KNOWDIVE

KGE - Knowledge Graph Engineering

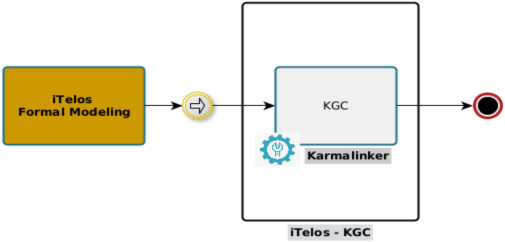
KGC - Data Integration phase

Building an Entity Graph

Fausto Giunchiglia

Contents

KGC - Data Integration phase



- **Input**: the final KG's ETG, plus the set of formal data resources.
- **Output**: the final KG.
- **Objective**: to merge the knowledge layer with data layer into a single exploitable resource.

KGC - Data Integration phase

- In the last phase of the methodology, the previous intermediate results are **composed** in order to build the final KG.
  - **Mapping** of the datasets cleaned, formatted and aligned, on the ETG (Teleontology).
- In other words we adopt the **unique information's representation** defined by the ETG, **for the different datasets** obtained until this phase.

This problem is called **Data Integration** (DI), which is addressed by iTelos using **Knowledge Graph Construction** (KGC) techniques.

KGC phase steps

- The last phase of the methodology is divided in three main steps:

1. **The identity problem**: to find and define the identity of the entities within the datasets considered.
2. **The entity matching**: to map different representation of the same entity, from different datasets, to the relative single etypes.
3. **The KG's evaluation**: to evaluate the quality of the final KG.

Contents

The problem of identity
Semantic Heterogeneity

- As already discussed, a crucial part of a KGE process is to address the **semantic heterogeneity** within the data to be exploited.
- Such kind of heterogeneity indicates the presence of multiple representation of the same real world entity, within the resources collected.
- The semantic heterogeneity is a:

"consequence of the more general phenomenon of the diversity of the world and of the world descriptions." (Giunchiglia, Fumagalli 2020)

The problem of identity

- With the objective to unify the representation of the information, the semantic heterogeneity brings the need to **identify the different entities.**
- More in details, we need to:
  - identify an entity within a **single dataset**;
  - adopt the same identification, if the same entity is represented in two (or more) different ways, **within different datasets**.
- To this end we need a way to identify entities (entity identification).

The problem of identity
Entity Identification

- An entity (like the etypes) is identified by its properties.
- Sometimes (datasets well formed) within datasets it is already present a specific property aiming at identifying the entity it belongs to.
  - This property is called **Identifier**.
- There **multiple kinds of identifiers**, depending on how the entities need to be identified.

The problem of identity - Identifiers

- **URI**: A Uniform Resource Identifier (URI) is a unique sequence of characters that identifies a logical or physical resource used by web technologies.
- A URI can be defined as:
  - **URL**: A Uniform Resource Locator (URL) is a URI that specifies the means of acting upon or obtaining the representation of a resource, i.e. specifying both its primary access mechanism and network location.
  - **URN**: A Uniform Resource Name (URN) is a URI that identifies a resource by name in a particular namespace.
  - Examples and more details can be found directly at Wikipedia URI
- Nevertheless, identifiers are not always provided in the datasets.

The problem of identity - Identifying Sets

- When an identifier (a single entity's property) is not available, an entity can be identified uniquely by the union of the values from two (or more) of its properties.
  - Such a property composition is called **Identifying Set**.

Identifying Set: a set of etype's properties which, through their values, identify uniquely an entity (defined for such an etype) within the whole set of entity considered.

Identifying Sets - Example

Bus in dataset A:

- Production-year: 2007
- Manufacturer: "Iveco"
- Model: "AX-123"
- Engine-type: "Electric engine"
- Fuel-type: "Electricity"

Bus in dataset B:

- Production-year: 2007
- Line-number: "13-A"
- Seats: 30
- Daily-travel-time: 650
- Model: "AX-123"

The Identifying Set (IS) is defined as follow:

$$IS_{Bus} = \textit{Production-year, Model}$$

allows the matching between the two *Bus* entities into a single one.

