Knowledge Graphs: Theory & Applications

Ian Horrocks & Jiaoyan Chen
University of Oxford & Oxford Semantic Technologies
What is a Knowledge Graph?

Architectural Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Height</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>324</td>
<td>Paris</td>
</tr>
<tr>
<td>Shard</td>
<td>310</td>
<td>London</td>
</tr>
</tbody>
</table>

Tower

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Building</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>Paris</td>
<td></td>
</tr>
<tr>
<td>Shard</td>
<td>London</td>
<td></td>
</tr>
</tbody>
</table>

City

<table>
<thead>
<tr>
<th>Name</th>
<th>Capital Of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>France</td>
</tr>
<tr>
<td>London</td>
<td>UK</td>
</tr>
</tbody>
</table>

Member

<table>
<thead>
<tr>
<th>Country</th>
<th>Union</th>
</tr>
</thead>
<tbody>
<tr>
<td>France</td>
<td>EU</td>
</tr>
<tr>
<td>UK</td>
<td>EU</td>
</tr>
</tbody>
</table>
Why use a Knowledge Graph?

✔ Intuitive (e.g., no “foreign keys”)
Why use a Knowledge Graph?

Architectural Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Height</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>324</td>
<td>Paris</td>
</tr>
<tr>
<td>Shard</td>
<td>310</td>
<td>London</td>
</tr>
</tbody>
</table>

Tower

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Height</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td></td>
<td>324</td>
<td>Paris</td>
</tr>
<tr>
<td>Shard</td>
<td></td>
<td>310</td>
<td>London</td>
</tr>
</tbody>
</table>

City

<table>
<thead>
<tr>
<th>Name</th>
<th>Capital Of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>France</td>
</tr>
<tr>
<td>London</td>
<td>UK</td>
</tr>
</tbody>
</table>

Building

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td></td>
</tr>
<tr>
<td>Shard</td>
<td></td>
</tr>
</tbody>
</table>

Member

<table>
<thead>
<tr>
<th>Country</th>
<th>Union</th>
</tr>
</thead>
<tbody>
<tr>
<td>France</td>
<td>EU</td>
</tr>
<tr>
<td>UK</td>
<td>EU</td>
</tr>
</tbody>
</table>
Why use a Knowledge Graph?

- Intuitive (e.g., no “foreign keys”)
- Data + schema (ontology)
Why use a Knowledge Graph?

Architectural Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Height</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eiffel Tower</td>
<td>324</td>
<td>Paris</td>
</tr>
<tr>
<td>Shard</td>
<td>310</td>
<td>London</td>
</tr>
</tbody>
</table>

City

<table>
<thead>
<tr>
<th>Name</th>
<th>Capital Of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>France</td>
</tr>
<tr>
<td>London</td>
<td>UK</td>
</tr>
</tbody>
</table>

Member

<table>
<thead>
<tr>
<th>Country</th>
<th>Union</th>
</tr>
</thead>
<tbody>
<tr>
<td>France</td>
<td>EU</td>
</tr>
<tr>
<td>UK</td>
<td>EU</td>
</tr>
</tbody>
</table>
Why use a Knowledge Graph?

- Intuitive (e.g., no “foreign keys”)
- Data + schema (ontology)
- URIs not strings
Why use a Knowledge Graph?

- Intuitive (e.g., no “foreign keys”)
- Data + schema (ontology)
- URIs not strings
- Flexible & extensible
Why use a Knowledge Graph?

- Intuitive (e.g., no “foreign keys”)
- Data + schema (ontology)
- URIs not strings
- Flexible & extensible
Why use a Knowledge Graph?

- Intuitive (e.g., no “foreign keys”)
- Data + schema (ontology)
- URIs not strings
- Flexible & extensible
- Rule language
  - location + capital → location
Why use a Knowledge Graph?

