A Note on Tutorial 1

• Question 1:
  1. Judge if it is a basic fact or a complex fact (complex facts always have some unknown/unspecified entities)
  2. If it is a complex fact, decide whether it is pattern 1 (about some) or pattern 2 (about all)

• Question 2:
  1. Transform the statements into DL syntax
  2. Check what needs to be proved
  3. Check what reasoning service is needed; in this case, we need Instance Checking --- Mike : MountainClimber
A Note on Tutorial 1

- “Mike dislikes whatever Tony likes and likes whatever Tony dislikes”:
  1. Mike dislikes whatever Tony likes
     1. $\exists \text{like}\cdot\{\text{Tony}\} \subseteq \neg \exists \text{like}\cdot\{\text{Mike}\}
  2. And (Mike) likes whatever Tony dislikes
     1. $\neg \exists \text{like}\cdot\{\text{Tony}\} \subseteq \exists \text{like}\cdot\{\text{Mlike}\}$
  3. Axiom (1.1) can be rewritten into
     1. $\exists \text{like}\cdot\{\text{Mike}\} \subseteq \neg \exists \text{like}\cdot\{\text{Tony}\}$
  4. Axioms (2.1) and (3.1) together are equivalent to the following axiom:
     1. $\exists \text{like}\cdot\{\text{Mike}\} \equiv \neg \exists \text{like}\cdot\{\text{Tony}\}$

Lecture Outline

- Motivation
- Overview of KGs Construction
- Detailed Discussions on Competency Questions
- Practical
Knowledge Graph

- **Knowledge Graph (KG):** KB of interconnected entities with vocabulary defined by a schema

- **Three levels of knowledge**
  - Entity
  - Triple (unit of a multi-relational graph)
  - Schema (defining the vocabulary, making KGs different from other graphs)

[Photo: https://www.w3.org/TR/rdf11-primer/]

Freebase

-Alan Turing- Description

  Alan Mathison Turing, OBE, FRS (23 June 1912 – 7 June 1954) was an English mathematician, logician, and cryptographer. Turing is often considered to be the father of modern computer science. Turing provided an influential formalisation of the concept of the algorithm and

  Description
  Alan Mathison Turing, OBE, FRS (23 June 1912 – 7 June 1954) was an English mathematician, logician, and cryptographer. Turing is often considered to be the father of modern computer science. Turing provided an influential formalisation of the concept of the algorithm and
Freebase and Google

Google Acquires Metaweb To Make Search Smarter

Google has bought semantic search startup Metaweb, according to recent post on the search giant's blog. Terms of the deal were not disclose.

Metaweb develops both semantic data storage infrastructure for the web, and Freebase, an "open, shared database of the world's knowledge". Freebase is a massive, collaboratively edited database of cross-linked data. The idea behind the product is to create a system for building the semantic web. Freebase allows anyone to contribute, structure, search, copy and use data. It sounds like Wikipedia, but instead of arranging by articles, it is more of an almanac, organized like a database, and readable by people and software. You can read our previous coverage of Freebase here.

Clearly, Google is acquiring Metaweb to boost its own search offerings. Metaweb's database of tagged data will help make Google search smarter. And Freebase will be maintained as a knowledge graph.

Knowledge Graphs

Jeff Z. Pan

Question Answering in IBM Watson

A 1992 movie starring Anthony Hopkins was based on an 1897 book by whom?.

• About a billion frames (shallow knowledge graphs) were used in Watson Jeopardy!

Bram Stoker’s famous book on vampires was published in which year?

Knowledge Graphs

Jeff Z. Pan
Knowledge Graphs

Jeff Z. Pan

Product Knowledge Graphs (PKG)

- Many Companies produced PKGs:
  - Amazon (2018)
  - Walmark (2019)
  - Jingdong (2019)
  - Microsoft (2019)
  - Alibaba (2020)

- Key applications
  - Improving search
  - Improving recommendation
  - Question answering
  - Shopping guide
  - Product classification
  - Product completion
Lecture Outline

- Motivation
- Overview of KG Construction
- Detailed Discussions on Data Lifting

Knowledge Construction and Maintenance (KCM) Lifecycle
KCM Lifecycle: Specification