Contents

The entity matching

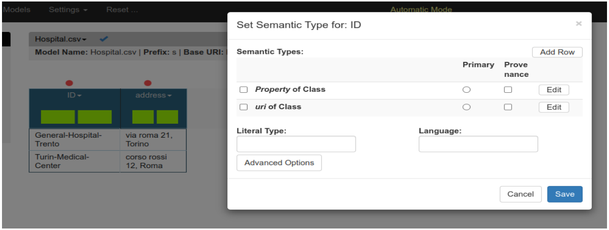
- Even if we found a way to identify entities, they can be represented through different properties, and properties values, within different datasets.
- This is known as **the entity matching problem**, and it has two main consequences:
    1. (Schema layer) The need to find **the right set of properties** (Identifying Set), between those specified by multiple representations of the same entity.
    2. (Data layer) The need to set **the correct property values**, if multiple representations share the same properties with different values .

The entity matching - solve conflicts

- **How to solve entity matching conflicts ?**
- Both for schema and data values conflicts, a possible solution is provided by **Metadata**.
- In particular, thus metadata carrying information about the provenance and the reliability of the entities having conflicts.
  - **Author** and **Organization** metadata allow us to understand who created the data, thus giving us a criteria in order to decide which property/value should be considered, or not.
  - **Creation Date** and **Modification Date**, similarly give us information about how much up-to-date the data are (too old or too new, depending by what our purpose requires).
  - Also for entity matching, **the purpose** is the main criteria to be used in order to solve conflicts.

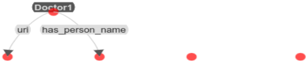
The entity matching - solve conflicts

- In practice, the **Karma** data mapping tool is used to solve entity matching conflicts.
- It allows the creation and the modification of datasets attributes in order to build up identifiers or identifying sets.
- It provides a dedicated *Semantic type mapping* called **"uri of Class"** used to define a URI identifier for all the entities of a specific etype.



The entity matching - solve conflicts

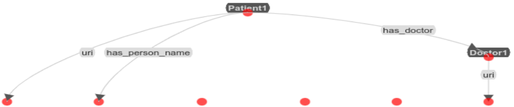
- Karma, using the URI property will automatically merge conflicts.



ID	Name	Surname	Hospital
Doctor-1	Mario	Draghi	General-Hospital-Trento
Doctor-2	Clara	Bella	Turin-Medical-Center

Patient .csv ✔

Model Name: Patient .csv | **Prefix:** s | **Base URI:** http://localhost:8080/source/ | **Github URL:** disabled



ID	Name	date-of-birth	address	Gender	Doctor
2	Bruce Banner	19901203	via roma 2, Trento	Male	Doctor-1
1	Tony Stark	19860511	via verdi 12, Roma	Male	Doctor-2
4	Anna Verdi	19750903	via castello 2, Tornio	Female	Doctor-1

Contents

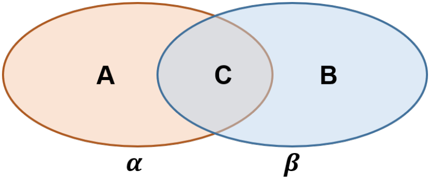
The KG's evaluation

- **How to evaluate the quality of the final KG ?**
- iTelos provides different criteria to evaluate the explicit and implicit goal of a KGE project:
  - (Explicit goal - purpose) How much the final KG is able to satisfy the Competency Queries ?
    - (Schema layer) Evaluation of CQs vs KG's ETG
    - (Data layer) Evaluation of KG connectivity
  - (Implicit goal - reusability) How much reusable is the final KG ?
    - Evaluation ETG vs Reference Ontologies

Evaluation metrics

- iTelos provides a set of metrics to be used for the above evaluations.
- Between them one of the most usefull is:
  - **Coverage**: How much a portion of knowledge (shaped as etypes and properties) is covered by a KG.
- In order to evaluate the **Knowledge layer** of the final KG we exploit the coverage mesuring as follows:
  - (ETG vs CQs) How much the ETG covers the Entities and properties extracted from the CQs.
  - (ETG vs Reference Ontologies) How much the ETG covers the etypes, and properties, extracted from the reference ontologies.

