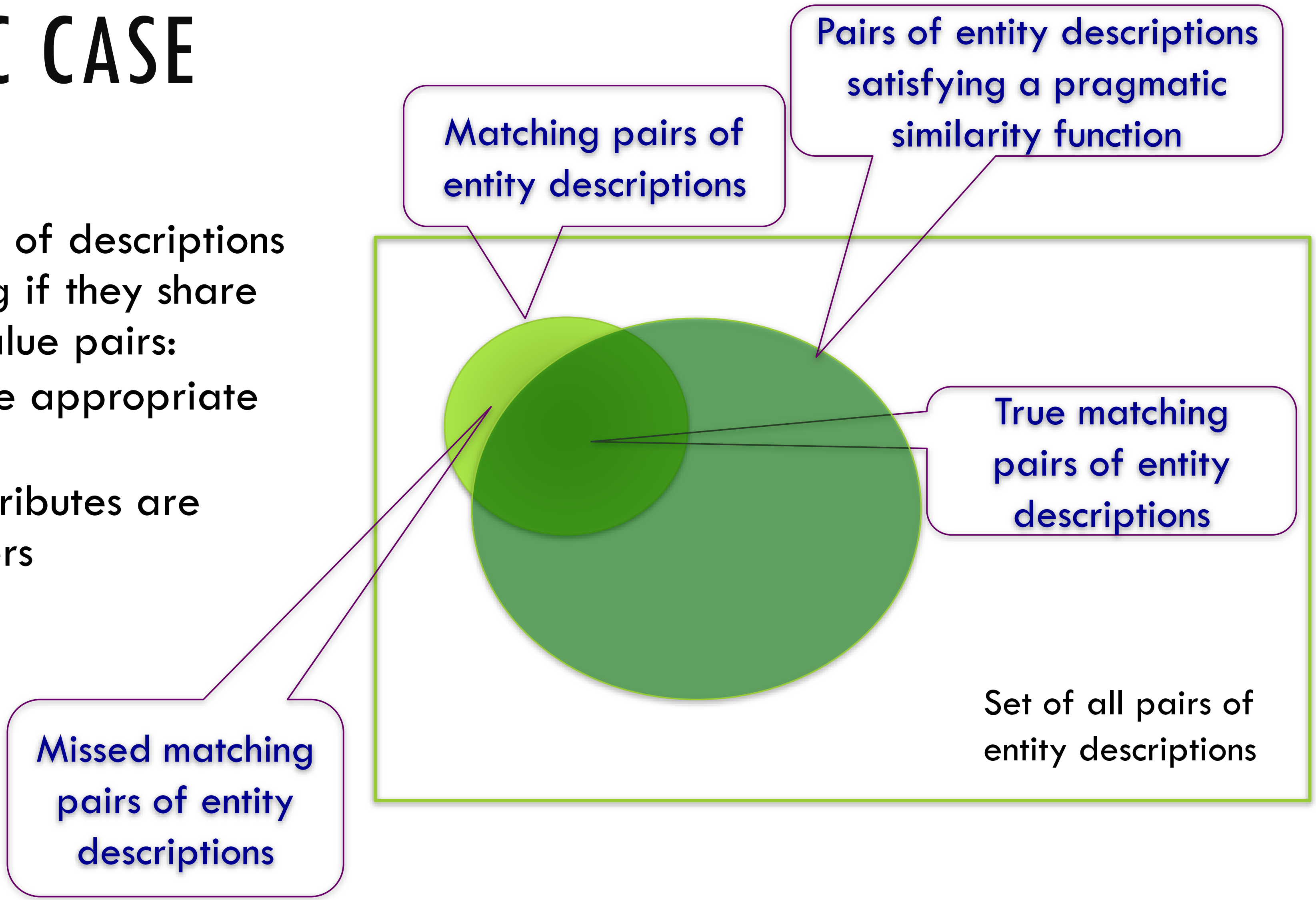
There is no general way of determining which attributes should count as salient in determining matching entity descriptions



A PRAGMATIC CASE

[Hogan et al., 2010]: a pair of descriptions is more likely to be matching if they share several common attribute-value pairs:

- Certain attributes are more appropriate to determine matches
- Certain values of these attributes are more discriminant than others



CONTENT-BASED ENTITY SIMILARITY

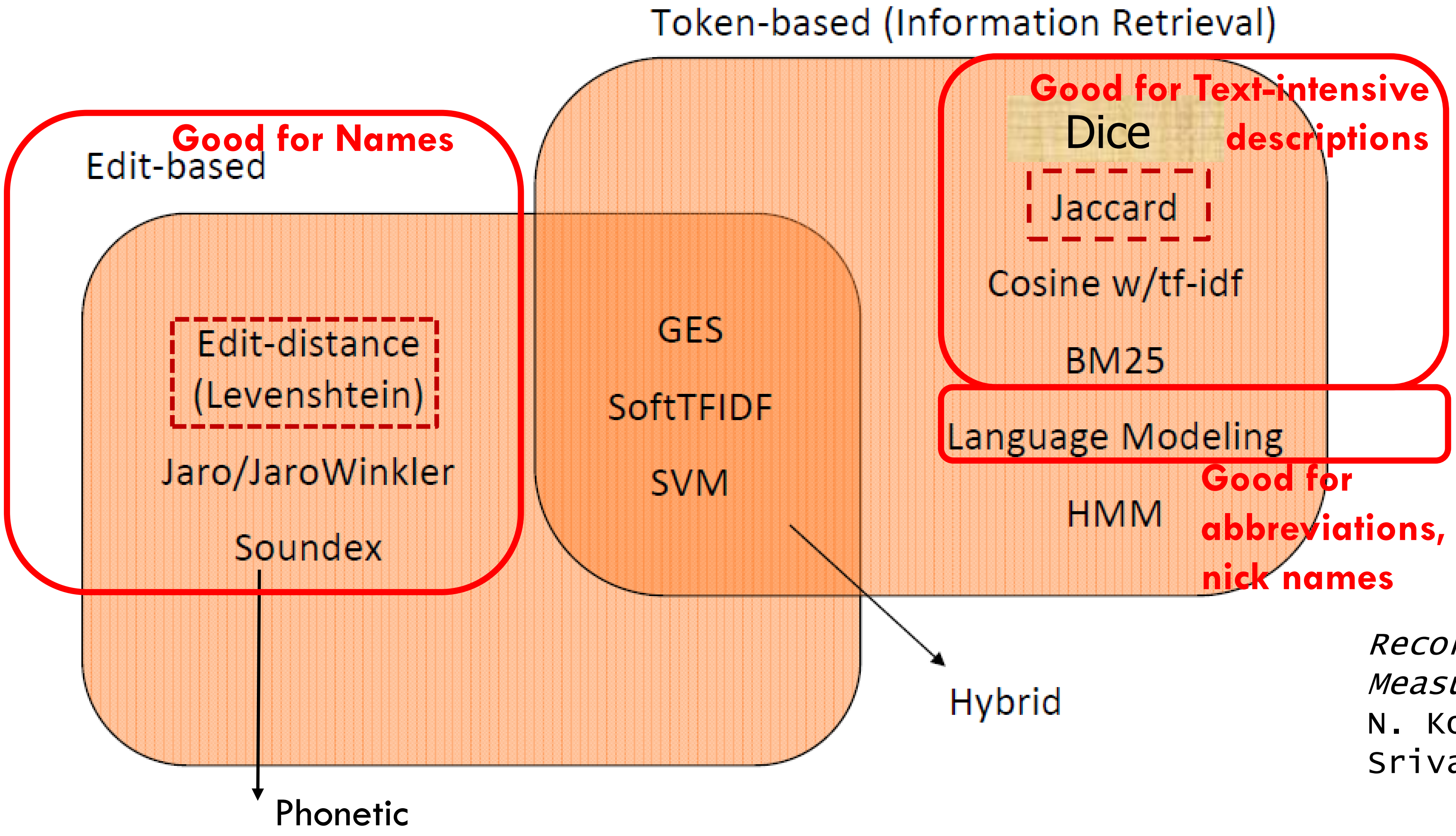
IN SEARCH OF ENTITY SIMILARITY MEASURES

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 neighbour descriptions via different entity relationships
  - Tree-based, graph-based

STRING SIMILARITY MEASURES



Record Linkage: Similarity Measures and Algorithms
N. Koudas S. Sarawagi D. Srivastava, SIGMOD06 Tutorial

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

name	White Tower	
location	Thessaloniki	
year-constructed	1450	e5

about	Eiffel Tower	
architect	Sauvestre	
year	1889	
located	Paris	e4

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

$\text{Jaccard}(e_1,e_3) = 1/8$ $\quad$ $\text{Jaccard}(e_2,e_3) = 3/7$

$\text{Jaccard}(e_1,e_4) = 1$ $\quad$ $\text{Jaccard}(e_2,e_4) = 1/11$

$\text{Jaccard}(e_1,e_5) = 1/8$ $\quad$ $\text{Jaccard}(e_2,e_5) = 0/11$

CONTEXT-BASED ENTITY SIMILARITY

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 neighbours 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') + β*context_sim(C(e),C(e'))−θ

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

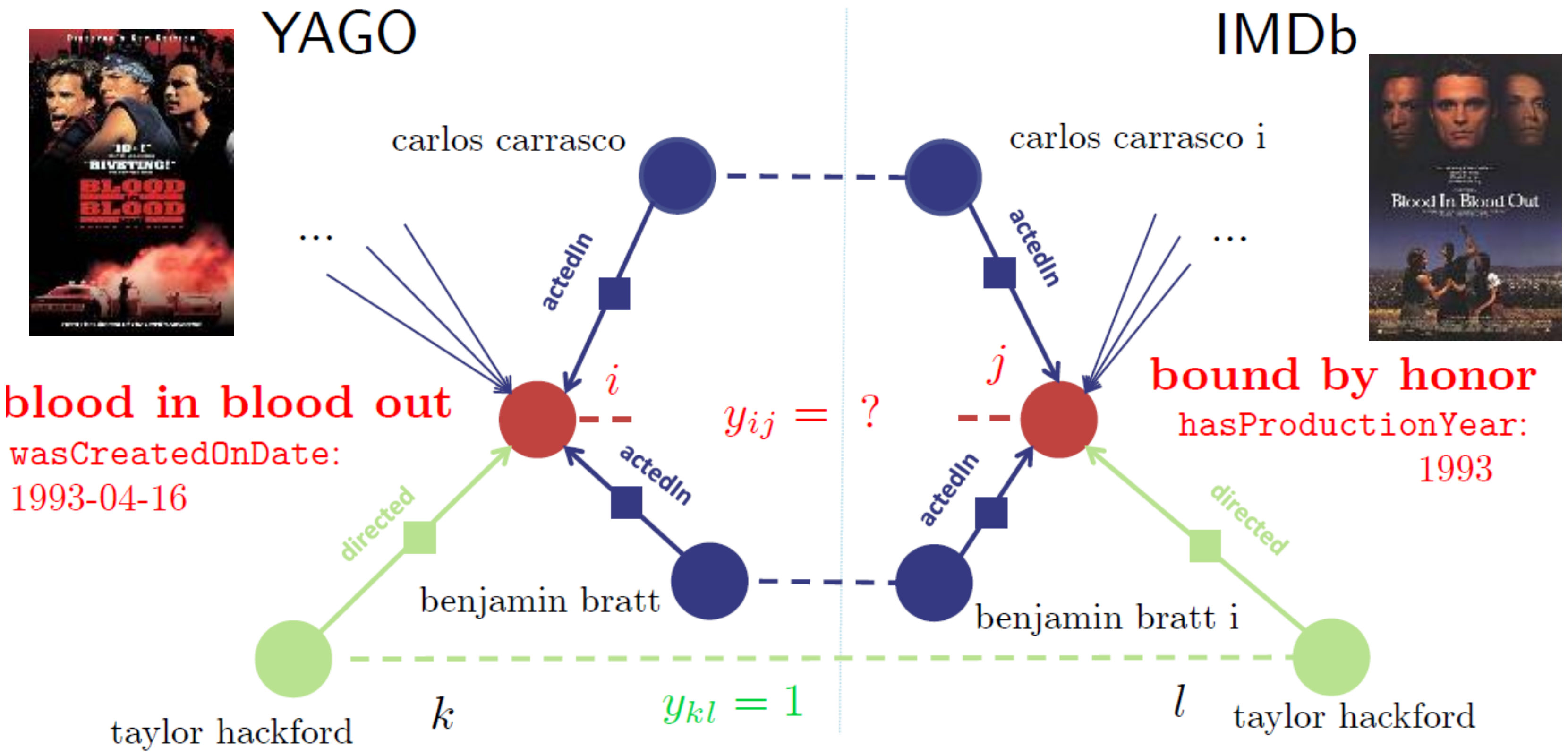
More common tokens & common neighbours that two descriptions have, the more likely they are to match

