Knowledge Graph Curation with Deep Learning and Semantic Reasoning

Jiaoyan Chen (陈矫彦)
Department of Computer Science, University of Oxford

jiaoyan.chen@cs.ox.ac.uk
https://www.cs.ox.ac.uk/ism/people/jiaoyan.chen/
Knowledge Graph (KG)

- The Knowledge Graph is a **knowledge base** used by **Google** and its services to enhance its search engine's results with information gathered from a **variety of sources**.
- Proposed around 2012

-------- From Wikipedia
Semantic Web

- The Semantic Web is an extension of the World Wide Web through standards by the World Wide Web Consortium (W3C)
  - URI: Uniform Resource Identification
  - RDF (Resource Description Framework)
    - Triple: <Subject, Property, Object>
  - RDF Schema
    - e.g., class hierarchy, property domain and range
  - Web Ontology Language (OWL)
  - SPARQL (RDF query language)
  - Semantic Reasoning

**KG**: RDF triples + RDF Schema or OWL Ontology
KG Applications

• Search engines

• Chat robots e.g., Alexa and Siri

• Data analytics and machine learning e.g., explanation

• Intelligent finance

• Knowledge engineering

• ...
From Data Curation to Knowledge Graph Curation

- **Data curation** (策展): A series of data management and cleaning tasks e.g., organization, integration, annotation, publication, maintenance, reuse, preservation, etc.

- From data curation to **KG Curation**:
  - Knowledge boosting
    - E.g., extract RDF triples from text, tabular data
  - Refinement
    - Canonicalization, error detection and correction
  - Quality measurement
  - Etc.
Tabular Data Matching
(for Data Analytics and KG Boosting)
Case Study

<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>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>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>

- W3C RDB to RDF (Ontology) Mapping
  - table “Basic_Info” into ontology class,
  - foreign key into object property
  - column “Income” into data property,
  - column “CEO_Name” into data property,
  - Etc.

However, such straight forward matching brings limited semantics.

E.g., What does “Jeff Bezos” mean?
Case Study

<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>

- W3C RDB to RDF (Ontology) Mapping (e.g., BootOX)
  - table “Basic_Info” into ontology class,
  - foreign key into object property
  - column Income into data property,
  - column “CEO_Name” into data property,
  - Etc.

- Semantic matching:
  - Column "Column Name" by class Company, column “CEO_Name” by class Person
  - Columns “Stock” and “CEO_Name” by property has_ceo,
  - Cell “Jeff Bezos” by entity Jeff_Preston_Bezos
  - Etc.
Tasks of Table Matching

• Task #1: *Entity Column* → *Class*
  - e.g., Column “CEO Name” to Class CEO and Class Person
  - equal to *Table* → *Class matching* if the entity column is a primary key

• Task #2: *Cell* → *Entity*
  - e.g., “Jeff Bezos” to Entity Jeff_Preston_Bezos
  - equal to *Row* → *Class matching* if the column is a primary key

• Task #3: *Column Pair* → *Property*
  - e.g., “Stock” and “CEO Name” → property has_ceo
  - includes data property and object property
  - two columns can come from two tables
  - column order matters
Related Work (1)
-- Probabilistic Graphical Model (PGM)

• Steps
  • Models tasks of #1, #2 and #3 with a PGM (e.g., Markov Random Fields): variable for table component; value for KG component
    • E.g., cell to entity by string similarity, column header to class
    • Searches for an assignment of values to the variables to maximize the joint probability

• Examples
  • [Limaye et al. 2010, Mulwad et al. 2013, Bhagavatula et al. 2015, Takeoka et al. 2019]

• Limitations:
  • Mostly rely on meta data
  • Low scalability and efficiency

• Mulwad, V., Finin, T., & Joshi, A. (2013). Semantic Message Passing for Generating Linked Data from Tables. In ISWC.
Related Work (2)
-- Iterative Approach

• A bootstrapping approach with two iterations (TableMiner+ [Zhang 2017]):
  • Get some initial matchings using partial data from the table e.g., column type by column header
  • Uses the initial matchings as constraints to interpret the rest matchings of the table

• Multiple iterations with an overall score (T2K Match [Ritze 2015])

• Limitations:
  • Performance drops without meta data
  • Hard to capture the contextual semantics

