Information extraction

2. Knowledge representation

Simon Razniewski
Winter semester 2019/20
Announcements

• Assignment results online
• Thanks to all that provided additional info on the survey
• Registration
  • Pass ≥ 6 assignments
  • Register in HISPOS till 7.1.2020
  • No other registration
• Further reading: now on website
Goal today: Model anything

• Anything? Cats (Q146), submarines (Q2811), philosophical schools (Q16895642), ...

• What’s different from databases?
  → Enormously rich schemas
  → Dynamics

• What follows is the standard data model of web-scale KBs, and the semantic web
  • Builds upon Database 101
Motivation (CACM 2019)

“Knowledge representation is a difficult skill to learn on the job. The pace of development and the scale at which knowledge-representation choices impact users and data do not foster an environment in which to understand and explore its principles and alternatives. The importance of knowledge representation in diverse industry settings [...] should reinforce the idea that knowledge representation should be a fundamental part of a computer science curriculum – as fundamental as data structures and algorithms.”

- industry experts behind Google, Microsoft, Facebook, Amazon, IBM knowledge graphs
Outline

• Entities and classes
• Relations
• Binary relations
• Schema
• Knowledge graphs
• Reification
• Canonic entities
• Open-world assumption
• Lab 2
Acknowledgment

• Slides courtesy of Fabian Suchanek (Telecom ParisTech University)
Entity

An entity (also resource, item, object) is any particular object of the world or of imagination, be it abstract or concrete.
Is this a good definition?

Is this an entity?

Or this?

How many entities are there?
Entity permanence?

Over time, all parts of a ship are replaced at some point of time. Then, is it still the same ship?

see: Theseus’s ship on Wikipedia

Humans replace their cells every 7 years
A class (also: concept) is a set of similar entities.

Entities that are not classes (and not literals, relations, ids) are called instances (or common entities).

Classes:
- Actors
- Cars
- Cities
- Rivers
- Universities
- Theories
- ...

Class Actors

Instances

Rowan Atkinson
Marilyn Monroe
Def: Instance of a class

An entity is an instance of a class
(also: belongs to a class, has the type, is of the class),
if the entity is an element of that class.
Def: Subclass, Taxonomy

A class is a subclass of another class, if all instances of the first class are also instances of the second class. A taxonomy is a hierarchy of classes.
Instance vs. class?

If we can say...

• “a/an X”, “every X”
• “Xs” (plural)
• “This is X”
• “X is a Y”
• “Every X is a Y”

then...

X is a class
X is a class
X is an instance of some class
X is an instance of Y
X is a subclass of Y

Try it out: city, Elvis, Coli bacteria, Ford, time

[Finding Needles in an Encyclopedic Haystack: Detecting Classes Among Wikipedia Articles, Pasca, WWW 2018]
Examples

iPhone -> smartphone
finger -> hand
apple -> orange
flower -> plant
Paris -> city
fruit -> food
France -> Europe
## YAGO examples

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Audio book narrators&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Columbia University alumni&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Community organizers&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Democratic Party Presidents of the United States&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Democratic Party United States Senators&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Harvard Law School alumni&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Illinois lawyers&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Illinois State Senators&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Living people&gt;</td>
</tr>
<tr>
<td>&lt;Barack Obama&gt;</td>
<td>rdf:type</td>
<td>&lt;wikicategory Nobel Peace Prize laureates&gt;</td>
</tr>
</tbody>
</table>

*(61 more)*
Limitations

Consider a taxonomy of the animal kingdom.

How do we deal with “male” and “female”?
Limitations

Consider a taxonomy of the animal kingdom.

How do we deal with “male” and “female”?
Intuition: Relations

A relation is like a table.

Relation “born”:

<table>
<thead>
<tr>
<th>Person</th>
<th>City</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atkinson</td>
<td>Consett</td>
<td>1955</td>
</tr>
<tr>
<td>Monroe</td>
<td>Los Angeles</td>
<td>1926</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Def: Relation

A relation (also: predicate) over classes is a subset of their cartesian product. The classes are called the domains of the relation. The number of classes is called the arity of the relation.

\[ R \subseteq C_1 \times C_2 \times \ldots \times C_n \]

\[ \text{born} \subseteq \text{person} \times \text{city} \times \text{year} \]

\[ \text{born} = \{ <\text{Atkinson, Consett, 1955}>, \]

\[ \quad <\text{Monroe, Los Angeles, 1926}>, \ldots \} \]
Def: Binary Relation, Triple

A binary relation is a relation of arity 2.

\[ \text{bornInCity} \subseteq \text{person} \times \text{city} \]

For binary relations, the first class is called the domain and the second class is called the range.