CONTENT & STRUCTURE SIMILARITY SIGMA [LACOST-JULIEN ET AL 2013]

Entities i and j have no tokens in common

The fact that several of their neighbors are matched together is an evidence that i and j should be matched together

- Use neighbors for scoring and suggesting candidate pairs

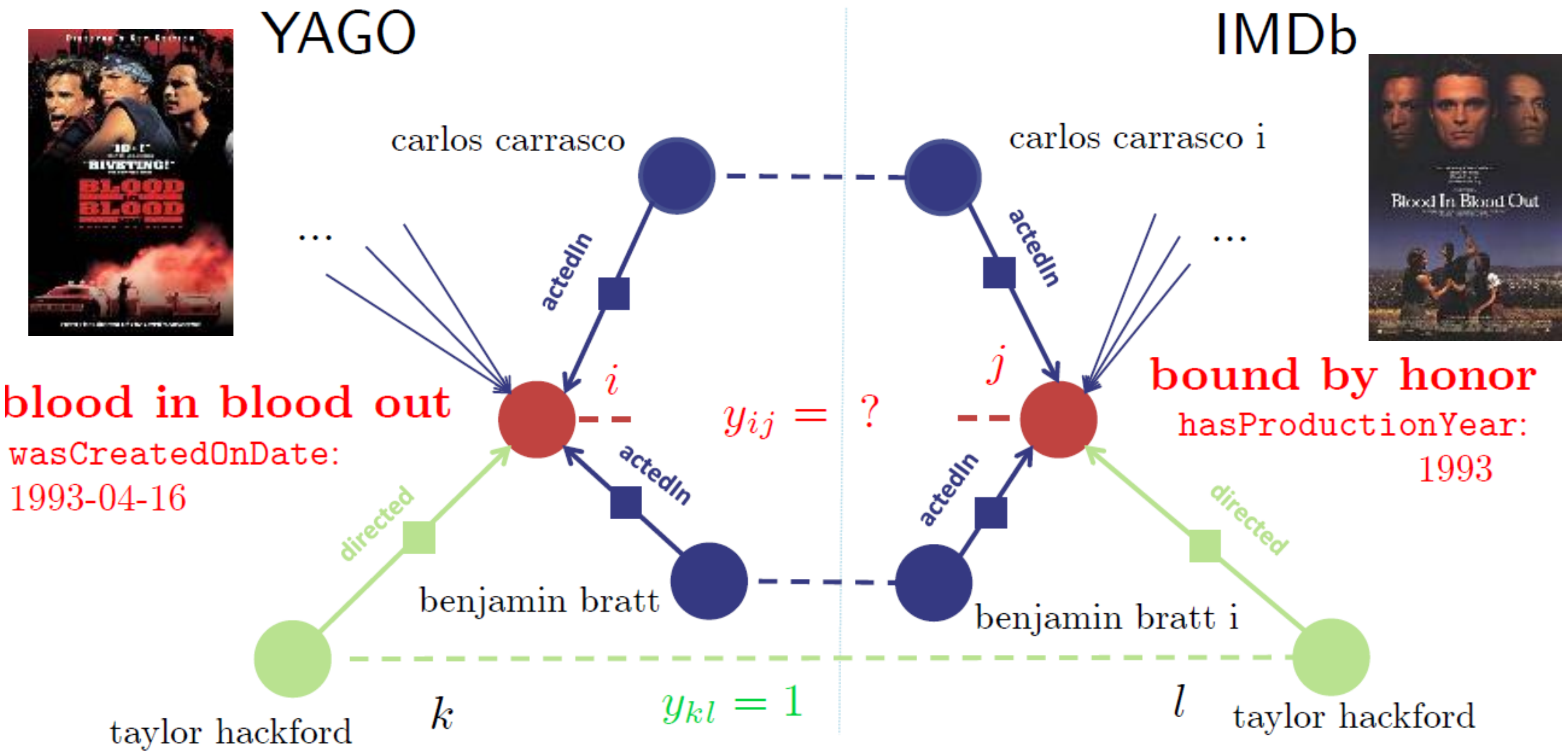


SiGMa: a scalable greedy iterative algorithm that exploits previous matching decisions as well as the relationship graph information between entities

SIGMA NEIGHBOURS 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}



Properties matching is provided by the users

String similarity: weighted Jaccard (IDF-like)

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)

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

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

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

SIGMA SIMILARITY MEASURES

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

IN SEARCH OF ENTITY SIMILARITY MEASURES

Defining ideal similarity measures is difficult, calls for more pragmatic approaches

- For highly similar entities content similarity (i.e., their attribute values) is sufficient
- For somehow similar entities we can consider the similarity of the structured context of entities in an iterative way
  - Identify *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: use schematic mappings provided by the users
  - Complex: assess similarity of attributes and relationships based on the similarity of their names or values

BLOCKING TECHNIQUES

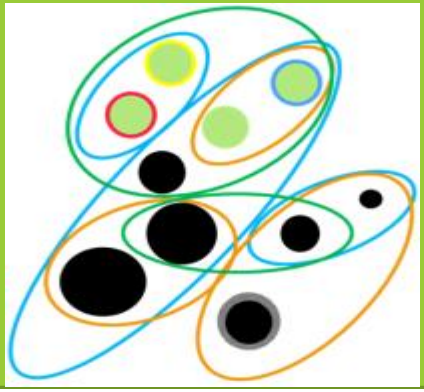
Reduce the number of comparisons not leading to resolved entities

$O(n)$
if hash-based

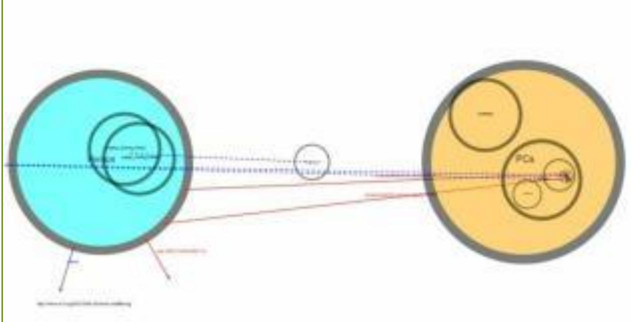
$O(n^2)$
n entity descriptions

$O(b \cdot d^2)$
b blocks of d descriptions (avg)

Blocking



Matching



Group similar enough entity descriptions

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 U 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)

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

Actually, an inverted index

name	White Tower
location	Thessaloniki
year-constructed	1450

e5

Generated Blocks

Eiffel	Tower	~~Statue~~	Liberty	~~White~~	1889	~~Bartholdi~~
e_1, e_2, e_3, e_4	e_1, e_4, e_5	$\cancel{e_2}$	e_2, e_3	$\cancel{e_5}$	e_1, e_4	$\cancel{e_2}$

NY	Paris	~~1886~~	~~1450~~	~~Lady~~	Sauvestre	~~Thessaloniki~~
e_2, e_3	e_1, e_4	$\cancel{e_2}$	$\cancel{e_5}$	$\cancel{e_3}$	e_1, e_4	$\cancel{e_5}$

ATTRIBUTE CLUSTERING [PAPADAKIS ET AL 2013]

Token blocking totally ignores the semantics of attributes

- When attribute mappings are not known, attribute clustering considers similarity of attributes 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 [PAPADAKIS ET AL 2013]

For each attribute of KB_1:

- Find the most similar attribute of KB_2

For each attribute of dataset KB_2:

- Find the most similar attribute of dataset KB_1

Compute the transitive closure of the generated pairs of attributes

Connected attributes form clusters

All single-member clusters are merged into a common 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

Finding the attribute of D2 that is most similar to the attribute "about" of D1:

values of about: {Eiffel, Tower, Statue, Liberty, Auguste, Bartholdi, Joan}

compared to (with Jaccard similarity on token sets) :

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

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

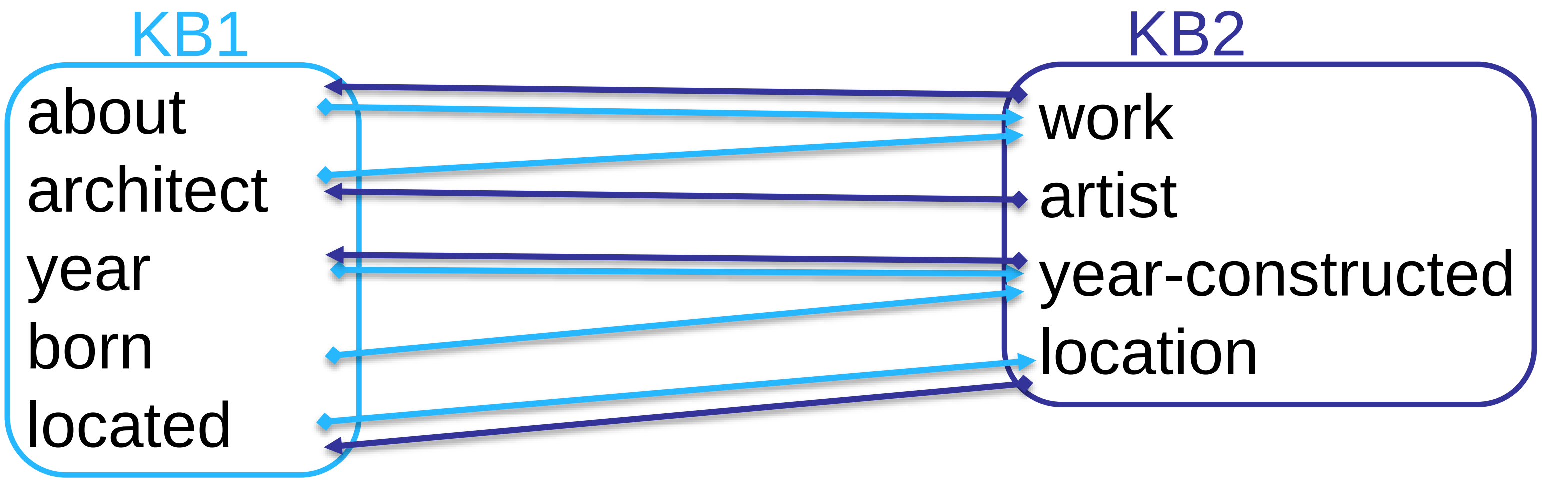
e15

work	Eiffel Tower
year-constructed	1889
location	Paris

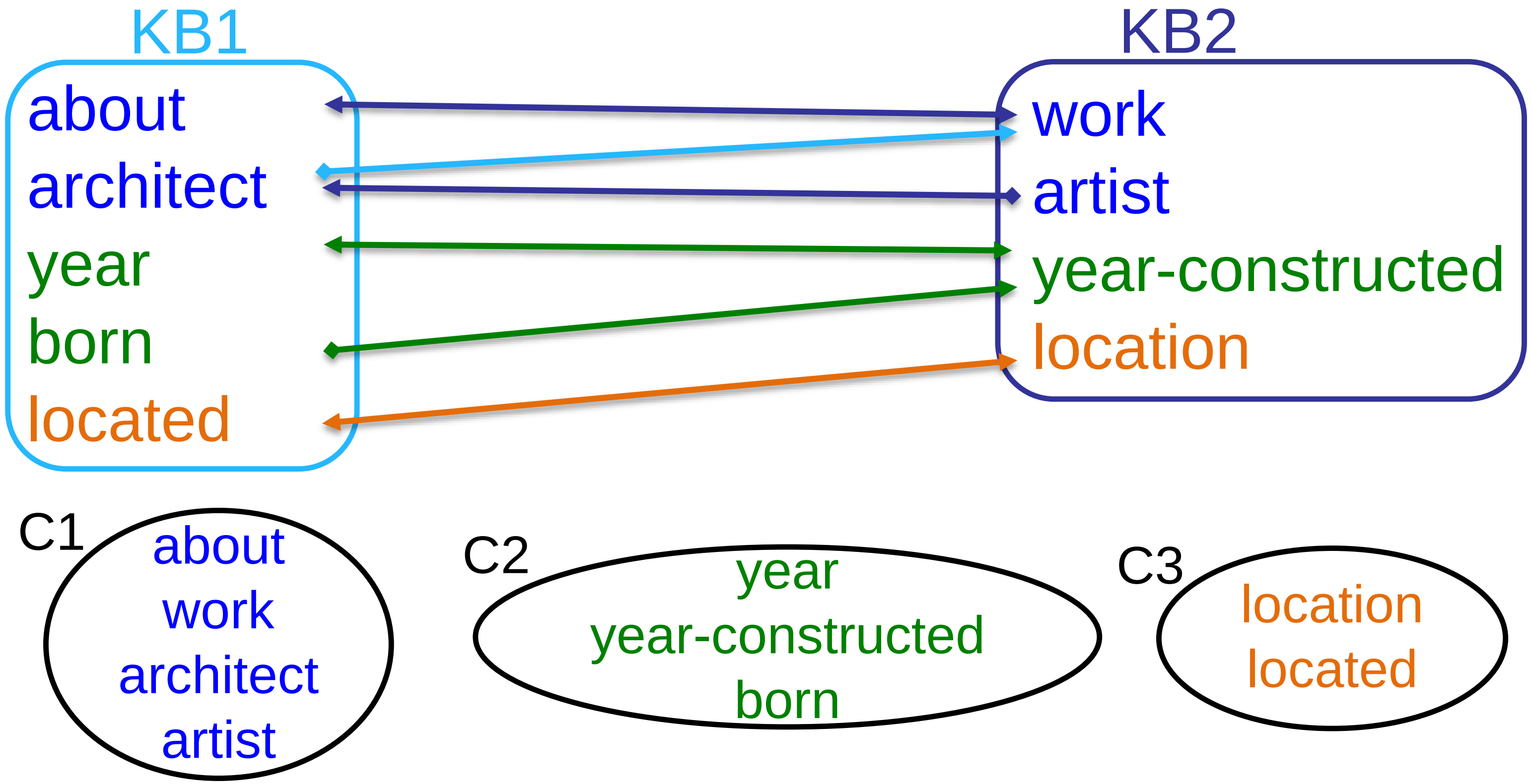
e16

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

e17



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



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	Washingt on D.C.