• Zhang, Z. (2017). Effective and efficient Semantic Table Interpretation using TableMiner+. Semantic Web, 8(6), 921–957.
Related Work (3)
--Deep Learning for the Semantics

• Disambiguate by learning representation of the context of the KG and the table
  • [Efthymiou 2017]: semantic embeddings of KG entities
  • [Luo 2018]: learn table features of cells

• Limitations
  • The context has not been fully explored (e.g., semantics of surrounding relations)
  • Focus more on cell to entity matching, but less on column type matching and relation matching
  • Need labeled sample sets (labor cost, generalization, transferability)

• Fetahu, et al. (2019), TableNet: An Approach for Determining Fine-grained Relations for Wikipedia Tables. WWW
Related Work (4)
-- From cell to entity and knowledge gap

• A straight forward solution: match cells to entities, and then determine the type by majority voting
  • e.g., [Zwicklbauer 2013]

• Knowledge gap: When (a part of) cells have no entity correspondences
  • [Pham 2016]: compare a target column with a set of labeled columns by machine learning
    • It’s hard to get enough labeled columns
  • [Quercini 2013]: Predict the type with web page queried by a search engine
    • It introduces additional noise and irrelevant content, both of which changes over time

• Quercini, G., Reynaud-Delaitre, C., & Reynaud, C. (2013). Entity Discovery and Annotation in Tables. EDBT/ICDT.
ColNet: An automatic column type matching framework
Problem: Column Type Prediction

• Problem
  • Annotating a column whose cells are text phrases (i.e., entity mentions) with classes of a KG, **without** assuming any metadata (e.g., column headers)

• Example
  • Annotate the following column “Name” by classes *Company* and *IT Company*

<table>
<thead>
<tr>
<th>Name</th>
<th>Employee #</th>
<th>CEO</th>
<th>Income ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>85,050</td>
<td>Sundar Pichar</td>
<td>...</td>
</tr>
<tr>
<td>Amazon</td>
<td>613,300</td>
<td>Jeff Bezos</td>
<td>...</td>
</tr>
<tr>
<td>Apple</td>
<td>132,000</td>
<td>Arthur D. Levinson</td>
<td>...</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>4,044</td>
<td>John S. Chen</td>
<td>...</td>
</tr>
<tr>
<td>Orange</td>
<td>155,202</td>
<td>Stéphane Richard</td>
<td>...</td>
</tr>
</tbody>
</table>
Difficulty in Column Type Prediction

• Multiple and hierarchical output classes
• Identify fine-grained but completely correct classes
  • In the above example: Company (√), IT Company (√√), US Company (X)
• Disambiguation
  • “BlackBerry” to “BlackBerry Limited (Company)” or “BlackBerry (Mobile Phone)” or “Blackberry (Fruit)”
• Column cells may have few or even empty KB entity correspondences (Knowledge Gap)
ColNet in a Nutshell

• Does **not** assume the existence of **metadata**

• Column Classifier
  • Convolutional Neural Networks (CNNs) to explore **contextual semantics** i.e., inter-cell correlations

• Knowledge-based learning
  • Relies on KG lookup and SPARQL query (reasoning) for **automatic sampling**
  • Transfer learning to bridge the **knowledge gap**
Step #1: Entity Lookup

Example

<table>
<thead>
<tr>
<th>Google</th>
<th>“Google”, “Amazon.com”, “Amazon rainforest”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>“Apple Inc.”, “Apple (fruit)”, “OS X”</td>
</tr>
<tr>
<td>Apple</td>
<td>“BlackBerry Limited”, “Blackberry (fruit)”</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>“Orange (fruit)”, “Orange County”, “Orange S.A.”, etc.</td>
</tr>
</tbody>
</table>
Step #2: Sampling & Training

Candidate Classes

Matched Entities

Entities & Triples

Schema

Knowledge Graph

Particular Entities

General Entities

Negative Entities

Synthetic Columns

CNNs

Candidate Class Example: US Company


“Google”

“Amazon.com”

“Apple Inc.”