- **Requirement:** competency questions
  - for specifying what the target KG should cover
- **Data source identification and documentation**
  - internal vs external
- **Data source analysis**
  - identify entities and relations covered
  - identify their schema
- **URI/IRI design**
  - use meaningful URIs
  - separate schema (ontology) and data (resource) URIs/IRIs, like in DBPedia

KCM Lifecycle: Schema Modelling

- Generate authoring tests from requirements
- KG schema: Ontology
- Test Driven Ontology Construction
  - try to reuse existing schema and vocabulary for passing the authoring tests
  - Check e.g. existing repositories such as LOV and semic.eu
  - Further improve the KG to pass all the authoring tests
Constructing an Ontology (from scratch)

- Join a small group (2-3 people)
  - Take out a piece of paper
- We are going to
  - Construct an animal ontology
  - To index a children's book of animals
  - Key dimensions for competency questions
    - Where they live
    - What they eat
    - How dangerous they are
    - How big they are

Collect Important Concepts

- Draw a small box in your paper
- Write down up to 10 important concepts
- Discuss in your group to agree upon the set of important concepts
  - Dog
  - Cat
  - ...
Extend the Concepts ‘Laddering’

• Take a group of things and ask
  – What they have in common
  – What other ‘siblings’ there might be

• Example:
  – Cat, Dog => Mammal

Identify Connections

• Choose concepts that can be defined based on others
  – Distinguish atomic concepts from definable concepts
  – Be aware of the level of details that one needs

• Identify relations
  – Such as “eats” …
… and Now We Have

- **Living Thing**
  - Animal
    - Mammal
      - Cat
      - Dog
      - Cow
      - Person
    - Fish
      - Carp
      - Goldfish
  - Plant
    - Tree
    - Grass
    - Fruit

- **Modifiers**
  - domestic
    - Pet
    - Farmed
      - Draft
      - Food
  - Wild
  - Health
    - Healthy
    - Sick
  - Gender
    - Male
    - Female
  - Age
    - Adult
    - Child

- **Relations**
  - eats
  - owns
  - parent-of
  - …

- **Definable**
  - Carnivore
  - Herbivore
  - Child
  - Parent
  - Mother
  - Father
  - Food Animal
  - Draft Animal

---

**Identify Domains and Ranges for Relations**

- Animal *eats* Living thing
  - ObjectProperty (*eats* domain (Animal) range (LivingThing))

- “If an individual has an eats relationship then it must be an instance of the class Animal”

- “If an individual is a value of the eats relationship then it must be an instance of the class LivingThing”
Clarify the Definitions

- A ‘Parent’ is an animal that is the parent of some other animal
  - Class (Parent complete intersectionOf (Animal restriction (parent_of someValuesFrom (Animal))))

- What does it mean?
  1. If an individual is an instance of the class Animal and has some parent_of relationship with some instance of the Animal class, then it is an instance of the Parent class (sufficient condition)
  2. If an individual is an instance of the Parent class, then it is an instance of the class Animal and has some parent_of relationship with some instance of the Animal class (necessary condition)

Lecture Outline

- Motivation
- Overview KG Construction
- Detailed Discussions on Data Lifting
KCM Lifecycle: Data Lifting

Knowledge Extraction:
- Extractor for texts
- Extractor for tables
- Type classifier

Knowledge Base
- beliefs
- candidate facts

Data Resources (e.g., corpora)

Subsystem Components
- CPL
- CSEAL
- CMC
- RL

How is DBpedia Constructed?

---

Knowledge Graphs
Jeff Z. Pan
Knowledge Graphs
Jeff Z. Pan

DBpedia Knowledge Graph: Construction

- Mapping Wikipedia Infobox template elements to elements in DBPedia elements
  - Mappings are created in a community-driven process
- Example: suppose we consider some wikipedia pages about actors

```
{{TemplateMapping
| mapToClass = Actor
| mappings =
  {{ PropertyMapping | templateProperty = name | ontologyProperty = foaf:name }}
  {{ PropertyMapping | templateProperty = birth_place | ontologyProperty = birthPlace }}
```

- `dbpedia:Vince_Vaughn foaf:name "Vince Vaughn"@en .`
- `dbpedia:Vince_Vaughn dbpedia-owl:birthPlace dbpedia:Minneapolis .`

Reasoning with Schema: Why?