e17

C1
about
work
architect
artist

C2
year
year-constructed
born

C3
location
located

Works only when values are quite similar → attribute clusters contain similar attributes

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*

OTHER BLOCKING TECHNIQUES

Infix blocking: The blocking key is the URI infix of the entity description

- Example: http://en.wikipedia.org/wiki/Linked_data#Principles.html
  - Infix is a local identifier
- Its effectiveness relies on the good naming practices of the KBs publishing entity descriptions

Frequent itemsets blocking: Build blocks for sets of tokens that frequently co-occur in descriptions

- May significantly reduce the number of candidate pairs
- May significantly increase missed matches between descriptions with few common tokens

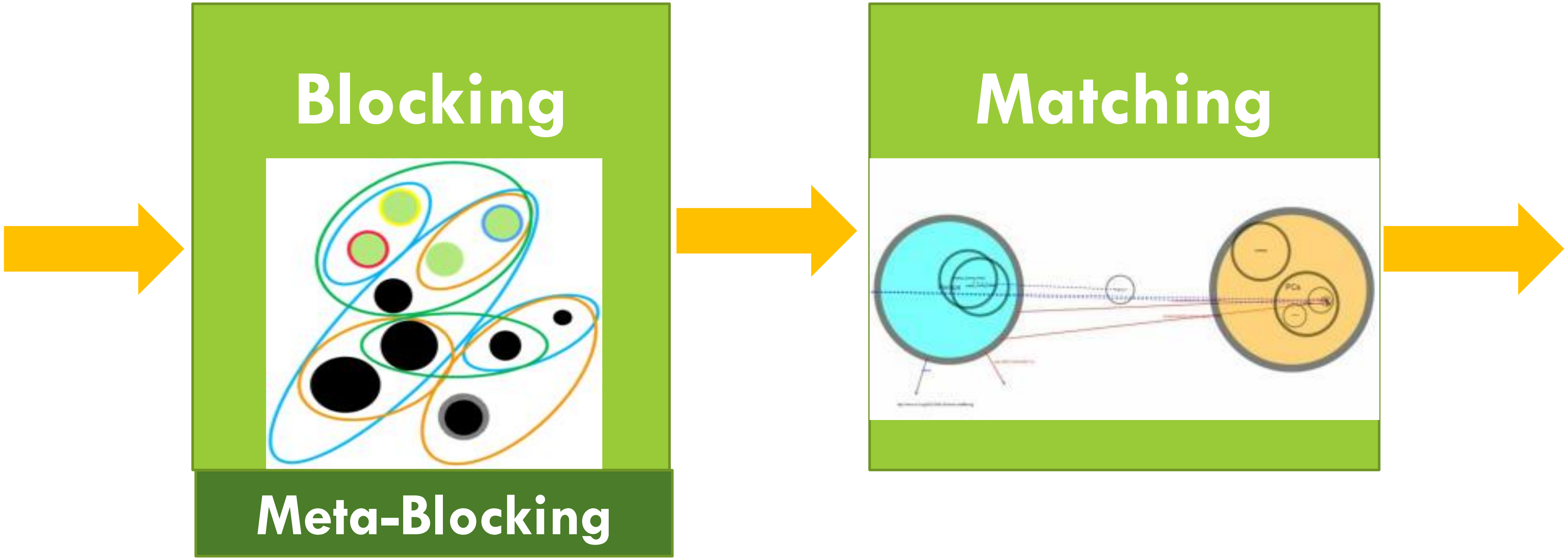
Multidimensional blocking: Construct a collection of blocks for each similarity function used to resolve entities and aggregate them into a single collection, taking into account the similarities of descriptions that share blocks

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
Frequent itemsets [Kenig and Gal, 2013]	The descriptions have frequently co-occurring tokens in their values

BLOCK POST-PROCESSING

META-BLOCKING: IMPROVE THE EFFICIENCY OF BLOCKING



<u>Goal:</u>

- Restructure a block collection into a new one that contains significantly fewer redundant and superfluous comparisons
- Maintaining the original number of matching ones

Blocks (ToB):

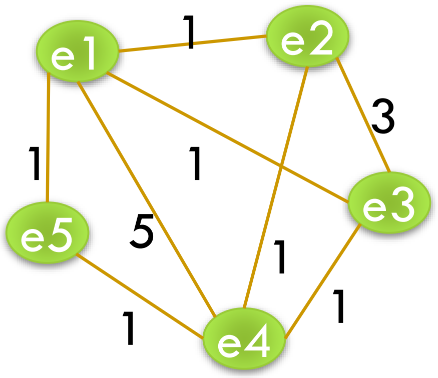
Eiffel	Tower	Liberty
e_1, e_4, e_2, e_3	e_1, e_4, e_5	e_2, e_3

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

Sauvestre
e_1, e_4

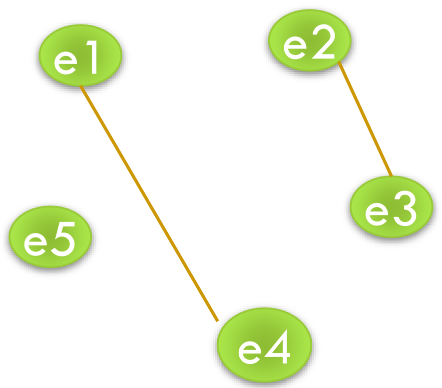
14 comparisons to identify 2 matches e_1-e_4 and e_2-e_3

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 & 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)

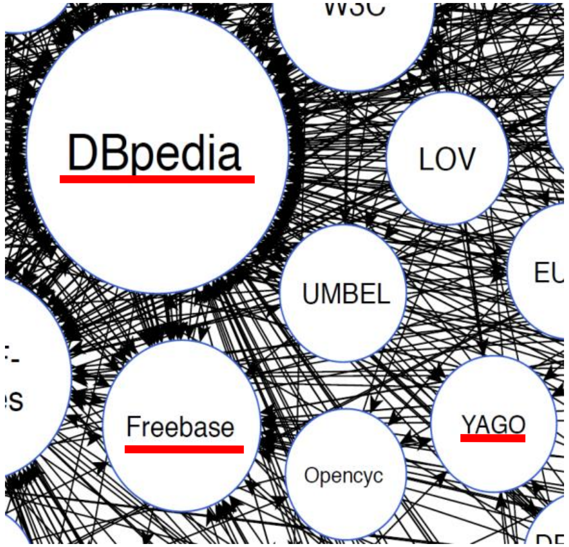
- WEP: Keep edges with weight above average
- CEP: Keep top-K edges overall
- WNP: Keep, for each node, the edges with weight above a local average
- CNP: Keep, for each node, its top-K edges

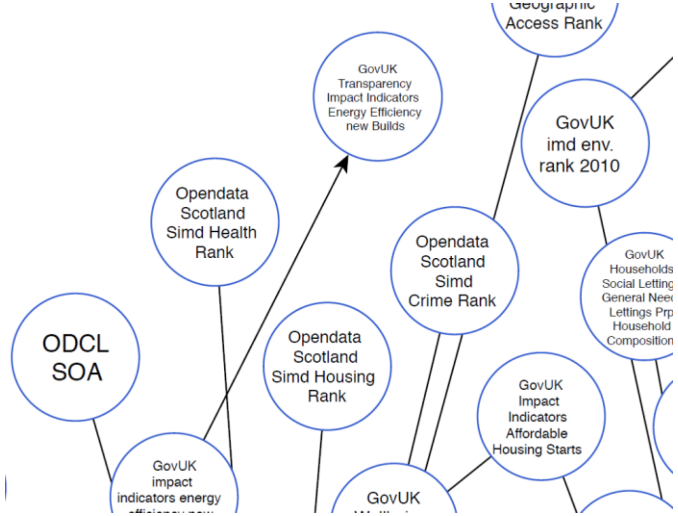
ITERATIVE RESOLUTION TECHNIQUES

CENTRAL VS. PERIPHERAL KBS

Zooming into the center of the LOD cloud, we can find KBs, such as Dbpedia and YAGO, containing millions of descriptions of thousands of different types, heavily interlinked

On the other hand, peripheral KBs are sparsely interlinked and they typically describe entities of vey specific types





IN SEARCH OF SIMILARITY EVIDENCE [EFTHYMIOU 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

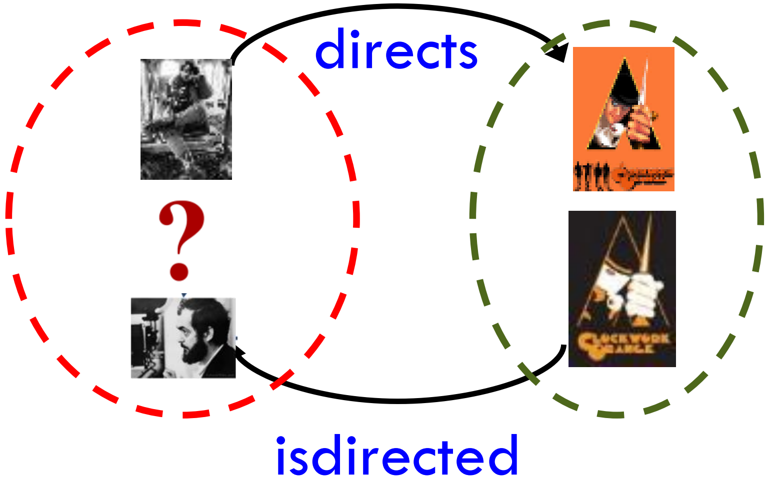
- Matching neighbors provide positive evidence
  - >92% of pairs with at least one matching neighbor, are matches in most KBs
- Some types of relationships provide strong negative evidence
  - Dissimilar values for `wasBornIn` indicate a non-matching pair

TYPES OF MISSED MATCHES

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

Applicable to identify matches within a KB

- Type B: matches of their neighbours

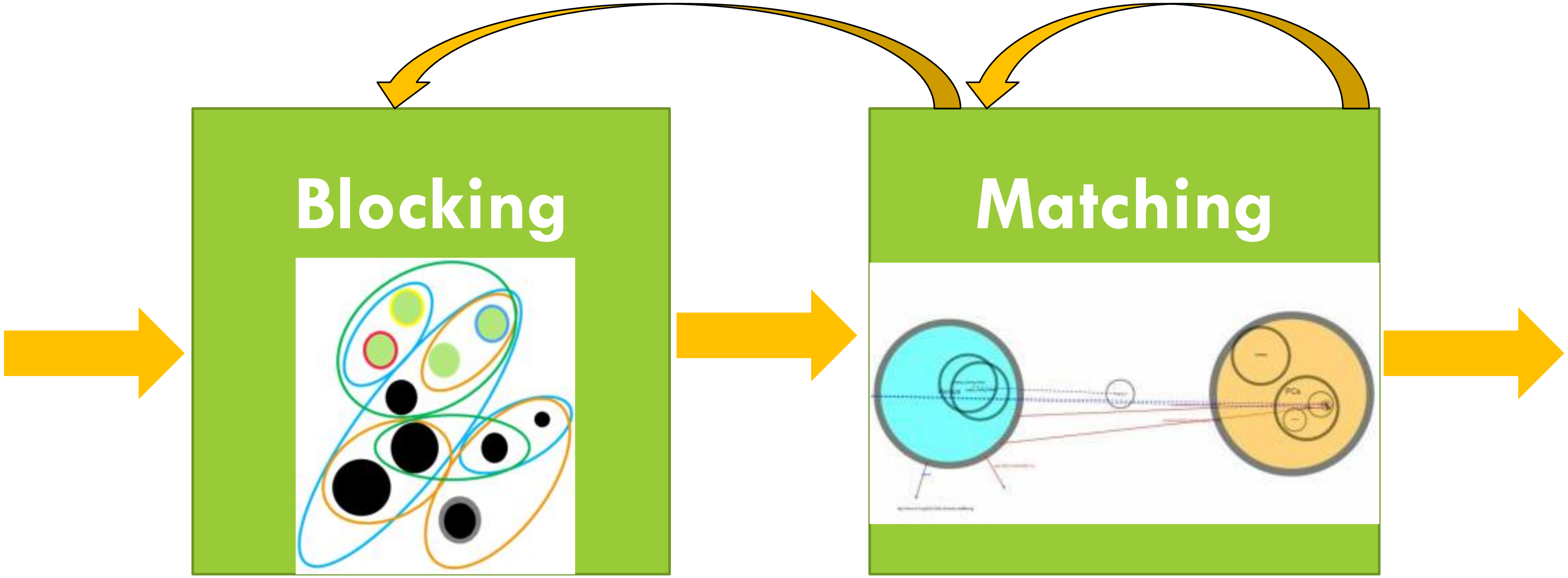


Can identify matches both within a KB and across different KBs

ITERATIVE ER

Generate new candidate pairs of descriptions not considered in a previous step

- Several passes increase the number of comparisons and reduce the Reduction Ratio