“BlackBerry Limited”

Synthetic Column

A binary classifier for US Company
Step #2: Sampling & Training (some details)

• One-vs-rest
  • Unfixed candidate classes, multiple and hierarchical output classes

• Synthetic column (size fixed): learn inter-cell correlations by CNNs
  • E.g., Given a column [“Google”, “BlackBerry”, “Apple”], predicting cell by cell gives a final score from 0.33 to 0.66 to Company, while considering their correlations increases the score to around 1.0

• Transfer learning and automatic sampling
  • Pre-training with general entities; fine-tuning with particular entities

• Negative sampling
  • Entities of disjoint classes
  • Give higher priority to classes that appear together in a column
Step #2: Sampling & Training (CNN)
Step #3: Prediction & Typing (Example 1)

A target column of IT companies

Google
Amazon
Apple
BlackBerry
Orange

Google
Amazon
Apple
BlackBerry
Orange

Predicted score: $p^c$

Company: 0.9
IT Company: 0.78
US Company: 0.38
Fruit: 0.08
Software: 0.03
Forest: 0.02
Administrative Region: 0.01
Etc.

Predict & Average

Synthetic Columns

Company: 1.0
IT Company: 1.0
US Company: 0.8
Fruit: 0.6
Forest: 0.2
Etc.

Voted score: $v^c$

$\sigma_1 = 0.9$
$\sigma_2 = 0.1$

$\tilde{s}^c = \begin{cases} v^c, & \text{if } v^c \geq \sigma_1 \text{ or } v^c < \sigma_2 \\ p^c, & \text{otherwise} \end{cases}$

Low knowledge gap
Step #3: Prediction & Typing (Example 2 with big knowledge gap)

A target column with large knowledge gap

Oxford Semantic Technology
DeepReason.ai
Oxstem
Oxbotica
Tripadvisor

Predict & Average

Predicted score: $p^c$

Company: 0.65
IT Company: 0.45
University: 0.21
Research Institute: 0.51
Etc.

Threshold: 0.5

Ensemble

Voted score: $v^c$

$\sigma^c = \begin{cases} 
  v^c, & \text{if } v^c \geq \sigma_1 \text{ or } v^c < \sigma_2 \\
  p^c, & \text{otherwise} 
\end{cases} 
\sigma_1 = 0.9 
\sigma_2 = 0.1$

High knowledge gap

Top-2 matching over DBPedia

KG Lookup & Vote
Evaluation #1: Web Table

- **DBpedia** as the KG, DBPedia lookup service, SPARQL endpoint
- **Word2vec**, trained by Wikipedia article dump
- **Limaye** (tables from Wikipedia pages), **T2Dv2** (tables from the Web)
  - Both have two sets of ground truth classes -- “Best” and “Okay”, resulting to two evaluation models – “Strict” and “Tolerant”.

<table>
<thead>
<tr>
<th>Name</th>
<th>Columns</th>
<th>Avg. Cells</th>
<th>Different “Best” (&quot;Tolerant&quot;) Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limaye</td>
<td>428</td>
<td>23</td>
<td>21 (24)</td>
</tr>
<tr>
<td>T2Dv2</td>
<td>411</td>
<td>124</td>
<td>56 (35)</td>
</tr>
</tbody>
</table>
Overall Results on Limaye

Precision, recall, F1 score (PK = Primary Key)

<table>
<thead>
<tr>
<th>Models</th>
<th>Methods</th>
<th>PK Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ColNet Ensemble</td>
<td>0.796, 0.799, 0.798</td>
</tr>
<tr>
<td></td>
<td>ColNet</td>
<td>0.762, 0.820, 0.791</td>
</tr>
<tr>
<td></td>
<td>Lookup-Vote</td>
<td>0.732, 0.660, 0.694</td>
</tr>
<tr>
<td></td>
<td>T2K Match</td>
<td>0.560, 0.408, 0.472</td>
</tr>
<tr>
<td></td>
<td>Efthymiou17-Vote</td>
<td>0.759, 0.414, 0.536</td>
</tr>
<tr>
<td>Tolerant</td>
<td>ColNet Ensemble</td>
<td>0.602, 0.639, 0.620</td>
</tr>
<tr>
<td></td>
<td>ColNet</td>
<td>0.574, 0.619, 0.597</td>
</tr>
<tr>
<td></td>
<td>Lookup-Vote</td>
<td>0.571, 0.447, 0.501</td>
</tr>
<tr>
<td></td>
<td>T2K Match</td>
<td>0.453, 0.330, 0.382</td>
</tr>
<tr>
<td></td>
<td>Efthymiou17-Vote</td>
<td>0.626, 0.357, 0.454</td>
</tr>
<tr>
<td>Strict</td>
<td>ColNet Ensemble</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ColNet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lookup-Vote</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T2K Match</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Efthymiou17-Vote</td>
<td></td>
</tr>
</tbody>
</table>