- Intuitive (e.g., no “foreign keys”)
- **Data + schema (ontology)**
- **URIs** not strings
- Flexible & extensible
- **Rule language**
  - location + capital → location
- **Other kinds of query**
  - navigation
  - similarity & locality
Knowledge Graph Use Cases
Use Cases

• Open Q&A
Use Cases

• Open Q&A
• Configuration
Use Cases

• Open Q&A
• Configuration
• Data integration
Use Cases

- Open Q&A
- Configuration
- Data integration
- Data wrangling & AI
Use Cases

• Open Q&A
• Configuration
• Data integration
• Data wrangling & AI
• Semantic search & browse
Use Cases

• Open Q&A
• Configuration
• Data integration
• Data wrangling & AI
• Semantic search & browse
Use Cases

• Open Q&A
• Configuration
• Data integration
• Data wrangling & AI
• Semantic search & browse
Use Cases

• Open Q&A
• Configuration
• Data integration
• Data wrangling & AI
• Semantic search & browse
Use Cases

• Open Q&A
• Configuration
• Data integration
• Data wrangling & AI
• Semantic search & browse
Technical Challenges
Knowledge Graph Systems

- Complexity
- Languages + algorithms
Knowledge Graph Systems

• Complexity
  • Languages + algorithms

• Scalability
  • Sheer size of KGs + rules/queries
Knowledge Graph Systems

• Complexity
  • Languages + algorithms

• Scalability
  • Sheer size of KGs + rules/queries

• Extensions
  • Aggregation, constraints, NAF, ...

“Find the average of the surrounding nodes”

[?a, average, ?avg] :-
AGGREGATE([?a, number, ?no]
ON ?a BIND AVERAGE(?no) AS ?avg) .
Knowledge Graph Systems

- Complexity
  - Languages + algorithms
- Scalability
  - Sheer size of KGs + rules/queries
- Extensions
  - Aggregation, constraints, NAF, ...
- System architecture
  - Security, persistence, APIs, ...
Knowledge Graph Systems

• Complexity
  • Languages + algorithms

• Scalability
  • Sheer size of KGs + rules/queries

• Extensions
  • Aggregation, constraints, NAF, ...

• System architecture
  • Security, persistence, APIs, ...

• KG Construction and Curation
Technical Challenges

Knowledge Graph Construction and Curation

• Table to Knowledge Graph Matching
• Ontology Embedding
• Ontology Alignment
• Knowledge Graph Refinement
• Knowledge Graph-based Zero-shot Learning
# Tables to KG (Example #1)

<table>
<thead>
<tr>
<th>StockTicker</th>
<th>Name</th>
<th>GICS Sector</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMZN</td>
<td>Amazon</td>
<td></td>
</tr>
<tr>
<td>GOOG</td>
<td>Alphabet</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stock</th>
<th>URL</th>
<th>CEO 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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ticker</th>
<th>Date</th>
<th>Income ($)</th>
<th>Liabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMZN</td>
<td>06/01/2018</td>
<td>177.86 billion</td>
<td>...</td>
</tr>
</tbody>
</table>

### Background_Info
- StockTicker: AMZN, GOOG
- Name: Amazon, Alphabet
- GICS Sector: ...

### Basic_Info
- StockTicker: AMZN
- Name: Amazon
- GICS Sector: ...

### Finance_Result
- Ticker: AMZN
- Date: 06/01/2018
- Income: 177.86 billion
- Liabilities: ...

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

### Shortcomings:
1. Rely on table headers such as column name
2. Often with incomplete semantics
Tables to KG (Example #1)

**Basic_Info**

<table>
<thead>
<tr>
<th>StockTicker</th>
<th>Name</th>
<th>GICS Sector</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMZN</td>
<td>Amazon</td>
<td>...</td>
</tr>
<tr>
<td>GOOG</td>
<td>Alphabet</td>
<td>...</td>
</tr>
</tbody>
</table>

**Background_Info**

<table>
<thead>
<tr>
<th>Stock</th>
<th>URL</th>
<th>CEO 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>
</tbody>
</table>