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

Increase the number of matching entities

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)
  - 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-BASED ITERATIVE RESOLUTION

SIMILARITY PROPAGATION

A graph structure for encoding the similarity between 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 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 neighbourhood nodes of e

ORDER OF COMPARISONS

In similarity-propagation approaches, the order of comparisons is dynamic

Graph traversal is 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:

- 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]

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 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 of a node is above a threshold θ

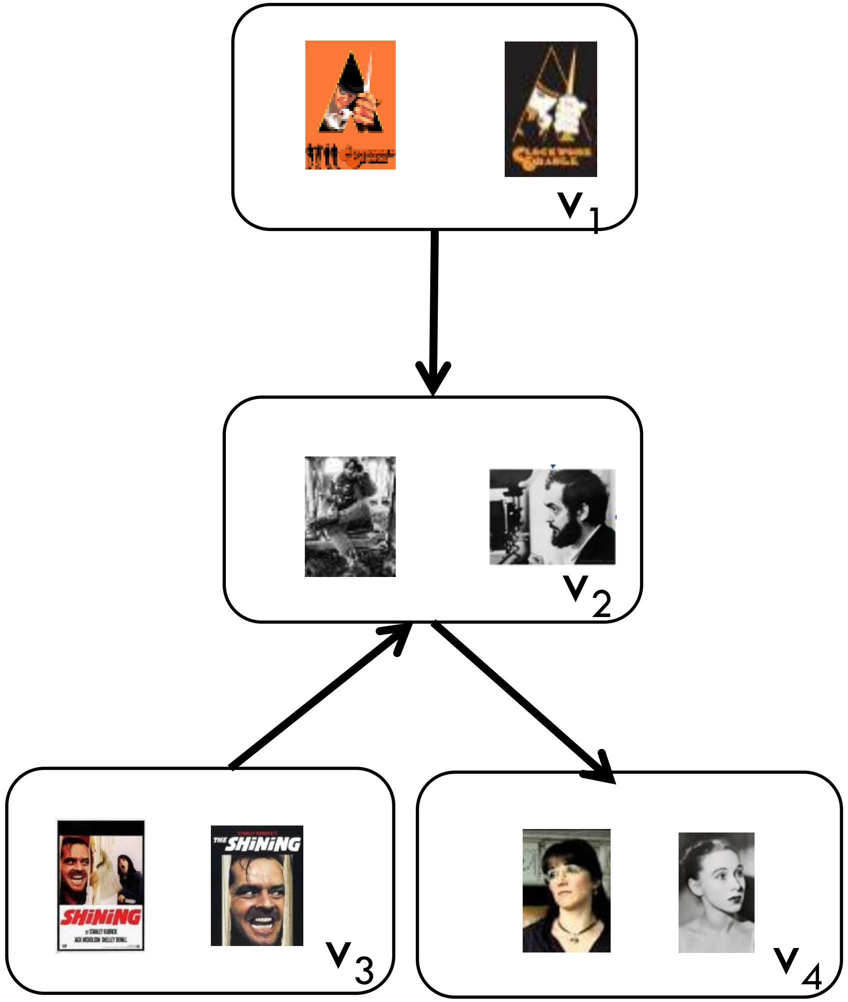
- If it exceeds the threshold, it is marked as match, otherwise as undecided
- If no more neighbors are undecided, it is marked as non-match

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} , $e_{bj} \in values(e_{ai}) \cup values(e_{aj})$

Include nodes whose two entities have the potential to be similar



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

TRAVERSING THE ER GRAPH

Nodes can be active, merged or inactive

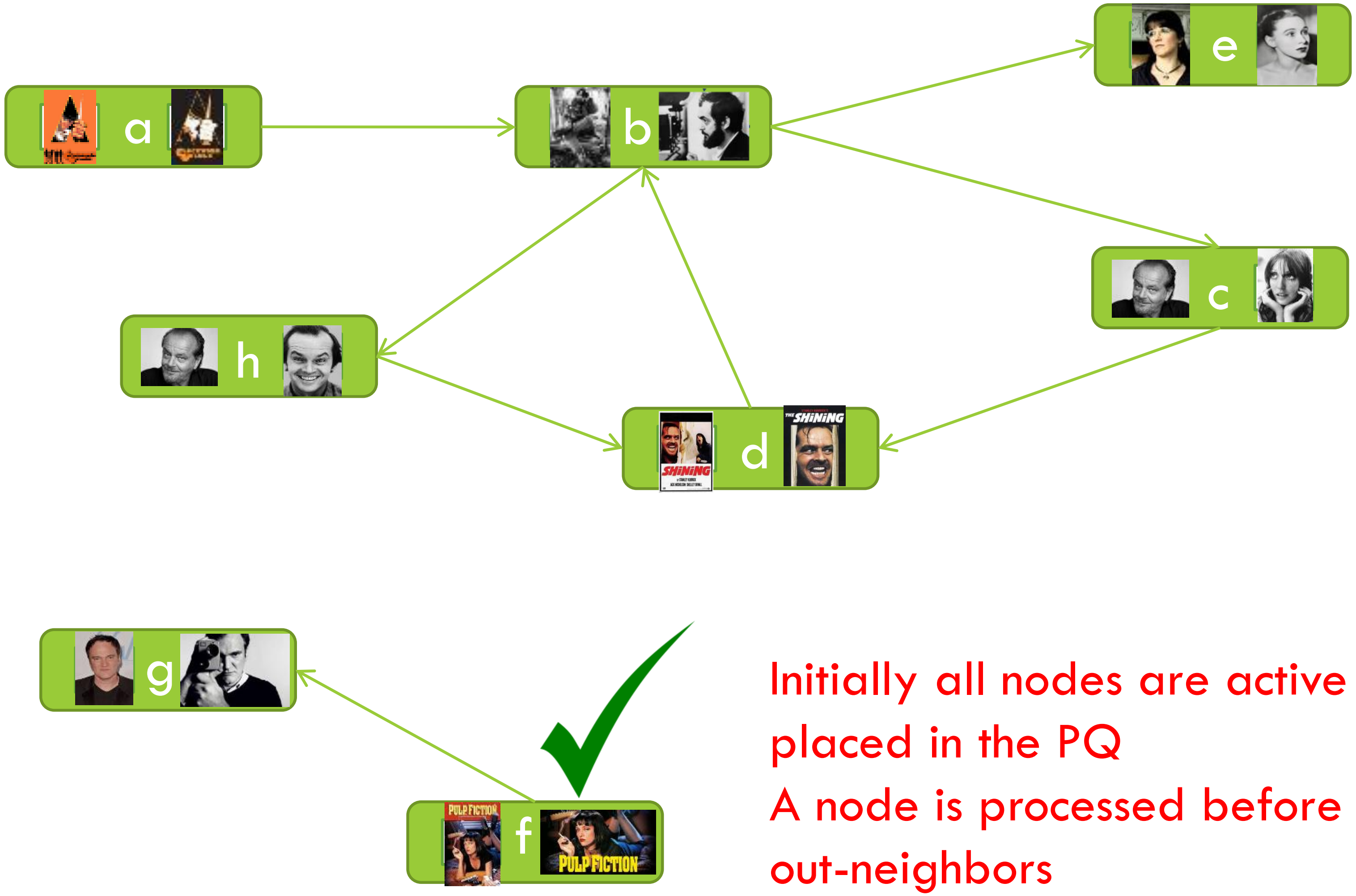
At each iteration step, the node in the head of the PQ is processed and its similarity is assessed (i.e., update its similarity)

If the similarly is above the threshold then it becomes merged, otherwise inactive

- In both cases, the node is removed from the PQ

If the updated similarity increase its similarity then all its inactive out-neighbors become active and inserted at PQ