- **Prediction impact**
  - ColNet\textsubscript{ensemble} and ColNet > Lookup-Vote
  - Improve recall dramatically

- **Ensemble impact**
  - ColNet\textsubscript{ensemble} > ColNet
  - Improve precision dramatically

- **Comparison with the state-of-the-art**
  - ColNet\textsubscript{ensemble} and ColNet > T2K Match
  - ColNet\textsubscript{ensemble} and ColNet has competitive precision as Efthymiou17-Vote, but much higher recall and F1 score

- **Baselines**
  - Lookup-Vote: DBPedia entity lookup + Voting
  - T2K Match (iterative approach)
  - Efthymiou17-Vote: entity matching by Efthymiou et al. 2017 (entity embedding) + Voting
Overall Results on T2Dv2

Prediction impact √

Ensemble impact √

Comparison with the state-of-the-art √

Knowledge gap impact

Limaye has shorter columns in average, which leads to larger knowledge gap

The absolute precision and recall is higher on T2Dv2 than on Limaye

Improvements of ColNet ensemble and ColNet are more significant on Limaye than on T2Dv2, especially w.r.t. Lookup-Vote

---

<table>
<thead>
<tr>
<th>Models</th>
<th>Methods</th>
<th>PK Columns</th>
<th>All Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolerant</td>
<td>ColNetEnsemble</td>
<td>0.796, 0.799, 0.798</td>
<td>0.917, 0.909, 0.913</td>
</tr>
<tr>
<td></td>
<td>ColNet</td>
<td>0.763, 0.820, 0.791</td>
<td>0.845, 0.896, 0.870</td>
</tr>
<tr>
<td></td>
<td>Lookup-Vote</td>
<td>0.732, 0.660, 0.694</td>
<td>0.909, 0.865, 0.886</td>
</tr>
<tr>
<td></td>
<td>T2K Match</td>
<td>0.560, 0.408, 0.472</td>
<td>0.664, 0.773, 0.715</td>
</tr>
<tr>
<td></td>
<td>Ethymiou17-Vote</td>
<td>0.759, 0.414, 0.536</td>
<td>0.853, 0.846, 0.849</td>
</tr>
<tr>
<td>Strict</td>
<td>ColNetEnsemble</td>
<td>0.602, 0.639, 0.620</td>
<td>0.941, 0.958, 0.949</td>
</tr>
<tr>
<td></td>
<td>ColNet</td>
<td>0.576, 0.619, 0.597</td>
<td>0.765, 0.811, 0.787</td>
</tr>
<tr>
<td></td>
<td>Lookup-Vote</td>
<td>0.571, 0.447, 0.501</td>
<td>0.862, 0.821, 0.841</td>
</tr>
<tr>
<td></td>
<td>T2K Match</td>
<td>0.453, 0.330, 0.382</td>
<td>0.624, 0.727, 0.671</td>
</tr>
<tr>
<td></td>
<td>Ethymiou17-Vote</td>
<td>0.626, 0.357, 0.454</td>
<td>0.729, 0.884, 0.799</td>
</tr>
</tbody>
</table>

---

Precision, recall, F1 score (PK = Primary Key)

Results on Limaye

Results on T2Dv2
Evaluation #2: CSV Files (AIDA)

• Three AIDA data sets:
  • Tundra Traits, Broadband, Home Electricity Survey
  • 20 different CSV files
  • 81 entity columns
### Broadband

#### November 2014
- **# rows**: 1,971
- **Col. id**: 3
- **Col. name**: Urban/rural
- **# unique**: 3
- **Type**: Place
- **Source**: dbpedia

#### November 2015
- **# rows**: 2,802
- **Col. id**: 4
- **Col. name**: Pack_number
- **# unique**: 38
- **Type**: broadband
- **Source**: wikidata (Q194163)

#### November 2016
- **# rows**: 4,748
- **Col. id**: 1
- **Col. name**: PANEL
- **# unique**: 2
- **Type**: Organisation
- **Source**: dbpedia