**Finance_Result**

<table>
<thead>
<tr>
<th>Ticker</th>
<th>Date</th>
<th>Income ($)</th>
<th>Liabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMZN</td>
<td>06/01/2018</td>
<td>177.86 billion</td>
<td>...</td>
</tr>
</tbody>
</table>

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

**Semantic annotation:**
- Match table semantics to KG semantics
- Name by class `Company` and name by object property `hasCompany`;
- **Lead to more semantics:** Company is a sub-class of `Organization`;
- `hasCompany`’s domain and range ...
- Enable KG population and knowledge integration

**External KGs (ontologies)**

- DBpedia
- Wikidata

**KG Schema**
Tables to KG (Example #2)

- **Semantic annotation (cell by entity):**
  - Sebastian Vettel = dbp:Vettel
  - Ferrari = dbp:Ferrari
  - Germany = dbp:Germany

- **New knowledge extraction and KG population**
  - Hamilton races-for Mercedes?
  - Hamilton lives-in England?
  - Hamilton rdf:type Racing Driver?
  - ......

<table>
<thead>
<tr>
<th>Alonso</th>
<th>McLaren</th>
<th>Spain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamilton</td>
<td>Mercedes</td>
<td>England</td>
</tr>
<tr>
<td>Sebastian Vettel</td>
<td>Ferrari</td>
<td>Germany</td>
</tr>
</tbody>
</table>

Table on Formula 1

Existing KG on Formula 1
Column Type Annotation

1. Does not assume the existence of metadata
2. Train and predict with Convolutional Neural Networks to explore contextual semantics i.e., inter-cell correlations and transfer learning to bridge the knowledge gap
3. Relies on KB (KG) lookup and SPARQL query for automatic sampling


SemTab: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching. In conjunction with ISWC. (https://www.cs.ox.ac.uk/isg/challenges/sem-tab/)
Technical Challenges

Knowledge Graph Construction and Curation

• Table to Knowledge Graph Matching
• **Ontology Embedding**
• Ontology Alignment
• Knowledge Graph Refinement
• Knowledge Graph-based Zero-shot Learning
OWL Ontologies

- **TBox**
  - Atomic concepts (classes) and atomic roles (relations/object properties)
  - Complex classes and relation properties (in description logic)
  - Constraints, subsumption, etc.

- **ABox**
  - Individuals (instances)
  - Membership assertions, role assertions (facts)

- **Meta data and literals**
  - Annotation properties: label, synonym, comment, definition, numeric values, etc.
Embedding Semantics of OWL Ontology

Semantic embedding: represent entities and their semantics (e.g., relationships) in a vector space

<table>
<thead>
<tr>
<th>Methods</th>
<th>Graph Structure</th>
<th>Axioms</th>
<th>(Text) Literals</th>
<th>Logical Constructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical KG embedding: RDF2Vec, TransE, HAKE, etc.</td>
<td>Yes</td>
<td>No</td>
<td>Possible</td>
<td>Ad-hoc for some simple ones</td>
</tr>
<tr>
<td>Onto2Vec</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Opa2Vec</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>EL Embedding</td>
<td>Partial</td>
<td>No</td>
<td>No</td>
<td>Partial ($\ell P^+$)</td>
</tr>
<tr>
<td>Quantum Embedding</td>
<td>Partial</td>
<td>No</td>
<td>No</td>
<td>Partial ($\mathcal{ALC}$)</td>
</tr>
<tr>
<td>OWL2Vec</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>OWL2Vec</strong>*</td>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
</tr>
</tbody>
</table>
From OWL Ontology to RDF Graph
- Reasoning
- W3C OWL to RDF Graph Mapping
- Projection rules

2. Documents (sequences)
- Structure Document
  - Sequences of entities (URIs)
  - Random walk, Weisfeiler-Lehman (WL) subtree kernel, Axioms (OWL Manchester Syntax)
- Lexical Document
  - Sequences of words
  - Text from annotation properties, from structure document
- Combined Document
  - Hybrid sequences of words and URIs

3. Word2Vec
   1. CBOW, pre-training, URI and/or word vectors

4. Extension (TODO): Transformer-based contextual word embedding and pre-trained language models e.g., BERT

https://github.com/KRR-Oxford/OWL2Vec-Star
Applications

• Ontology completion
  • Class membership and subsumption prediction
  • Evaluated on HeLis, FoodOn and GO with higher accuracy than baselines in the above table

• Others
  • Entity clustering
  • Ontology alignment
  • Neural-symbolic AI
  • etc.

