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
PUSH/PULL TECHNIQUES FOR RETRIEVING WEB CONTENT

Keyword search

Recommendations

Semantic search
CORE ENTITIES

Locations

Organizations

Persons

Movies
WHAT IS A KNOWLEDGE BASE (KB)?

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

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

Numbers from 2014
ENTITY-CENTRIC APPLICATIONS

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

- Primary objects: “things” (or descriptions of “things”)
- Links between “things” and not “strings”
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
KB publishers are encouraged to describe and interlink real world entities using the RDF data model.
THE LINK OPEN DATA (LOD) CLOUD

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

1,014 Knowledge Bases
60B Triples
649 vocabularies

2014-08-30 http://lod-cloud.net/
ENTITY INTERLINKING IN LOD

Only 56.11% of the KBs link to at least another KB

17.36% of them link to only one other KB (typically DBpedia)

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 entities 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)

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

Not identical descriptions, even if they have the same source
ENTITY RESOLUTION (ER)

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

A Clockwork Orange

Stanley Kubrick

Images are descriptions, not real-world entities
- Descriptions are partial, incomplete

highly similar descriptions

somehow similar descriptions

director
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
**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
- $M : E \times E \rightarrow \{\text{true, false}\}$ is a match function

The resolution of entities in $E$ results in a partition $P = \{p_1, ..., p_n\}$ of $E$, such that:

1. $\forall e_i, e_j \in E : M(e_i, e_j) = \text{true}, \exists p_k \in P : e_i, e_j \in p_k$
2. $\forall p_k \in P, \forall e_i, e_j \in p_k, M(e_i, e_j) = \text{true}$

- each partition contains only matching descriptions
- all the matching descriptions are in the same partition
**ER EXAMPLE**

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

**lmdb:director/8476**
- **lmdb:director_name** “Stanley Kubrick”
- **rdfs:type** foaf:Person
- **foaf:made** lmdb:film/1894
- **foaf:made** lmdb:film/2014
- **foaf:made** lmdb:film/2685

**lmdb:film/1894**
- **lmdb:film_name** “A Clockwork Orange”

**lmdb:film/2014**
- **lmdb:film_name** “A Clockwork Orange”

**lmdb:film/2685**
- **lmdb:film_name** “A Clockwork Orange”

**lmdb:film/2685**
- **lmdb:film_name** “A Clockwork Orange”

**lmdb:co0041067**
- **imbd:location** GeoNames:Buckinghamshire
- **imbd:filmography** “A Clockwork Orange”

**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** lmdb:director/8476
- **fbase:film.runtime** “136”
- **fbase:film_cut/runtime** “136”
- **fbase:film_cut/film** “A Clockwork Orange”
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

- dbpedia:Stanley_Kubrick
  - director
  - children
    - dbpedia:Katharina_Kubrick
    - dbpedia:The_Shining
  - spouse
    - freebase:Ruth_Sobotka
  - directs
    - linkedMDB:A_Clockwork_Orange
    - freebase:The_Shining
Matching decisions are independent

\[ \text{thresh} = 0.5 \]

\[ \text{sim}_c(e1, e3) = \text{Jaccard} \{ \text{Manhattan}, \text{Person}, \text{AmericanFilmDirectors}, \text{AmateurChessPlayers} \}, \{ \text{Stanley, Kubrick, Person, 1894, 2014, 2685} \} = 0.1 \]

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

**e3**
- director_name: “Stanley Kubrick”
- type: Person
- directs: film/1894
- directs: film/2014
- directs: film/2685

\( \text{sim}_c \): 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

- **e1**: dbpedia:Stanley_Kubrick
- **e2**: dbpedia:A_Clock_work_Orange
- **e3**: freebase:Stanley_Kubrick
- **e4**: linkedMDB:A_Clock_work_Orange
- **e5**: dbpedia:The_Shining
- **e6**: freebase:The_Shining
- **e7**: dbpedia:Katharina_Kubrick
- **e8**: freebase:Ruth_Sobotka