An element of a binary relation is called a fact (or: triple), and we usually visualize it by an arrow:

\[ \text{bornInCity}(\text{Atkinson}, \text{Consett}) \]

The first argument of a fact is the subject, the second the object.
n-ary facts as binary facts

Every n-ary fact can be represented as binary facts.

<table>
<thead>
<tr>
<th>WEDDINGS</th>
<th>Husband</th>
<th>Wife</th>
<th>WeddingDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emmanuel</td>
<td>Brigitte</td>
<td>2007</td>
<td></td>
</tr>
<tr>
<td>Elvis</td>
<td>Priscilla</td>
<td>1967</td>
<td></td>
</tr>
</tbody>
</table>

MacronWedding

- husband: Emmanuel
- wife: Brigitte
- date: 2007

PresleyWedding

- husband: Elvis
- wife: Priscilla
- date: 1967
Def: Event Entity

An event entity represents an n-ary fact.
Task: Event Entities

Draw a knowledge graph for the following facts.

Irma loves Mr. Bean since 1955.
Mr. Bean drives with Irma to the cinema.
Irma and Mr. Bean watch "Titanic".
The movie is about the trip of the ship "Titanic" from Europe to New York.

(There may be multiple solutions)
Binary relations are flexible

n-ary relations enforce the presence of all arguments:
(And nulls blow up the data for high-arity relations)

<table>
<thead>
<tr>
<th>born</th>
<th>Person</th>
<th>City</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Atkinson</td>
<td>Consett</td>
<td>1955</td>
</tr>
</tbody>
</table>

Binary relations don’t:
Binary vs n-ary

Binary and n-ary relations can represent the same facts.

- more relations
- less arity
- more flexibility

- less relations
- more arity
- more control
**Def: Inverse**

The *inverse* of a binary relation \( r \) is a relation \( r' \), such that \( r'(x,y) \) iff \( r(y,x) \).

- \( \text{livesInCity} \subseteq \text{person} \times \text{city} \)
- \( \text{livesInCity}(\text{Atkinson}, \text{Consett}) \) inverses of each other

- \( \text{hasInhabitant} \subseteq \text{city} \times \text{person} \)
- \( \text{hasInhabitant}(\text{Consett}, \text{Atkinson}) \)
Def: Function

A function (also: functional relation) is a binary relation that has at most 1 object for each subject.

\[
r \text{ functional} \equiv \forall x: |\{y : r(x,y)\}| \leq 1
\]

Examples:
- hasBirthPlace
- hasTaxID
- hasNumberOfTeeth
Def: Inverse Functional Rel.

An inverse functional relation is a relation whose inverse is functional.

\[ r \text{ inv. functional} \equiv \forall y: |\{x : r(x,y)\}| \leq 1 \]

Examples:
- hasTaxID
- hasEmailAddress
Functions and inverse functions

• Function + inverse function = identifier
  • has taxCode, VIAF_Identifier

• Use a relation or its inverse?
  • Preference for “more functional” direction
  • Or add both (Wikidata: hasPart/part of, head of government/position held, ...)

29
Equality

If two entities share the same object of an inverse functional relation, they are equal.

\[
\text{hasPassportNumber}(\text{Bean, 29640617}) \\
\text{hasPassportNumber}(\text{MrBean, 29640617}) \\
\Rightarrow \text{MrBean} = \text{Bean}
\]

\[
\text{born}(\text{Bean, 1955}) \\
\text{born}(\text{MrBean, 1955}) \quad \text{(Nothing follows)}
\]
Task: Functions

Which of the following relations are functional?
Def: Name

A name (also: label) of an entity is a human-readable string attached to that entity. The entity is called the meaning of the name.

Entity

Name

"Mr. Bean"

label
Def: Synonymy

If an entity has multiple names, the names are called **synonymous**.

(The adjective for the names is "synonymous", each name is a "synonym", the phenomenon is called "synonymy")

**Donald Trump**  (Q22686)

45th and current president of the United States

Donald John Trump | Donald J. Trump | Trump | The Donald | POTUS 45 | Donald J Trump | President Donald Trump | President Trump | President Donald J. Trump | President Donald John Trump | DJT

- **Rcocin**: Most relevant properties which are absent

**In more languages**

<table>
<thead>
<tr>
<th>Language</th>
<th>Label</th>
<th>Description</th>
<th>Also known as</th>
</tr>
</thead>
</table>
| English  | Donald Trump   | 45th and current president of the United States  | Donald John Trump
Donald J. Trump
Trump
The Donald
POTUS 45
Donald J Trump
President Donald Trump
President Trump
President Donald J. Trump
President Donald John Trump
DJT |
Def: Ambiguity

If a name is attached to multiple entities, the name is called ambiguous.