Metric definitions: Coverage



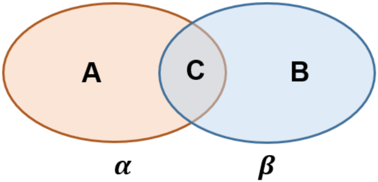
The Coverage is computed as the ratio between the intersection of α and β and the whole α sets:

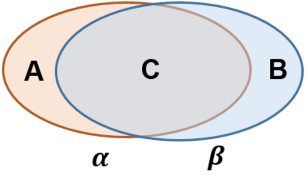
$$Cov = (\alpha \cap \beta)/\alpha = C/(A + C) \quad (1)$$

Where:

- α is a portion of knowledge to be verified.
- β is the KG's Knowledge layer.

Metric definitions: Coverage (extreme cases)





$$\text{Cov} = (\alpha \cap \beta)/\alpha = C/(A + C)$$

$\text{Cov} \simeq 0$

$\text{Cov} \simeq 1$

Metric definitions: Coverage

About the Coverage used to evaluate KGs: $Cov = (\alpha \cap \beta)/\alpha$

- Values are always within the interval [0,1].
- **High values of Coverage** mean that the KG's knowledge is appropriate for the domain.
- For **low values of Coverage**, we can have two possibilities.
    - The reference schema is not appropriate for the domain and maybe a further lookup should be performed.
    - The domain targeted by the knowledge graph is mostly unexplored.

ETG vs CQs

Given a set of (CQ), the **etype coverage** (Cov_E) of the ETG is:

$$Cov_E(CQ_E) = \frac{|CQ_E \cap ETG_E|}{CQ_E} \tag{2}$$

Where:

- CQ_E is the number of etypes extracted from the CQs.
- ETG_E is the number of etypes of the ETG.

Given a set of (CQ), the **property coverage** (Cov_p) of the ETG is:

$$Cov_p(CQ_p) = \frac{|CQ_p \cap ETG_p|}{CQ_p} \tag{3}$$

Where:

- CQ_p is the number of properties extracted from the CQs.
- ETG_p is the number of properties of the ETG.

ETG vs Reference Ontologies (ROs)

Given a set of (RO), the **etype coverage** (Cov_E) of the ETG is:

$$Cov_E(RO_E) = \frac{|RO_E \cap ETG_E|}{RO_E} \tag{4}$$

Where:

- RO_E is the number of etypes extracted from the ROs.
- ETG_E is the number of etypes of the ETG.

Given a set of (RO), the **property coverage** (Cov_p) of the ETG is:

$$Cov_p(RO_p) = \frac{|RO_p \cap ETG_p|}{RO_p} \tag{5}$$

Where:

- RO_p is the number of properties extracted from the ROs.
- ETG_p is the number of properties of the ETG.

The KG’s evaluation - Data layer

- Evaluating the KG’s data layer, aims to understand how **"dense" or "connected"** is the KG.
- The **connectivity** of a KG cna be evaluated over two dimensions:
  - **Entity connectivity**: How much the entities are connected to each other.
  - **Property connectivity**: How much the entities are connected to their properties.

The KG's evaluation - Data layer

- To understand the KG's connectivity we can evaluate two different aspects:
  - **The final result**: this evaluation aims to understand how much connected is the KG at the end of the process.
  - **The construction**: this evaluation aims to understand how much each single dataset, handled during the process, improve the connectivity of the final KG.