The relationships shown in the diagram include:
- Stanley Kubrick directs A Clockwork Orange and The Shining.
- Katharina Kubrick is a child of Stanley Kubrick.
- Ruth Sobotka is the spouse of Stanley Kubrick.
MATCHING NEIGHBORHOOD

\[
\text{sim}_c(e_2, e_4) = \text{Jaccard} \left( \{e_1, 136, A, \text{Clockwork}, \text{Orange}\}, \{e_3, 136, A, \text{Clockwork}, \text{Orange}\} \right) = 0.66
\]
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

<table>
<thead>
<tr>
<th>High similarity in structure</th>
<th>Low similarity in structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>High similarity in content</td>
<td></td>
</tr>
<tr>
<td>Low similarity in content</td>
<td>set sim. in the values of specific atts from two relations</td>
</tr>
</tbody>
</table>

- **string**: sim. in the values of specific atts from one relation
- **att & value**: sim. in a network of relations

The definition of what is similar is domain-dependent
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}
- $M(e_i, e_j) = true \Rightarrow e_i, e_j$ are matches
- $M(e_i, e_j) = false \Rightarrow e_i, e_j$ are non-matches

*Imbalanced: typically, $O(E)$ matches, $O(E^2)$ non-matches*
MATCH FUNCTION: FORMAL PROPERTIES

The match function $\mathcal{M}()$ introduces an equivalence relation (owl:sameAs) among entity descriptions:

- **Reflexivity**: $\forall e_i \in E, \mathcal{M}(e_i, e_i) = true$
- **Symmetry**: $\forall e_i, e_j \in E, \mathcal{M}(e_i, e_j) = \mathcal{M}(e_j, e_i)$
- **Transitivity**: $\forall e_i, e_j, e_k \in E$, if $\mathcal{M}(e_i, e_j) = true$ and $\mathcal{M}(e_j, e_k) = true$, then $\mathcal{M}(e_i, e_k) = true$
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 $\theta$:

- $M(e_i, e_j) = \text{true, if } sim(e_i, e_j) \geq \theta$
- $M(e_i, e_j) = \text{false, otherwise}$

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 e₂ and e₄ are highly similar

e₁ and e₃ are at best somehow similar
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
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
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**

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

- Jaccard\( (e_1,e_3) = \frac{1}{8} \)
- Jaccard\( (e_1,e_4) = 1 \)
- Jaccard\( (e_1,e_5) = \frac{1}{8} \)
- Jaccard\( (e_2,e_3) = \frac{3}{7} \)
- Jaccard\( (e_2,e_4) = \frac{1}{11} \)
- Jaccard\( (e_2,e_5) = 0 \)
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
The contextual similarity of nodes $n$ and $m$ is:

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

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

where $x_{n,m}$ is 1, if $n$, $m$ are identified as matches, and 0 else and \text{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.
The similarity score for descriptions $e$ and $e'$ is:

$$\text{sim}^{\text{LINDA}}(e,e') = \text{content_sim}(e,e') + \beta \times \text{context_sim}(C(e),C(e')) - \theta$$

where $\beta$ controls the contextual influence, $\theta$ is used for re-normalization to values around 0,

$$\text{content_sim}(e,e') = \frac{|N_n \cap N_m|}{\min(|N_n|,|N_m|) + \ln(|N_n| - |N_m| + 1)}$$

$\text{sim}^{\text{LINDA}}$ 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
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 $j$

- $N_{ij} = \{(k, l): (i, r, k) \in KB1$ and $(j, s, l) \in KB2$ and relationship $r$ is matched to $s\}$

Properties matching is provided by the users

String similarity: weighted Jaccard (IDF-like)
Content similarity: static score of both the string representation of entities (rdfs:label) and their other property values

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

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)

\[ \delta g_{ij}(y) = \sum_{(k,l) \in N_{ij}} y_{kl} (w_{ij,kl} + w_{kl,ij}) \]

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

\[ g_{ij}(y) = \sum_{(k,l) \in N_{ij}} y_{kl} (\gamma_i w_{ik} + \gamma_j w_{jl}) \]
SIGMA SIMILARITY MEASURES

Global score:

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

- **Blocking**
  - O(n)
    - if hash-based
  - Group similar enough entity descriptions