Technical Challenges

Knowledge Graph Construction and Curation

• Table to Knowledge Graph Matching
• Ontology Embedding
• Ontology Alignment
• Knowledge Graph Refinement
• Knowledge Graph-based Zero-shot Learning
LogMap and LogMap-ML

- LogMap is a classic ontology matching system based on **lexical matching** (by index), **structure matching** and reasoning-based repair
- Leading performance in many tasks

- A ML framework to augment LogMap and other ontology alignment systems
  - **Distant supervision** by selecting precise mappings from the original system with the assistance of class disjointness (branch conflict rules)
  - **Class embedding** by OWL2Vec* or Word2Vec; feature learning from paths by Siamese Neural Network

All about LogMap:
https://github.com/ernestojimenezruiz/logmap-matcher
Case Study

- **HeLiS**
  - Knowledge on food and healthy lifestyles

- **FoodOn**
  - Detailed food knowledge and knowledge of relevant domains such as agriculture, chemistry and environment

- **Purpose**
  - Knowledge integration & quality assurance e.g., “Soybean Milk” is categorized as “Beverage” but not as “Soybean Food Product” in FoodOn

- **Challenges**
  - Ambiguity on classes with similar names or similar neighborhood structures
  - Branch conflicting mappings e.g., “Caesar’s Mushrooms” vs “Caesar’s Mushroom”, due to different categorizations

- **Results**
  - LogMap-ML outperforms classic systems LogMap and AML, especially on Recall

<table>
<thead>
<tr>
<th></th>
<th>Classes #</th>
<th>Instances #</th>
<th>Axioms #</th>
<th>Description Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>HeLiS</td>
<td>277</td>
<td>20,318</td>
<td>172,213</td>
<td>(\mathcal{ALCHIQ}(D))</td>
</tr>
<tr>
<td>FoodOn</td>
<td>28,182</td>
<td>359</td>
<td>241,581</td>
<td>(\mathcal{SRIQ})</td>
</tr>
</tbody>
</table>

On going work!

Key idea: **fine-tuning** a pre-trained language model (BERT) as a **synonym classifier** for discovering mappings, by task specific corpora which include

1. **intra-ontology corpus** (pairs of labels of the same class of the input ontology)
2. **inter-ontology corpus** (pairs of labels of mapped classes)
3. **complementary corpus** (pairs of labels of the same class from an external ontology)
**KG Entity Alignment**

- **Benchmarking** study: challenges, limitations of embedding-based solutions, and new datasets [COLING’20]
  - Suffer from sample shortage and domain adaption
  - Lexical matching and reasoning-based systems such as LogMap and PARIS outperform state-of-the-art embedding-based methods

- **PARSE**: combine probabilistic reasoning (PR) by PARIS e.g., utilizing relation functionality to identify equivalent entities with semantic embeddings (SE) [IJCAI’21][CIKM’21]
  - Mappings by PR for the training of SE
  - Similarity by SE for augmenting PR

---

**KG Entity Alignment**

![Diagram of KG Entity Alignment](image)

- **KG Entity Alignment**
  - **KG1**
    - Agent
    - Person
    - ORG
    - Politician
    - Royalty
  - **KG2**
    - Organisation
    - University
    - Educational Organization
    - Queen Victoria
    - predecessor
    - employer
    - William IV
    - John Brown
    - Lovan Ho
    - rector
    - VICTORIA
    - locationCity
    - Frogmore Estate