- **Note**: the improvement of connectivity brought by a single dataset to the KG, can be different when the dataset is added to the partial KG (during construction), respect to the connectivity evaluated over the same dataset's values, over the final KG.
  - The difference is caused by the **entity matching conflicts** and their solutions.

The KG's evaluation
Data layer final result

To evaluate the connectivity of the final KG, we have to measure:

- **Entity connectivity**:
  - The number of entities E(T) for each etype T in the KG.

    $$\sum_{k=1}^{N} E(T_k)$$, (Where N is the total number of etypes)

  - The number of object property values **not null** Op(T), for each etype T in the KG.

    $$\sum_{k=1}^{N} Op(T_k)$$, (Where N is the total number of etypes)

The KG's evaluation
Data layer final result

To evaluate the connectivity of the final KG, we have to measure:

- **Property connectivity**:
  - The number of data property values **not null** Dp(T), for each etype T in the KG.

    $\sum_{k=1}^{N} Dp(T_k)$, (Where N is the total number of etypes)

The KG's evaluation
Data layer construction

- To evaluate the connectivity improvement brought by a new dataset that has to be integrated into the KG, we have to consider the following cases.

- It is possible to apply the **entity and property connectivity metrics** (see previous slides) to measure the impact of new datasets over the KG, in construction.

- **Assumption**: There are, a new dataset D_1 and the partially built graph KG. Moreover, D_1 has an etype E_1, with its property set A_1 and KG has an etype E_2, with its property set A_2.

The KG's evaluation
Data layer construction

- **Case 1**: $[E_1 = E_2]$ The E_1 in D_1 is already present in KG.
- **Consequence**: By integrating D_1 into KG we are increasing the number of entities of E_1, thus **increasing the entity connectivity**.
  - **Case 1.1**: $[A_1 = A_2]$ The etypes share the same set of properties.
  - **Consequence**: Conflicts are possible between the value set of A_1 and A_2.
    - How many conflicts ?
    - How many new entities from D_1 are integrated into the KG ?
    - How many properties, in the property set A_1, with not null values remain after solving such conflicts ?

The KG's evaluation
Data layer construction

- **Case 1**: $[E_1 = E_2]$ The E_1 in D_1 is already present in KG.
- **Consequence**: By integrating D_1 into KG we are increasing the number of entities of E_1, thus **increasing the entity connectivity**.
  - **Case 1.2**: $[A_1 \neq A_2]$ The etypes have different sets of properties.
  - **Consequence**: There are no conflicts between the value set of A_1 and A_2, and there is a greater increase of the integration over the etype E_1. Notice how in this case also **the property connectivity increases**.
    - How many new entities from D_1 are integrated into the KG ?
    - How many properties, in the property set $A_1 \cup A_2$, with not null values remain after the integration of D_1 ?

The KG's evaluation
Data layer construction

- **Case 2**: $[E_1 \neq E_2]$ The E_1 in D_1 is not yet present in KG.
- **Consequence**: By integrating D_1 into KG we are increasing the number of etypes of KG.
  - **Case 2.1**: E_1 and E_2 are linked by at least one object property.
  - **Consequence**: The resulting KG, after the integration of D_1, is connected.
    - How many connections ?
    - How many entities of E_1 have not null values for the object properties linking E_1 with KG ?

The KG's evaluation
Data layer construction

- **Case 2**: $[E_1 \neq E_2]$ The E_1 in D_1 is not yet present in KG.
- **Consequence**: By integrating D_1 into KG we are increasing the number of etypes of KG.
  - **Case 2.2**: There are no object properties linking E_1 and E_2.
  - **Consequence**: The resulting KG, after the integration of D_1, is not connected.
  - The integration of D_1 doesn't increase the connectivity, thus the information carried by D_1 cannot be reached by the KG.

KGE - Knowledge Graph Engineering

Fausto Giunchiglia

KGC - Data Integration phase

Building an Entity Graph