- **Matching**
  - O(b^2d^2)
    - b blocks of d descriptions (avg)

**Preliminary experiment over 9M entity descriptions in a cluster of 15 VMs:**
- ER workflow without blocking: >200 hrs
- ER workflow with blocking: 11 hrs
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)
Actually, an inverted index
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
For each attribute of $\text{KB}_1$:
- Find the most similar attribute of $\text{KB}_2$

For each attribute of dataset $\text{KB}_2$:
- Find the most similar attribute of dataset $\text{KB}_1$

Compute the transitive closure of the generated pairs of attributes

Connected attributes form clusters

All single-member clusters are merged into a common cluster
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 | year-constructed | 1889 | e15 |
| artist | Bartholdi | location | Paris | e16 |
| work | Eiffel Tower | year-constructed | 1889 | e17 |
| work | Bartholdi Fountain | year-constructed | 1876 | e18 |
| location | Washington D.C. | location | | e19 |
- Compute the **transitive closure** of the generated attribute pairs
  - Connected attributes form **clusters**

- Example: Pairs (about, work), (work, about), (artist, architect), (architect, work)
<table>
<thead>
<tr>
<th>about</th>
<th>Eiffel Tower</th>
<th>about</th>
<th>Statue of Liberty</th>
<th>about</th>
<th>Auguste Bartholdi</th>
<th>about</th>
<th>Joan Tower</th>
</tr>
</thead>
<tbody>
<tr>
<td>architect</td>
<td>Sauvestre</td>
<td>architect</td>
<td>Bartholdi</td>
<td>born</td>
<td>1834</td>
<td>born</td>
<td>1938</td>
</tr>
<tr>
<td>year</td>
<td>1889</td>
<td>year</td>
<td>1886</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>located</td>
<td>Paris</td>
<td>located</td>
<td>NY</td>
<td>work</td>
<td>Eiffel Tower</td>
<td>work</td>
<td>Bartholdi Fountain</td>
</tr>
<tr>
<td>work</td>
<td>Lady Liberty</td>
<td>year-constructed</td>
<td>1889</td>
<td></td>
<td></td>
<td>year-constructed</td>
<td>1876</td>
</tr>
<tr>
<td>artist</td>
<td>Bartholdi</td>
<td>location</td>
<td>Paris</td>
<td>location</td>
<td></td>
<td>location</td>
<td>Washington D.C.</td>
</tr>
<tr>
<td>location</td>
<td>NY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

→ compare Lady Liberty to Auguste Bartholdi
OTHER BLOCKING TECHNIQUES

Infix blocking: The blocking key is the URI infix of the entity description
o Example: http://en.wikipedia.org/wiki/Linked_data#Principles.html
  o Infix is a local identifier
  o 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
o May significantly reduce the number of candidate pairs
o 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Blocking [Papadakis et al., 2011]</td>
<td>The descriptions have a common token in their values</td>
</tr>
<tr>
<td>Attribute Clustering Blocking [Papadakis et al., 2013]</td>
<td>The descriptions have a common token in the values of attributes that have similar values in overall</td>
</tr>
<tr>
<td>Prefix-Infix(-Suffix) [Papadakis et al., 2012]</td>
<td>The descriptions have a common token in their literal values, or a common URI infix</td>
</tr>
<tr>
<td>Frequent itemsets [Kenig and Gal, 2013]</td>
<td>The descriptions have frequently co-occurring tokens in their values</td>
</tr>
</tbody>
</table>
BLOCK POST-PROCESSING
META-BLOCKING: IMPROVE THE EFFICIENCY OF BLOCKING

**Goal:**
- 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):

<table>
<thead>
<tr>
<th>Entity</th>
<th>Eiffel</th>
<th>Tower</th>
<th>Liberty</th>
<th>NY</th>
<th>1889</th>
<th>Paris</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(e_1, e_4, e_2, e_3)</td>
<td>(e_1, e_4, e_5)</td>
<td>(e_2, e_3)</td>
<td>(e_2, e_3)</td>
<td>(e_1, e_4)</td>
<td>(e_1, e_4)</td>
</tr>
</tbody>
</table>