TRAVERSING THE ER GRAPH

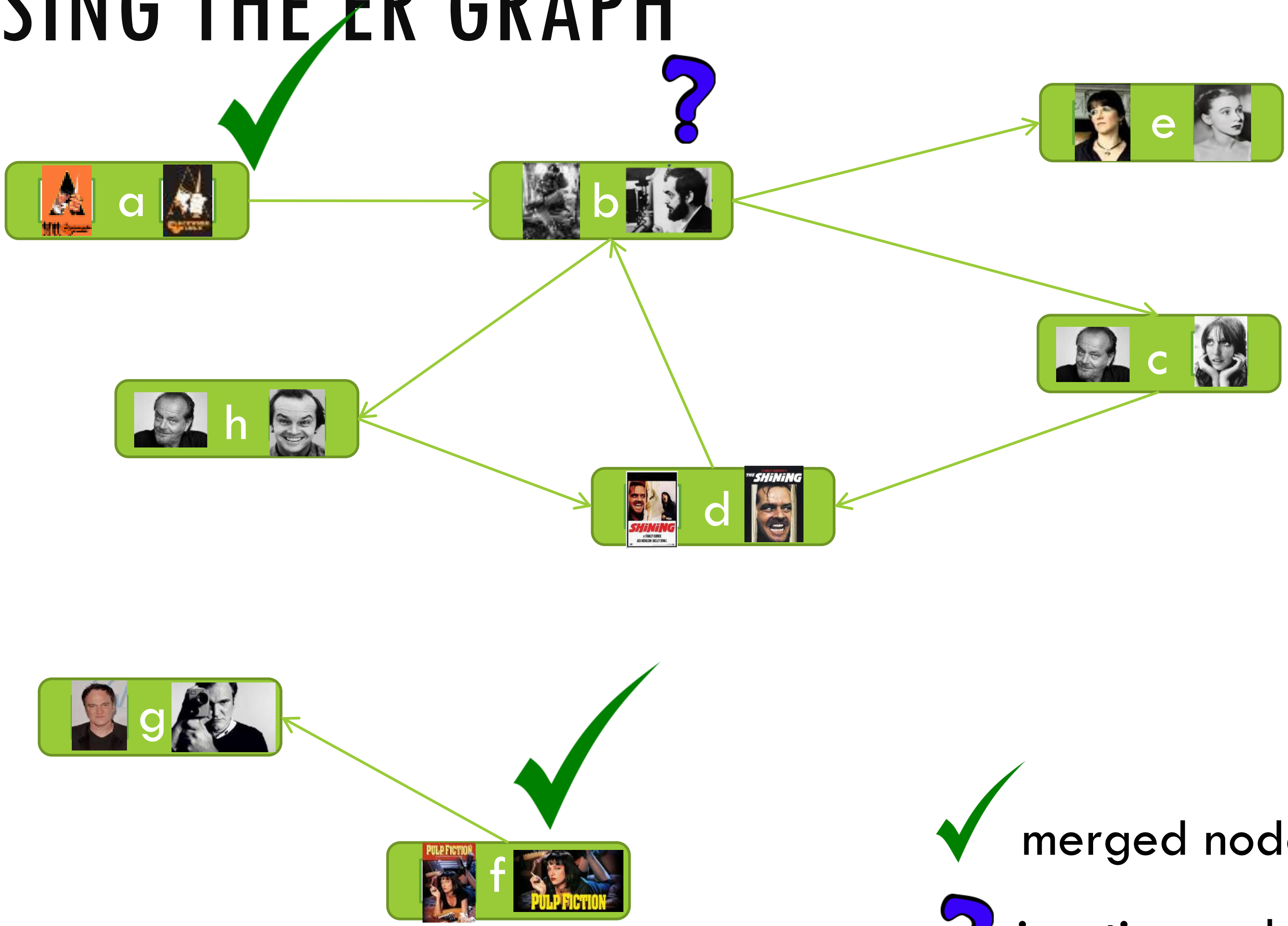


PQ
f
a
b
c
h
d
e
g

Initially all nodes are active and placed in the PQ

A node is processed before its out-neighbors

TRAVERSING THE ER GRAPH

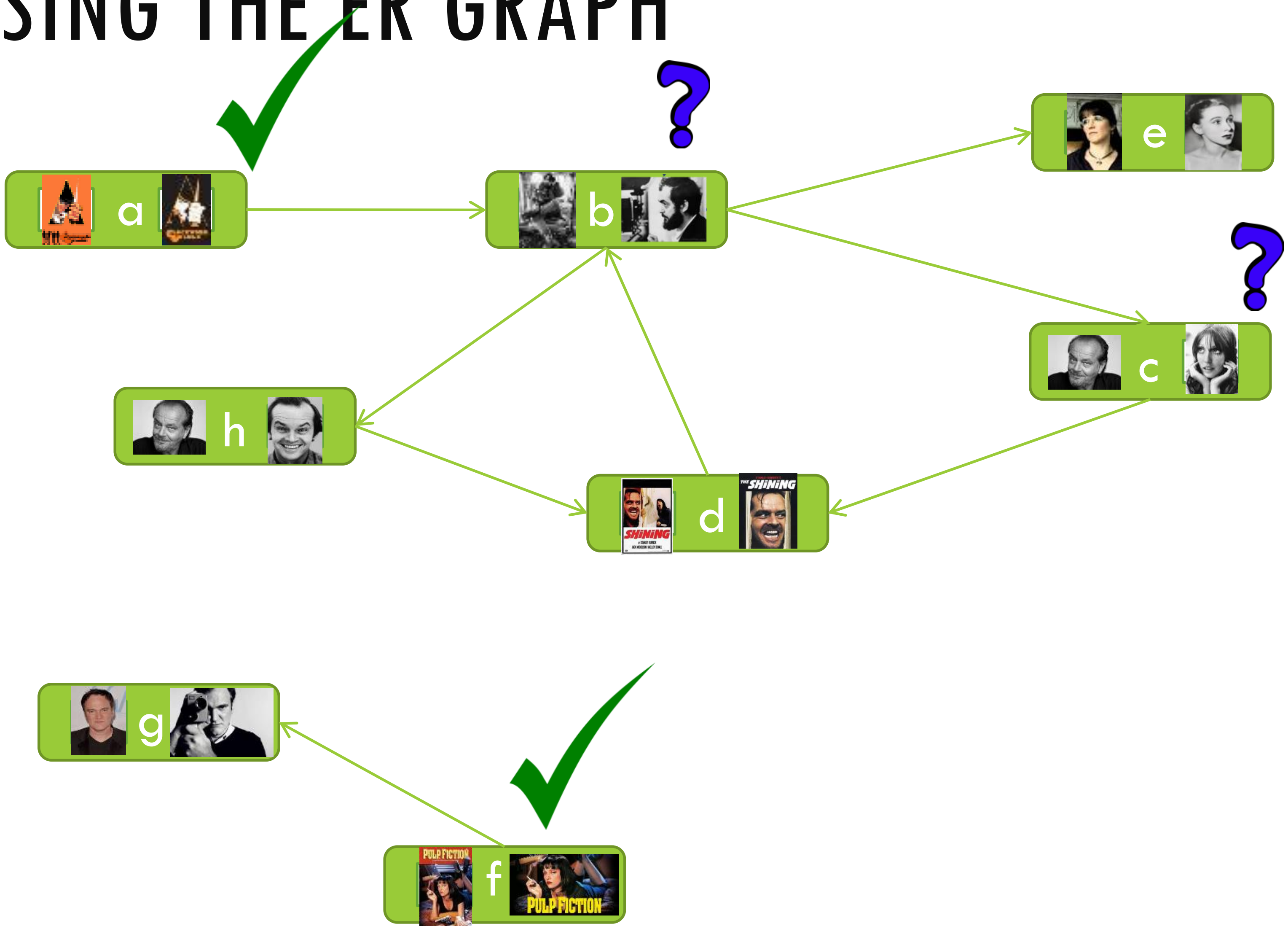


PQ
b
c
h
d
e
g

merged node

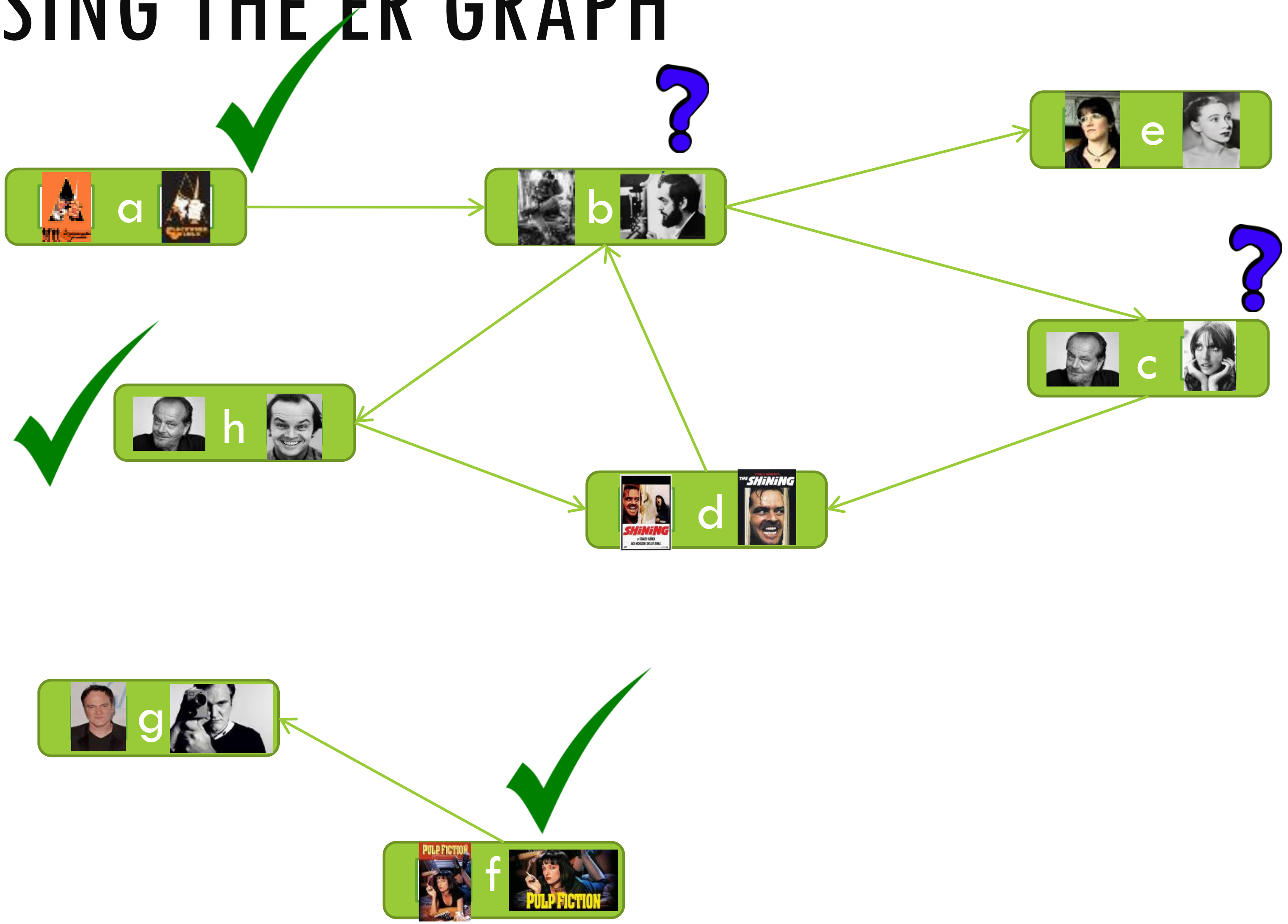
inactive node

TRAVERSING THE ER GRAPH



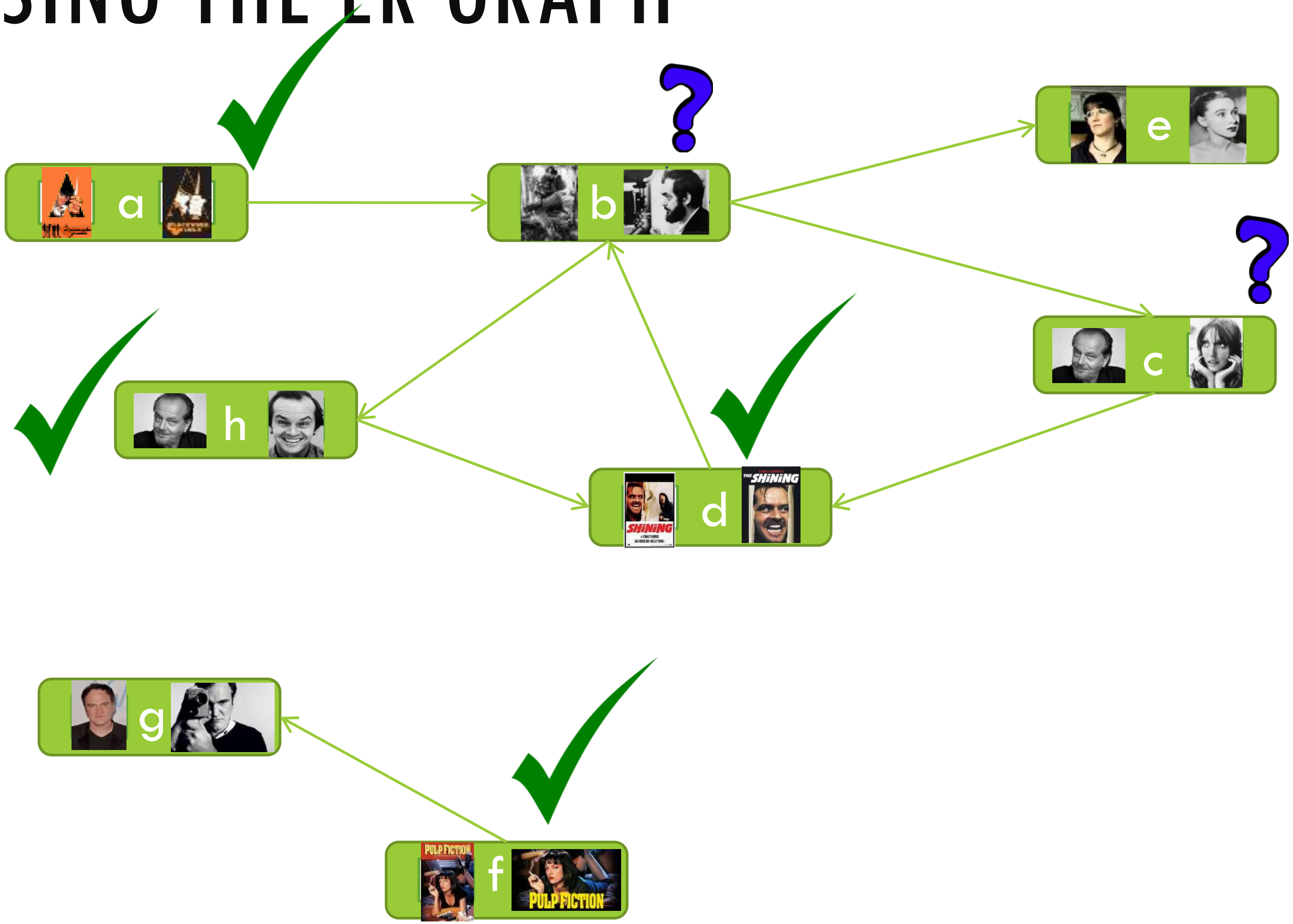
PQ
c
h
d
e
g

TRAVERSING THE ER GRAPH



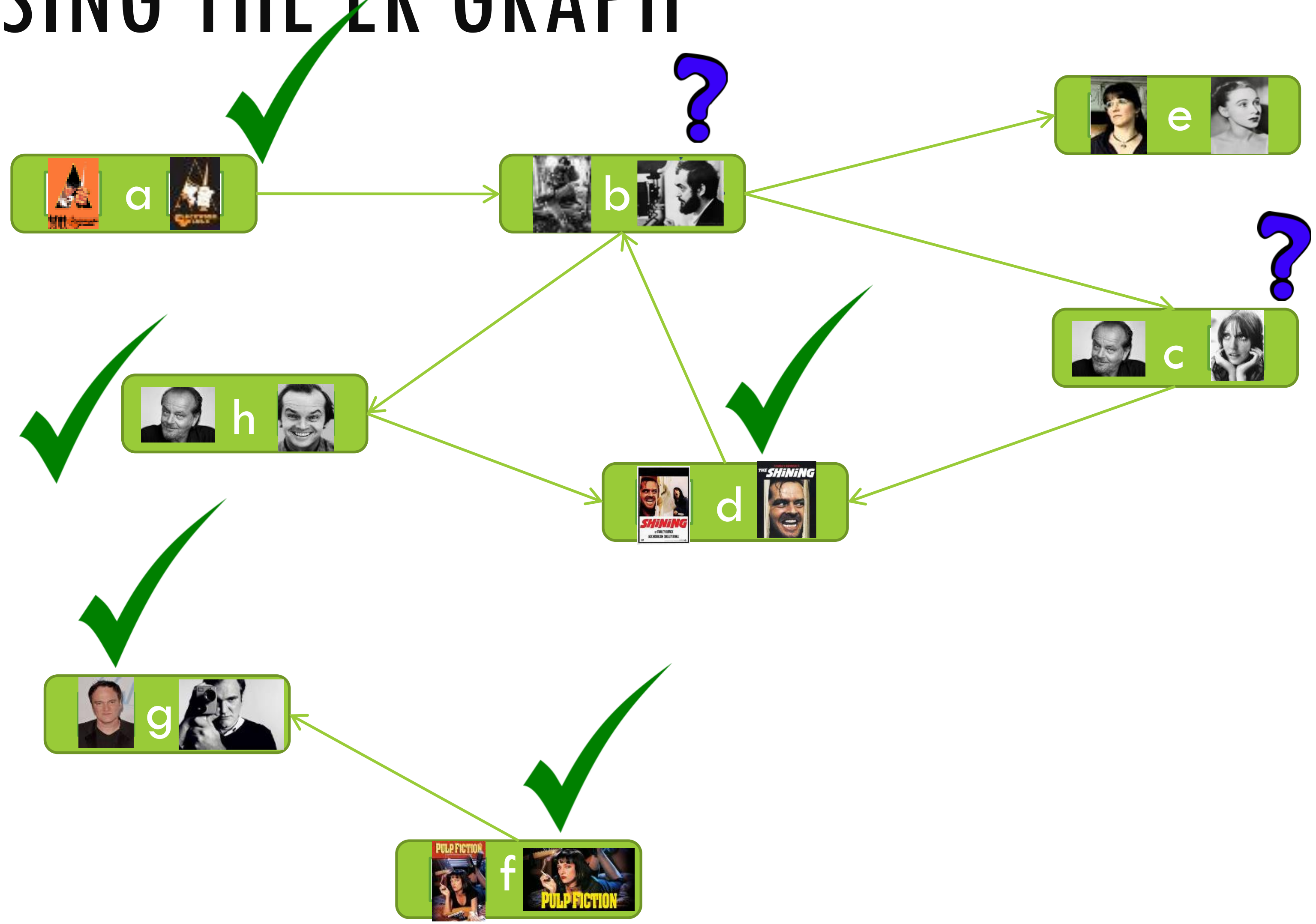
PQ
h
d
e
g

TRAVERSING THE ER GRAPH



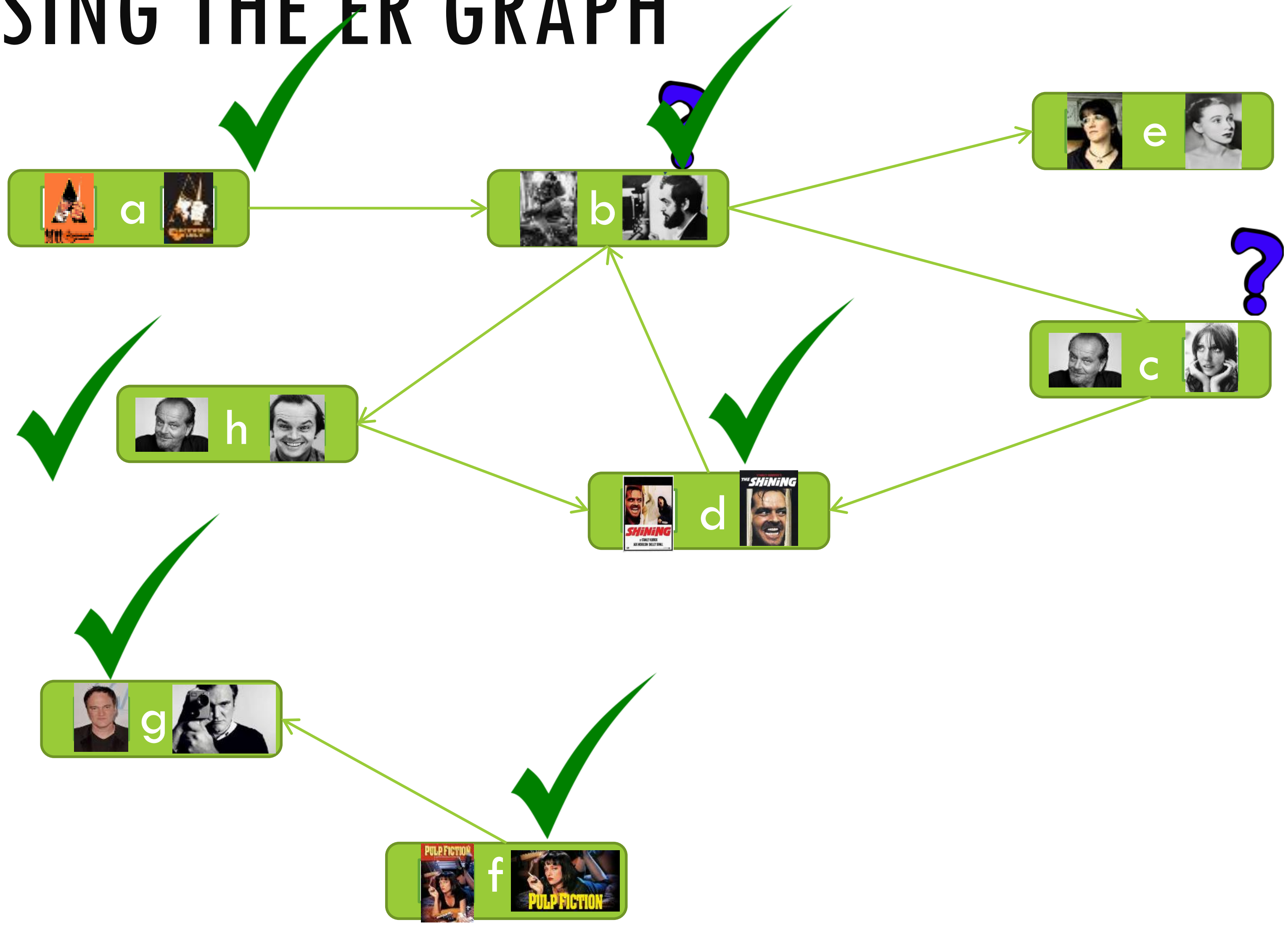
PQ
d
e
g

TRAVERSING THE ER GRAPH



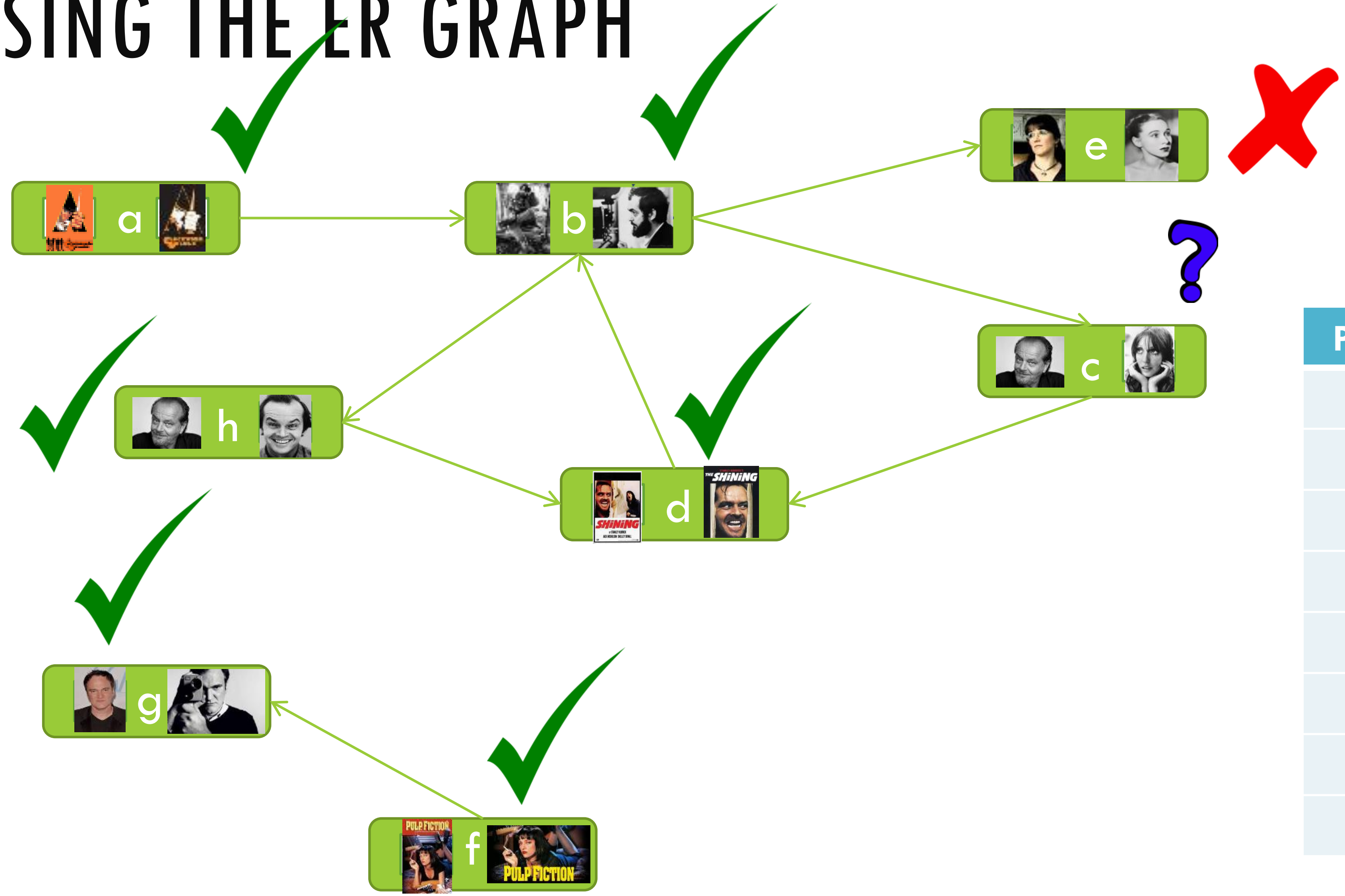
PQ
g
b
e

TRAVERSING THE ER GRAPH



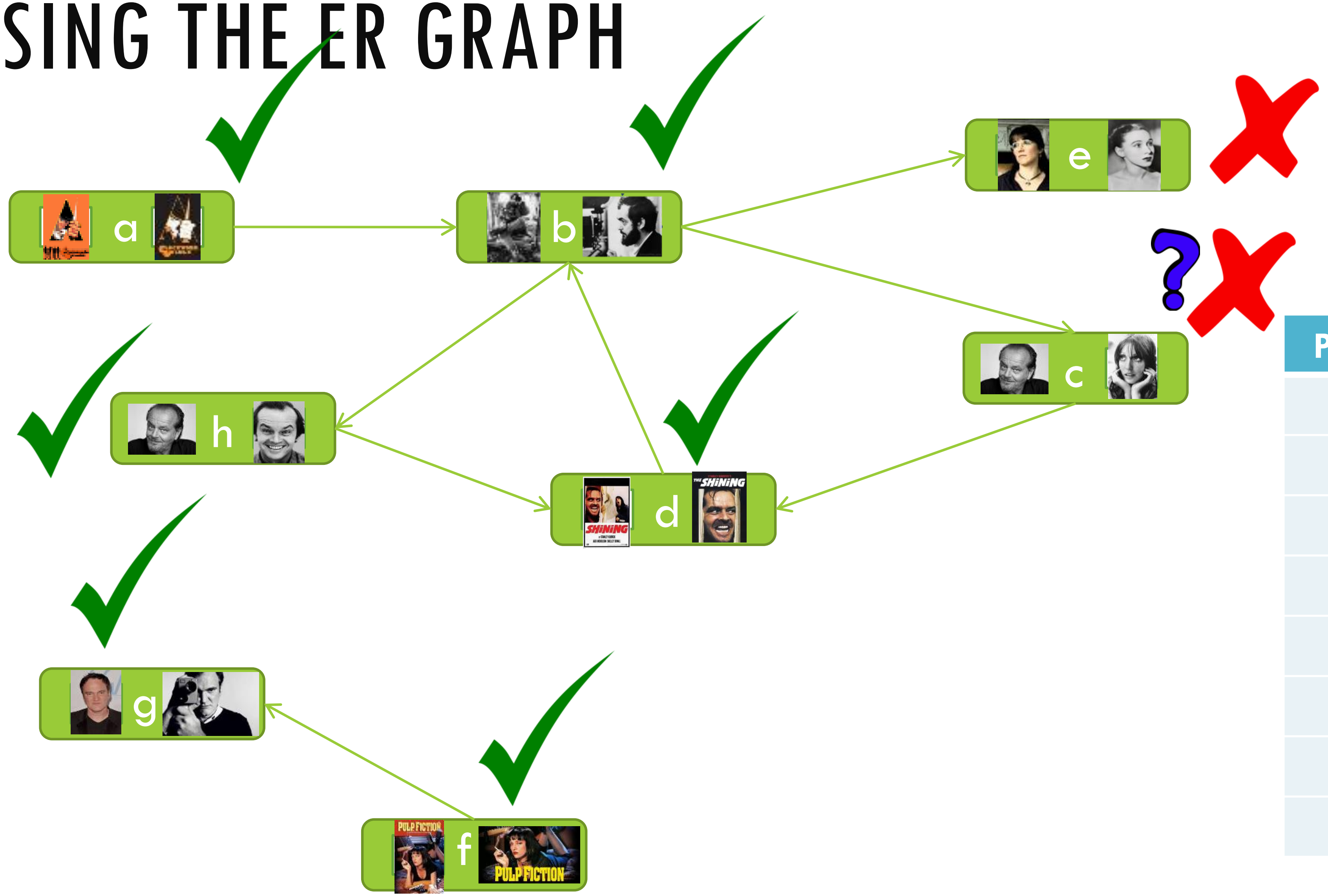
PQ
b
e

TRAVERSING THE ER GRAPH



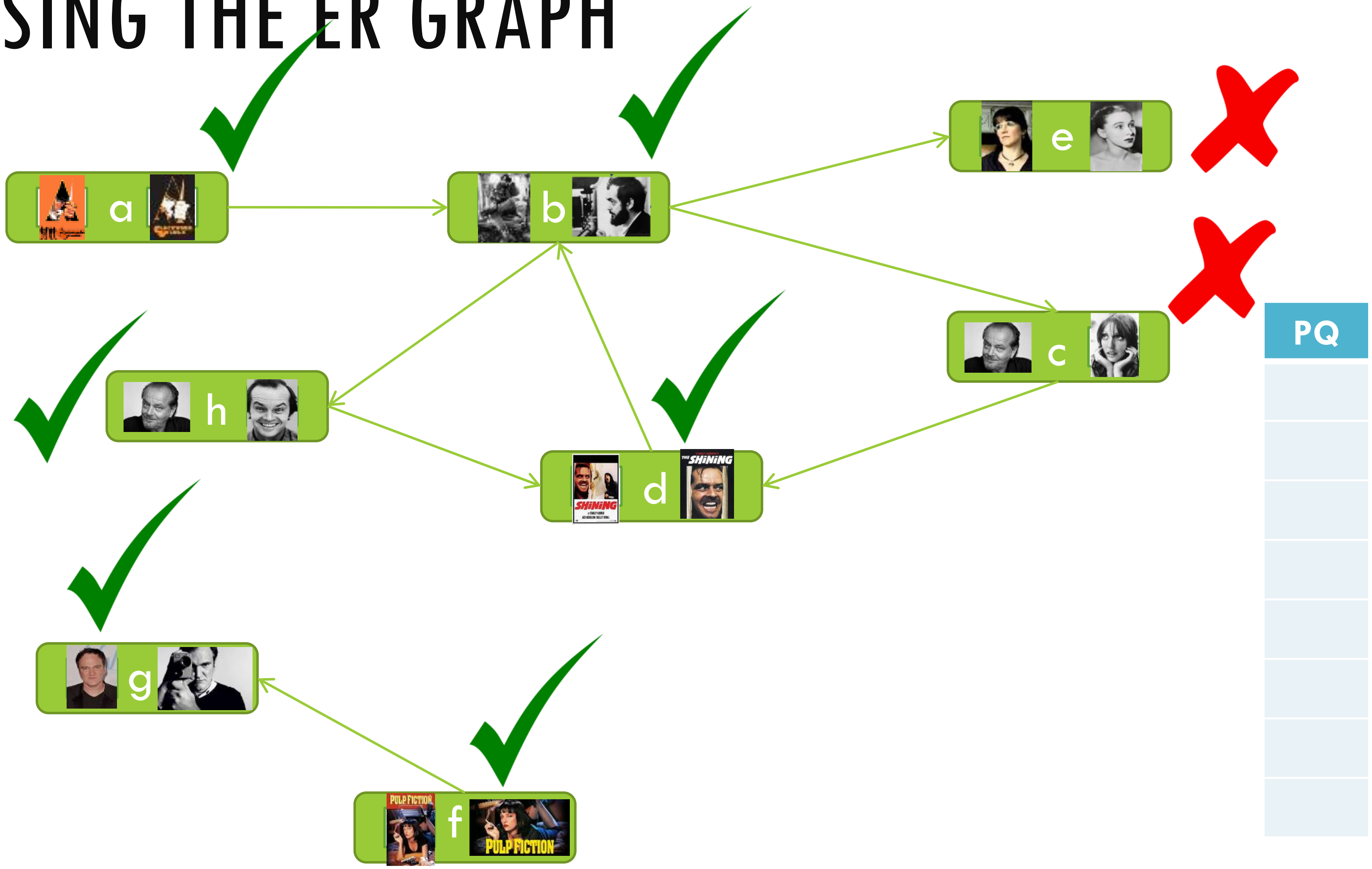
PQ
e
c

TRAVERSING THE ER GRAPH



PQ
c

TRAVERSING THE ER GRAPH



LINDA [BÖHM ET AL. 2012]