- **PR Module**
- **SE Module**
- **Entity Mappings (\(\tilde{Y}^P\))**
- **Entity Embeddings (\(\tilde{E}^E\))**
- **Unaligned Candidates (\(\tilde{U}^P\))**
- **Alignment Seeds (\(S\))**
- **Output**

![Diagram of KG Entity Alignment](image)

- **KG Entity Alignment**
  - **KG1**
    - Agent
    - Person
    - ORG
    - Politician
    - Royalty
  - **KG2**
    - Organisation
    - University
    - Educational Organization
    - Queen Victoria
    - predecessor
    - employer
    - William IV
    - John Brown
    - Lovan Ho
    - rector
    - VICTORIA
    - locationCity
    - Frogmore Estate

- **PR Module**
- **SE Module**
- **Entity Mappings (\(\tilde{Y}^P\))**
- **Entity Embeddings (\(\tilde{E}^E\))**
- **Unaligned Candidates (\(\tilde{U}^P\))**
- **Alignment Seeds (\(S\))**
- **Output**
Technical Challenges

Knowledge Graph Construction and Curation

• Table to Knowledge Graph Matching
• Ontology Embedding
• Ontology Alignment
• Knowledge Graph Refinement
• Knowledge Graph-based Zero-shot Learning
Canonicalizing Literals

For each class $c$ in $C_{PM}$, sampling (triple $< s, p, l >$, label) Particular & General samples & Sample refinement

Neural Network (Classifier)
- Triple $< s, p, l >$ to Sequence $[word_1, ..., word_T]$
- Bidirectional RNNs + Attention Layer

For each class $c$ in $C_{PM}$

Training

Sample: (triple $< s, p, l >$, label)

Correcting Facts

- Informally represented facts
  - <Yangtze_River, passesArea, “three gorges district”>
  - “tree gorges district”, as a phrase, is short of semantics such as types and sometimes leads to confusion (errors)

- Erroneous facts caused by confusions
  - <Sergio_Agüero, playsFor, Manchester_United> to <Sergio_Agüero, playsFor, Manchester_City>

A lexical matching, KG embedding and reasoning-based correction framework

Technical Challenges

Knowledge Graph Construction and Curation

• Table to Knowledge Graph Matching
• Ontology Embedding
• Ontology Alignment
• Knowledge Graph Refinement
• Knowledge Graph-based Zero-shot Learning
Knowledge-driven Zero-shot Learning (ZSL)

• What is ZSL?
  • Predict samples with new classes that have never appeared in training
  • Seen classes vs unseen classes

Zero-shot Image Classification

Zero-shot Knowledge Graph Completion (Inductive Settings)
Knowledge-driven Zero-shot Learning

1. Textual description:
   “Zebras are white animals with black stripes, they have larger, rounder ears than horses ...”

2. Attribute descriptions, e.g., visual properties of animals

3. Taxonomy (category)

4. Knowledge Graph (Relational Facts + Categories + Literals)

5. Logics & rules

---

OntoZSL

An ontology-based general ZSL framework (has been evaluated with image classification and KG completion)

- **Ontology Encoder** which embeds the ontological schema and learns a vector representation for each class (i.e., class embedding).
- **Feature Extractor** which extracts the features of raw samples (e.g., ResNet), used in training the generation model.
- **Generation Model (Generative Adversarial Network Discriminator and Generator)** which generates features for unseen classes conditioned on their class embeddings.
- **Zero-shot Classifier** which learns to predict the testing samples of unseen classes.

Summary
Summary

• KGs are powerful tool for representing & reasoning about knowledge

• Many applications: semantic search and browse, ...

• System challenges: complexity, scalability, extensions, architecture, ...

• Application challenges: KG construction and curation
Thanks for Listening
Any Questions?

Background reading:

• **Basic algorithms & data structures** Motik et al. *Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems*. AAAI 2014.


• **Modular materialisation** Hu et al. *Modular Materialisation of Datalog Programs*. AAAI 2019.


• **KG Curation** Chen et al. *An Assertion and Alignment Correction Framework for Large Scale Knowledge Bases*. Semantic Web Journal.