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

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

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

2 comparisons to identify 2 matches

edge weights = \#common blocks

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): \( w_{i,j} = CBS \cdot \log(|B|/|Bi|) \cdot \log(|B|/|Bj|) \)

**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
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 very specific types.
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)
  
- Type B: matches of their neighbours

Applicable to identify matches within a KB

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:

- \( M^n(e_i,e_j) = \text{true}, \text{ if } \text{sim}^{n-1}(e_i,e_j) \geq 0 \)
- \( M^n(e_i,e_j) = \text{false}, \text{ if } \text{sim}^{n-1}(e_i,e_j) \leq 0 \)
- \( M^n(e_i,e_j) = \text{undecided}, \text{ otherwise} \)

Total similarity:

\[ \text{sim}(e_i,e_j) = a \times \text{sim}_{\text{nbr}}(e_i,e_j) + (1-a) \times \text{sim}_{\text{nbr}}(\text{nbr}(e_i),\text{nbr}(e_j)) \]

where \( \text{nbr}(e) \) 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 $\theta$
- 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**
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 $e_{bi}, e_{bj} \in \text{values}(e_{ai}) \cup \text{values}(e_{aj})$

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
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
TRACING THE ER GRAPH

Initially all nodes are active and placed in the PQ.
A node is processed before its out-neighbors.
TRACING THE ER GRAPH

merged node
inactive node
TRAVERSING THE ER GRAPH
TRaversing the ER Graph
TRAVERSING THE ER GRAPH
TRACING THE ER GRAPH

a → b
b → c
b → d
d → e
d → f
g → h
h → b
h → g

PQ
- g
- b
- e
TRAVERSING THE ER GRAPH
TRAVERSING THE ER GRAPH
TRAVERSING THE ER GRAPH
TRAVERSING THE ER GRAPH
**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
A priority queue, derived by an initial similarity computation between all pairs, based on their attribute values.

<table>
<thead>
<tr>
<th>Matches</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>e5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
## LINDA EXAMPLE

<table>
<thead>
<tr>
<th>Matches</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>e2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>e3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4</td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The head of PQ is a match by default.
LINDA EXAMPLE

<table>
<thead>
<tr>
<th>Matches</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>e2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**PQ**

- unique mapping constraint (1-1 Assumption)
  - $e2 \rightarrow e4$
  - $e1 \rightarrow e3$
  - $e2 \rightarrow e3$
  - $e5 \rightarrow e3$
  - ...

**Similarity re-computation**, based on the matching neighbors and the names of the links to them.
### LINDA EXAMPLE

**Matches**

<table>
<thead>
<tr>
<th></th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>e2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>e3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e5</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PQ**

- e2 – e3
- e5 – e3
- ...

![Diagram showing directed relationships between elements e1, e2, e3, e4, and e5](diagram.png)
**LINDA EXAMPLE**

<table>
<thead>
<tr>
<th>Matches</th>
<th>e1</th>
<th>e2</th>
<th>e3</th>
<th>e4</th>
<th>e5</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>e2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>e3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**PQ**

<table>
<thead>
<tr>
<th>e5—e3</th>
</tr>
</thead>
</table>

**unique mapping constraint (1-1 Assumption)**

---

Diagram:

- **e3** directs **e4**
- **e2** directs **e1**
- **e5** 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]

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.
Black-box blocking phase

- Avoid building a dependency graph with all the description pairs

Scheduling phase: divide the total cost budget into several windows of equal cost

- For each window, a comparison schedule is generated
  - Choose among the schedules whose cost 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
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.
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

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


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**

**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]
Entity Resolution in the Web of Data

Vassilis Christophides
Vasilis Efthymiou
Kostas Stefanidis

Synthesis Lectures on The Semantic Web: Theory and Technology
Ying Ding and Paul Gruber, Series Editors
LICENSE

These slides are made available under a Creative Commons Attribution-ShareAlike license (CC BY-SA 3.0): http://creativecommons.org/licenses/by-sa/3.0/

You can share and remix this work, provided that you keep the attribution to the original authors intact, and that, if you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
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