Key Idea: the more matching neighbours 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 neighbours: *checked iteratively*

Two square matrices ($|E| \times |E|$) are used:

- X captures the identified matches (binary values)
- Y captures the pair-wise similarities (real values) (is used only for the PQ)
  - Initialization: common neighbors & string similarity of literals
  - Updates: use the new identified matches of X

Until PQ becomes empty:

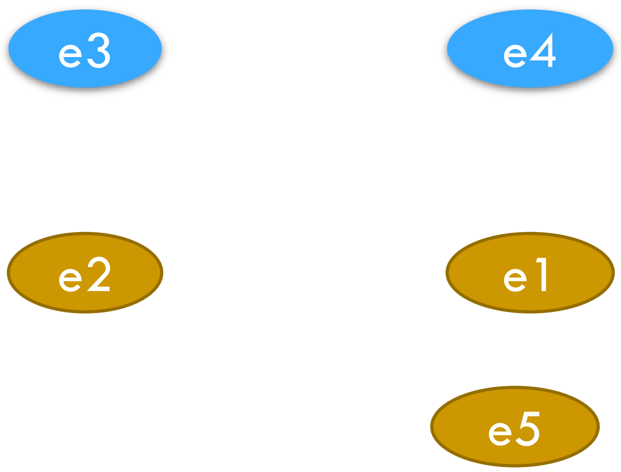
- Get the pair (e_i, e_j) with the highest similarity: match by default!
  - Update X: matches of e_i are also matches of e_j
- Update the similarity of nodes influenced by the new matches

LINDA 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

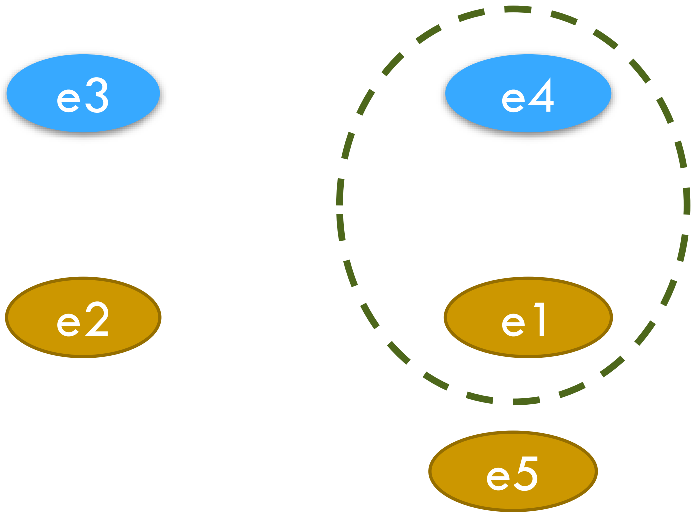


LINDA EXAMPLE

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



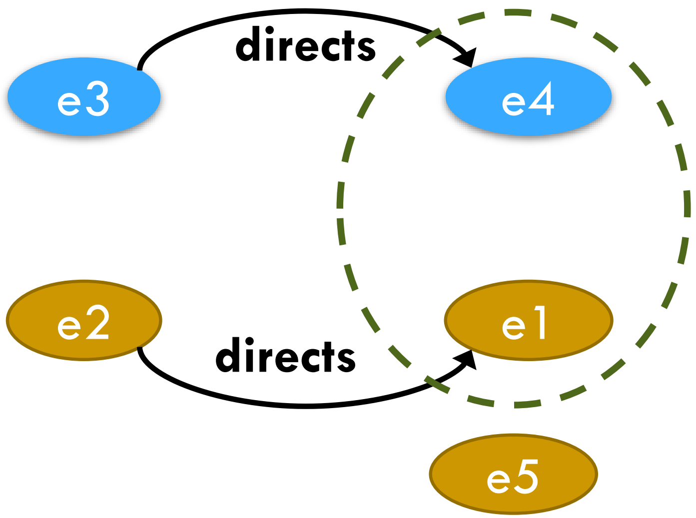
LINDA EXAMPLE

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)