(The adjective for the names is "ambiguous", the phenomenon is called "ambiguity")

“Schwenker”
Def: Knowledge Graph

A knowledge graph (also: Entity-Relationship graph, Knowledge base, KB) is a directed labeled multi-graph that has an edge $x \rightarrow y$ with label $r$, iff $r(x,y)$.

\[
\text{loves}(\text{Irma}, \text{MrBean}) \\
\text{type}(\text{Irma}, \text{person}) \\
\text{type}(\text{MrBean}, \text{person}) \\
\text{livesIn}(\text{MrBean}, \text{England})
\]
Def: Triple Store

A triple store is a table that contains a KB of binary relations in the form of 3 columns: subject, relation, object.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Relation</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Irma</td>
<td>loves</td>
<td>MrBean</td>
</tr>
<tr>
<td>Irma</td>
<td>type</td>
<td>person</td>
</tr>
</tbody>
</table>

(The middle column is often called “Predicate”)

Popular triple stores are:
• BlazeGraph
• Jena
• Virtuoso
• ...or classical databases
Classes as binary relations

One way to represent a class is by the binary relations $type$, $subclassOf$.

$$type \subseteq entity \times class$$
$$type(Atkinson, actor)$$

$$subclassOf \subseteq class \times class$$
$$subclassOf(actor, person)$$
Digression: Classes and Relations

A fact can be modeled as a class or as a relation.
Domains as binary relations

The domain and range of relations can be expressed by binary relations *domain* and *range*.

\[
domain \subseteq \text{relation} \times \text{class} \\
domain(\text{livesIn, person})
\]

\[
\text{range} \subseteq \text{relation} \times \text{class} \\
\text{range}(\text{livesIn, city})
\]
Def: Schema

The schema/ontology is the part of a knowledge graph that consists of
• the taxonomy (= set of classes with their subclassOf-links)
• relation definitions (= a set of relations with domains and ranges)
Inferences

Further reading:
• RDFS
• OWL
• Description logics
• ...
Task: Schema

1. Define a schema for the domain of movies (guided by statements below).

2. In that schema, express that The Audition is a short film, that De Niro and DiCaprio acted in it, and that Scorsese directed it.
A reified statement is an entity that represents a statement. This phenomenon is called reification.
Reification Vocabulary

\[\text{statement} = \text{class of reified statements}\]
\[\text{subject} \subseteq \text{statement} \times \text{entity}\]
\[\text{predicate} \subseteq \text{statement} \times \text{relation}\]
\[\text{object} \subseteq \text{statement} \times \text{entity}\]
Example: Reification

thinks(Trump, s42)
subject(s42, Johnson)
predicate(s42, type)
object(s42, strong_leader)

Simplified notation:

thinks(Trump, type(Johnson, strong_leader))
Reification and Event Entities

Just as event entities, reification allows higher-order relations and nesting

- **subject**: s42
- **predicate**: Johnson
- **object**: strong_leader
- **type**: type
Task: Reification

Write down a knowledge base with some reified facts.
Can you reify facts that have reified arguments?
**Def: Canonic Entities**

An entity is canonic in a KB, if there is no other entity in the KB that represents the same real-world object.

<table>
<thead>
<tr>
<th>Alizée</th>
<th>produced</th>
<th>Gourmandises</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Jacotey</td>
<td>hasProduced</td>
<td>Psychédélites</td>
</tr>
</tbody>
</table>

... ... ...

not canonical
Def: Canonic Relations

An relation is canonic in a KB, if there is no other relation in the KB that represents the same real-world relation.

\[
\begin{array}{ccc}
\text{Alizee} & \xrightarrow{\text{produced}} & \text{Gourmandises} \\
\text{Alizee} & \xrightarrow{\text{hasProduced}} & \text{Psychédéliques} \\
\ldots & \xrightarrow{\ldots} & \ldots \\
\end{array}
\]

not canonical
Use of Canonicity

Canonicity is essential for

- Counting
- Confidence consolidation
- Constraint satisfaction

```
[| Alizée | produced | Gourmandises |
  | Alizée | hasProduced | Psychédélites |
  | ...    | ...        | ...          |
```

not canonical
# Canonicity and Names

A canonic entity can have multiple names.