Without DOLCE Lite ontology, a total of 97,749 statements (0.65% of all statements) was found to be inconsistent, with 630 different axioms involved in the corresponding explanations. With DOLCE Lite ontology, this number increases to 3,654,255 statements (24.36% of all statements), with 1,467 axioms involved in the corresponding explanations.

[Credit: Paulheim and Gangemi]
### KCM Lifecycle: Data Lifting

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Name</th>
<th>take-course</th>
</tr>
</thead>
<tbody>
<tr>
<td>p001</td>
<td>John</td>
<td>cs3019</td>
</tr>
<tr>
<td>p002</td>
<td>Tom</td>
<td>cs3023</td>
</tr>
</tbody>
</table>

- [csd:p001 rdf:type csd:Student .]
- [csd:p002 rdf:type csd:Student .]
- [csd:p001 csd:name “John” .]
- [csd:p002 csd:name “Tom” .]
- [csd:p001 csd:take-course csd:cs3019 .]
- [csd:p002 csd:take-course csd:cs3023 .]

### Lifting Tables to KG

- **Table to RDF Mapping:**
  - Table and column to class
  - Inter-column relation to data or object property
  - Cell to entity

**Shortcomings:** rely on table headers, often with incomplete semantics!

[Credit: J Chen]
### Lifting Tables to KG

- **Table to RDF Mapping:**
  - Table and column to class
  - Inter-column relation to data or object property
  - Cell to entity

<table>
<thead>
<tr>
<th>Stock</th>
<th>URL</th>
<th>Name</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMZN</td>
<td><a href="http://www.amazon.com">www.amazon.com</a></td>
<td>Jeff Bezos</td>
<td>USA</td>
</tr>
<tr>
<td>GOOG</td>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>Larry Page</td>
<td>USA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stock</th>
<th>Name</th>
<th>Sector</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMZN</td>
<td>Amazon</td>
<td>AWS Sector</td>
</tr>
<tr>
<td>GOOG</td>
<td>Alphabet</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ticker</th>
<th>Date</th>
<th>Income ($B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMZN</td>
<td>06/01/2018</td>
<td>177.86 billion</td>
</tr>
<tr>
<td>GOOG</td>
<td>05/31/2018</td>
<td>92.64 billion</td>
</tr>
</tbody>
</table>

**Semantic annotation:**
- Name of Basic_info by class Company and object property hasCompany.
- Lead to more semantics: Company is a sub-class of Organization; the definition of hasCompany ...

### Texts to Semantic Network

"Spain keeper Iker Casillas equalled Edwin van der Sar's record of nine clean sheets."
Separating Concepts and Individuals

Knowledge Graphs
Jeff Z. Pan

31

32
Identifying Concept/Property Subsumption and Individual Classifications

- Player
- Record
- Performance
- Keeper
- Spain
- Iker Casillas
- Edwin van der Sar
- Iker’s record
- Edwin’s record
- Clean sheet

Identifying Object Properties and Datatype Properties

- Player
- Record
- Performance
- Keeper
- Spain
- Iker Casillas
- Edwin van der Sar
- Iker’s record
- Edwin’s record
- Clean sheet
The procedure of constructing an ontology (KG schema, TBox) is also a procedure in which you examine and refine your semantic networks!

A typical example: multi-entity relationships
- E.g. “Six players from Barcelona were in the Spanish starting line-up”
  - This is a complex relationship involving a "Country's team", a "club team", and "6" players
- In semantic networks, one may draw a figure like this or similar:

- This is not semantically precise:
  - Barcelona has 6 players?
  - No, it has 6 players in the Spanish start line-up
  - Barcelona represents Spain?
  - No, its 6 players represent Spain in the start line-up
- This relationship makes sense only when you take all these entities into account!

An alternative representation — there could be others
- Constructing ontology helps you re-evaluate and refine your semantic network.
Lecture Outline

• Motivation: KG is widely used in downstream applications
• Introduction: Knowledge Construction and Maintenance Lifecycle
• Focus: Modelling and Data Lifting
• Exercises (Next time we introduce Description Logics in details)
  – Young ⊓ Person
  – ∀has_pet.Young
  – Class (Parent complete intersectionOf (Animal restriction (parent_of someValuesFrom (Animal))))