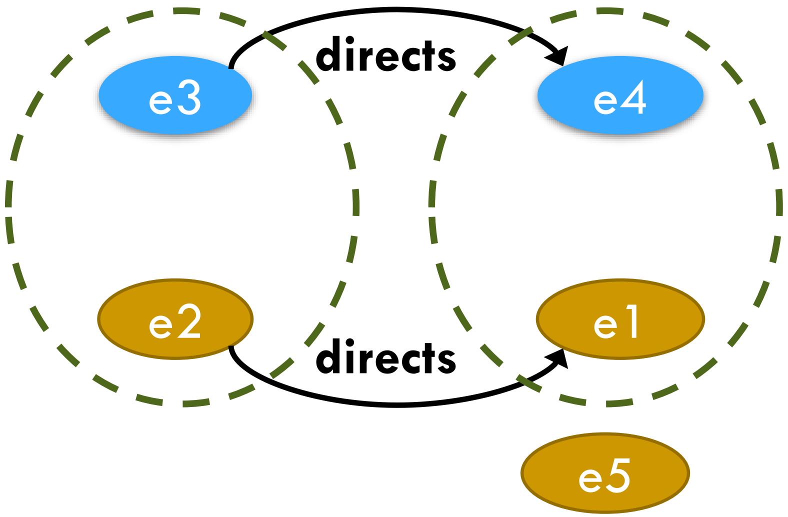
similarity re-computation, based on the matching neighbors and the names of the links to them



LINDA 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
…

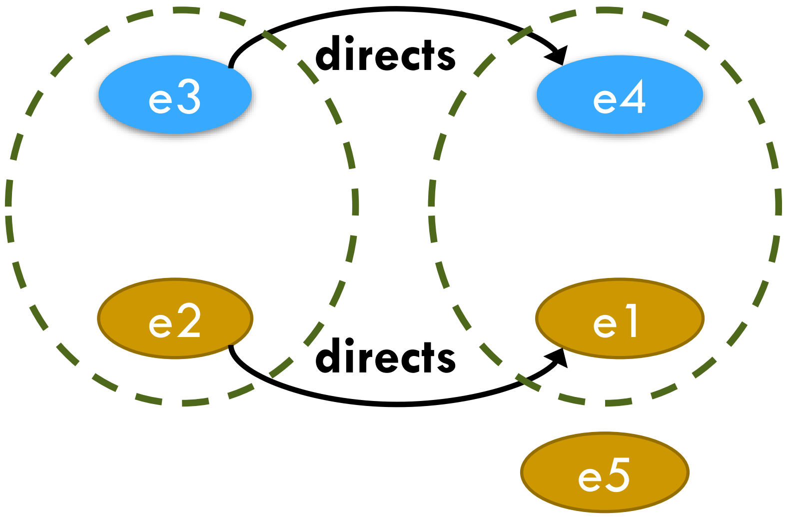


LINDA EXAMPLE

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

PROGRESSIVE RESOLUTION TECHNIQUES

PROGRESSIVE ER

Extend the typical ER workflow with a *planning phase*

- Select which pairs of descriptions, that have resulted from blocking, will be compared in the entity matching phase and in what order

The goal: Favour the more promising comparisons, i.e., those that are more likely to result in matches

- Those comparisons are executed before less promising ones and thus, more matches are identified early on in the process

[Optional phase] Update: Propagate the results of matching, such that a new scheduling phase will promote the comparison of pairs that were influenced by the previous matches