<table>
<thead>
<tr>
<th>Alizée</th>
<th>produced</th>
<th>Gourmandises</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alizée</td>
<td>produced</td>
<td>Psychédélies</td>
</tr>
<tr>
<td>Alizée</td>
<td>label</td>
<td>&quot;Alizée&quot;</td>
</tr>
<tr>
<td>Alizée</td>
<td>label</td>
<td>&quot;A. Jacotey&quot;</td>
</tr>
<tr>
<td>produced</td>
<td>label</td>
<td>&quot;produced&quot;</td>
</tr>
<tr>
<td>produced</td>
<td>label</td>
<td>&quot;has produced&quot;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example: Non-canonicity

Open Information Extraction

Argument 1: entity: Donald Trump
Argument 2: 
Relation: 

138 answers from 568 sentences (results truncated)
You were directed to the entity "Donald Trump".
Show all results for "Donald Trump"
Example: Canonicity

<table>
<thead>
<tr>
<th>itemLabel</th>
<th>type</th>
<th>typeLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
<tr>
<td>John Smith</td>
<td>q w.d.Q5</td>
<td>human</td>
</tr>
</tbody>
</table>
Example: Non-Canonicity

Open Information Extraction

"Who built the pyramids?"

<table>
<thead>
<tr>
<th>Argument 1:</th>
<th>Relation: built</th>
</tr>
</thead>
</table>

192 answers from 865 sentences

<table>
<thead>
<tr>
<th>all</th>
<th>person (29)</th>
<th>deceased person (16)</th>
<th>location (13)</th>
<th>monarch (12)</th>
</tr>
</thead>
</table>

Egyptians (278) -> correct

Pharaoh (41) -> not bad

Aliens (35) -> less likely

the Ancient Egyptians (31) -> duplicate

people (25)

https://openie.allenai.org/
Example: Canonicity

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation above sea level</td>
<td>95 metre</td>
<td>0</td>
</tr>
<tr>
<td>Architect</td>
<td>Hemiuunu</td>
<td>1</td>
</tr>
<tr>
<td>Architectural style</td>
<td>ancient Egyptian architecture</td>
<td>0</td>
</tr>
<tr>
<td>Heritage designation</td>
<td>UNESCO World Heritage Site</td>
<td>1979</td>
</tr>
</tbody>
</table>
Canonicity as Trade-Off

non-canonic

- easier to extract
- less easy to use
- more noise
- more data

canonic

- difficult to extract
- easy to use
- less noise
- less data
What is the meaning of data?

<table>
<thead>
<tr>
<th>won</th>
<th>name</th>
<th>award</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>John</td>
<td>Oscar</td>
</tr>
<tr>
<td></td>
<td>Mary</td>
<td>FieldsMedal</td>
</tr>
<tr>
<td></td>
<td>Bob</td>
<td>DijkstraAward</td>
</tr>
</tbody>
</table>

Closed-world assumption

- \( \text{won}(\text{John}, \text{Oscar})? \) → Yes
- \( \text{won}(\text{Ellen}, \text{DijkstraAward})? \) → No

Open-world assumption

- \( \text{won}(\text{John}, \text{Oscar})? \) → Yes
- \( \text{won}(\text{Ellen}, \text{DijkstraAward})? \) → Maybe

- (Relational) databases traditionally employ the closed-world assumption (CWA)
- KBs necessarily operate under the open-world assumption (OWA)
Open-world assumption

- Q: *Hamlet* written by Goethe?
  KB: Maybe
- Q: Schwarzenegger lives in Dudweiler?
  KB: Maybe
- Q: Trump brother of Kim Jong Un?
  KB: Maybe

→ Open-world assumption can be absurd
How to proceed?

• Formal solution:

  Partial-closed world assumption
  • Uses additional metadata to record where OWA/CWA should be applied

• Practical implementation:
  • Obtaining metadata not trivial
  • Application-specific
Outline

- Entities and classes
- Relations
- Binary relations
- Schema
- Knowledge graphs
- Reification
- Canonic entities
- Open-world assumption
- Lab 2
Lab 2

• Goals:
  • 1. Model a domain
  • 2. Get to know SpaCy
POS tagging

• Libraries: spaCy

```python
import spacy

nlp = spacy.load("en_core_web_sm")

text = ("We import the import that was like a like."")
doc = nlp(text)

for token in doc:
    print(token.text, token.pos_)
```

https://spacy.io/usage/spacy-101
https://spacy.io/api/annotation#pos-tagging
Dependency parsing

```python
# print(token.text, token.pos_, token.dep_)

import spacy
nlp = spacy.load("en_core_web_sm")
text = ("Mary likes icecream. John gives Mary a book.")
doc = nlp(text)

for token in doc:
    print(token.text, token.dep_, token.is_space)
    for w in token.children:
        print("  " + w.text, w.dep_)

spacy.displacy.render(doc, style='dep')
```

- [https://spacy.io/api/annotation#dependency-parsing](https://spacy.io/api/annotation#dependency-parsing) -> English
Take home

• Triples can express everything
  • Event entities
  • Reification
• Schema as part of the data
• Canonicity vs. redundancy
• Interpretation of KB data needs caution