PROGRESSIVE ER

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

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

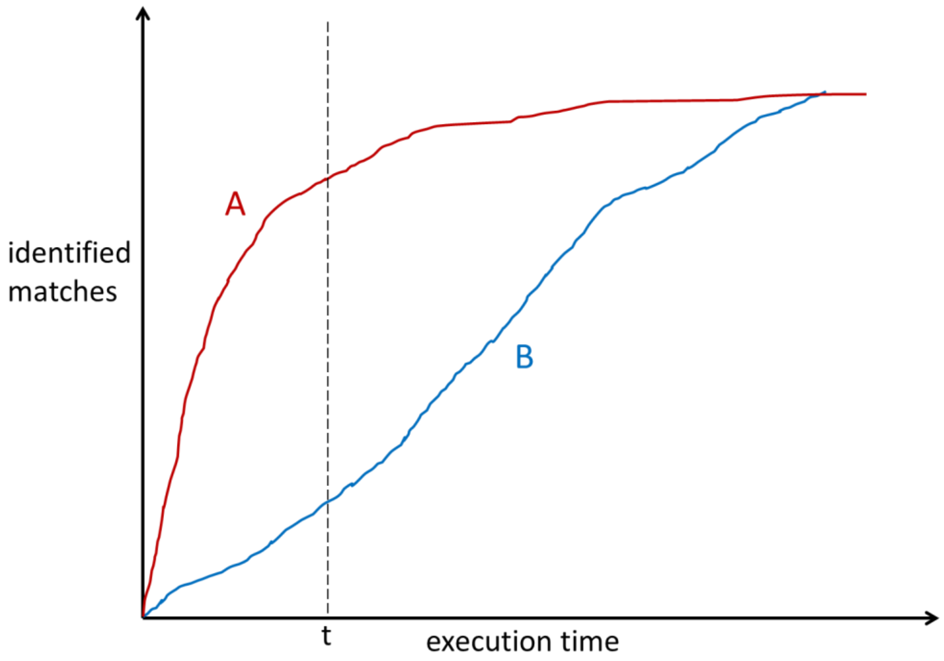
Good for high Velocity

This iterative process continues until the pre-defined computing budget is consumed

PROGRESSIVE RELATIONAL ER [ALTOWIM ET AL 2014]

Key Idea: Divide ER into several windows and generate a resolution plan for each window

- Specify which blocks and entity pairs within these blocks will be resolved during the plan execution phase of a window
- Associate with each identified pair the order in which to apply the similarity functions on the attributes of the two entities

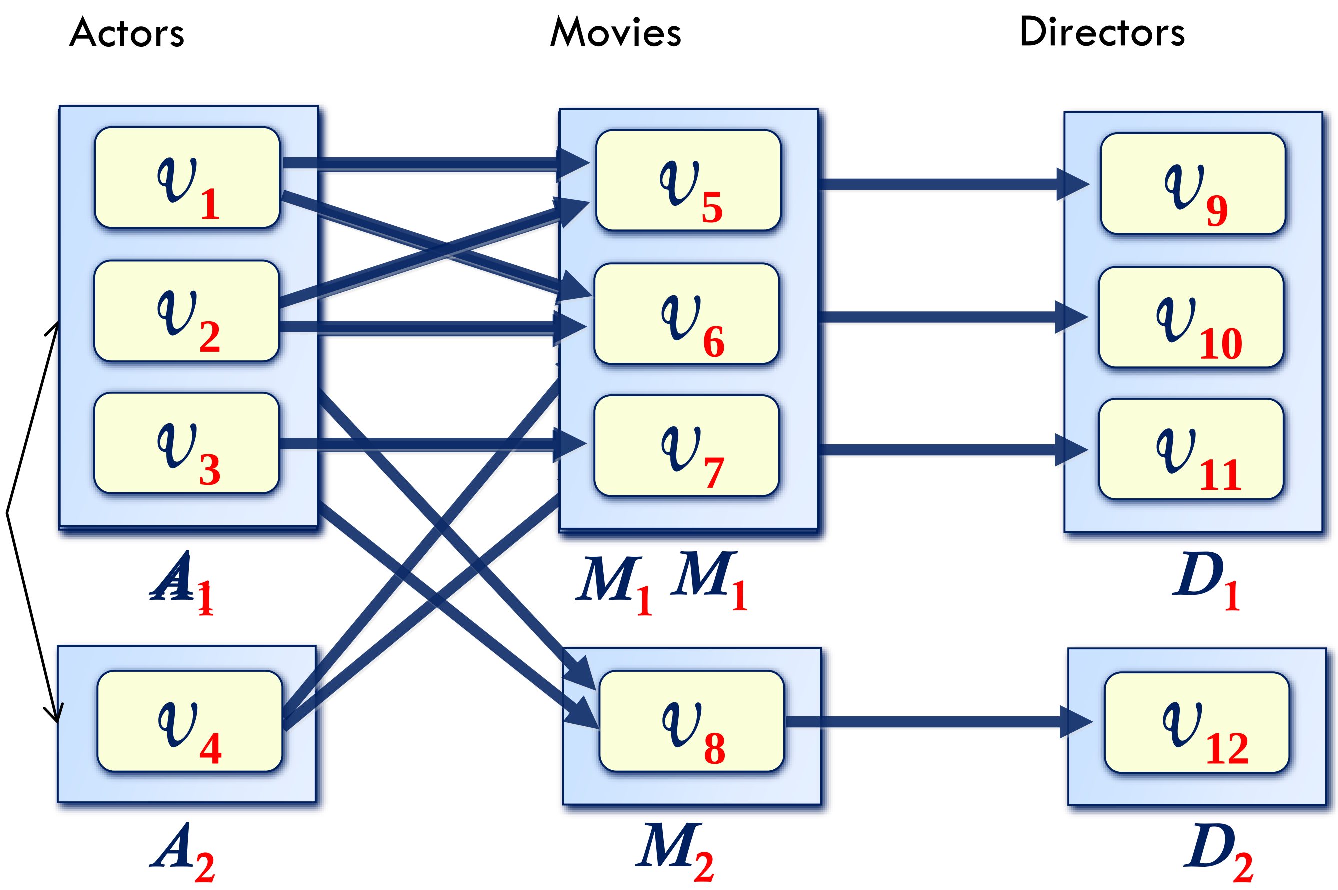


Lazy resolution strategy to resolve pairs with the smallest cost

- Unlike single entity type resolution a block based prioritization is significantly more important when resolving multiple types

PROGRESSIVE RELATIONAL ER [ALTOWIM ET AL 2014]

Blocks with entities of the same type



Nodes: Pairs of entity descriptions of the same type (relation)
Edges: Dependency between pairs (foreign keys) - an edge indicates that the resolution of a node influences the resolution of another node

PROGRESSIVE RELATIONAL ER [ALTOWIM ET AL 2014]

Black-box <u>blocking phase</u>

- Avoid building a dependency graph with all the description pairs

<u>Scheduling phase</u>: divide the total cost budget into several windows of equal cost

- For each window, a comparison schedule is generated
  - Choose among the schedules whose <u>cost</u> does not exceed the current window, the one with the highest expected *benefit*
    - The cost of a schedule is computed by considering the cost of finding the description pairs in a block according to the available storage policy (in memory/disk/cloud), and the cost of resolving every description pair

PROGRESSIVE RELATIONAL ER [ALTOWIM ET AL 2014]

Schedule benefit:

- How many matches are expected to be found by this schedule – *direct benefit*
- How useful it will be to declare those nodes as matches, in identifying more matches within the cost budget – *indirect benefit*

A node is more likely to be a match, when it is influenced by more matching nodes, and it is more influential, when it is expected to be a match and it has many direct dependent nodes

PROGRESSIVE RELATIONAL ER [ALTOWIM ET AL 2014]

Update phase

- After schedule execution: matching decisions are propagated to all influenced nodes, whose expected benefit now increases and have, thus, higher chances of being chosen by the next schedule

The algorithm terminates when the cost budget has been reached

- All unresolved pairs are considered non-matches – statistically, matches are significantly fewer than non-matches

OPEN ISSUES

OPEN ISSUES

Tight coupling of Blocking with Iterative Matching/Merging

- Better control of block characteristics w.r.t. the entity similarity subsequently used [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 [D. Firmani et al. 2016], Variety, and Volume entity descriptions [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 [Efthymiou et al. 2015, 2016]

OPEN ISSUES

Crowdsourced ER

- Reduce the crowdsourcing cost for obtaining ground truth [Chai et al. 2016, Gokhale et al. 2014, Wang et al. 2012]

Temporal ER

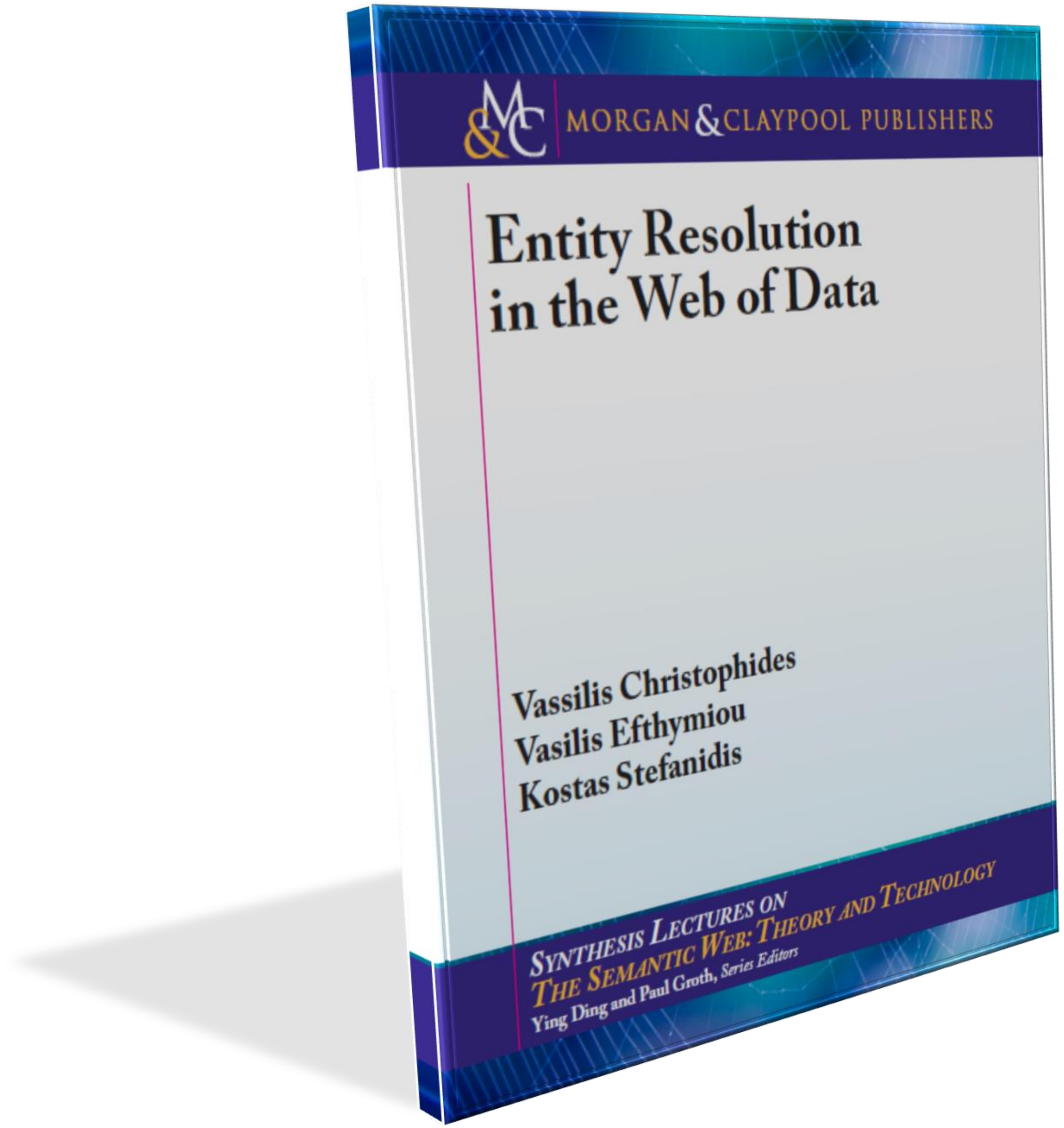
- Resolve evolving entity descriptions and analyse the history of descriptions [Dong & Tan 2015]

Uncertain ER

- Consider confidence scores when resolving certain & uncertain entity descriptions [Gal 2014, Demartini et al. 2013]

Privacy-aware ER

- Trade-off between entity obfuscation techniques and ER results quality [Whang & Garcia-Molina 2013]



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

LICENSE

THANK YOU!

Questions?

Kostas Stefanidis

- University of Tampere, Finland
- Kostas.Stefanidis@uta.fi

Vassilis Christophides

- INRIA-PARIS, France & University of Crete, Greece
- Vassilis.Christophides@inria.fr

Vasilis Efthymiou

- ICS-FORTH & University of Crete, Greece
- vefthym@ics.forth.gr