### Home Electricity Survey

#### Appliance Codes
- **# rows**: 256
- **Col. id**: 1
- **Col. name**: Name
- **# unique**: 255
- **Type**: Device
- **Source**: dbpedia

#### Appliance Data
- **# rows**: 4,600
- **Col. id**: 2
- **Col. name**: Room
- **# unique**: 30
- **Type**: ArchitecturalStructure
- **Source**: dbpedia

#### Appliance Groups
- **# rows**: 57
- **Col. id**: 1
- **Col. name**: ApplianceGroup
- **# unique**: 57
- **Type**: Device
- **Source**: dbpedia

#### Diary Tumble Dryer
- **# rows**: 495
- **Col. id**: 2
- **Col. name**: Programme
- **# unique**: 65
- **Type**: washing
- **Source**: wikidata (Q23841)

#### Ipsos Anonymised
- **# rows**: 251
- **Col. id**: 8
- **Col. name**: AnnualMonthly
- **# unique**: 3
- **Type**: frequency
- **Source**: wikidata (Q115652)

### Tundra Trait

#### File
- **# rows**: 66,308
- **Col. id**: 1
- **Col. name**: AccSpeciesName
- **# unique**: 538
- **Type**: Plant
- **Source**: dbpedia

#### Dataset Final
- **# rows**: 66,308
- **Col. id**: 8
- **Col. name**: SubsiteName
- **# unique**: 239
- **Type**: Location
- **Source**: wikidata (Q47574)
Evaluation #2: CSV Files

- Comparison with Baseline:
  - ColNet > Lookup-Vote, ColNet > T2K Match
ColNet Extension: Richer Context for The Classifier
Column Classifier Nutshell

• **Semantics from surrounding rows and columns**

<table>
<thead>
<tr>
<th>Col 1.A</th>
<th>Col 1.B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>Arthur D. Levinson</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>John S. Chen</td>
</tr>
<tr>
<td>Orange</td>
<td>Stéphane Richard</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Col 2.A</th>
<th>Col 2.B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>Vitamin C 7%</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>Vitamin C 35%</td>
</tr>
<tr>
<td>Orange</td>
<td>Vitamin C 88%</td>
</tr>
</tbody>
</table>

Impact of the surrounding column: Col 1.A belongs to *Company* while Col 2.A belongs to *Fruit*

• **Hybrid Neural Network** (HNN): table feature learning that captures the contextual semantics – cell phrases and cell correlations

• A KG lookup and semantic query algorithm for **column features**
HNN Insights and Architecture

- **Bidirectional Recurrent Neural Networks (BiRNNs)**
  - Cell phrase embedding with word correlations

- **Attention Layer**
  - Higher weight on “Limited” than on “BlackBerry” w.r.t. Company;

- **Convolution Layer**
  - Encode the correlation between cells within a column or a row;
  - e.g., Conv(“Apple”, “Google Inc.”) > “Apple”
Remaining Challenges

• **Unfixed relative column position** makes it different from traditional feature learning for text and images

• The potential relations (properties) between columns are quite predictive and transferable
  
  • E.g., “has_CEO” vs. “has_Vitamin”

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>Arthur D. Levinson</td>
<td>Apple</td>
<td>Vitamin C 7%</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>John S. Chen</td>
<td>BlackBerry</td>
<td>Vitamin C 35%</td>
</tr>
<tr>
<td>Orange</td>
<td>Stéphane Richard</td>
<td>Orange</td>
<td>Vitamin C 88%</td>
</tr>
</tbody>
</table>

• However, the neural network fails to capture such **semantic relations**
Property Vector (P2Vec)

- P2Vec is a multi-hot vector: one slot represents the likelihood of one property
- Example: assume that has_CEO corresponds to the first slot, has_Vitamin corresponds to the second last slot

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>Arthur D. Levinson</td>
<td>Apple</td>
<td>Vitamin C 7%</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>John S. Chen</td>
<td>BlackBerry</td>
<td>Vitamin C 35%</td>
</tr>
<tr>
<td>Orange</td>
<td>Stéphane Richard</td>
<td>Orange</td>
<td>Vitamin C 88%</td>
</tr>
</tbody>
</table>

\[ [1, 0, 0, \ldots, 0, 0] \quad \text{vs} \quad [0, 0, 0, \ldots, 1, 0] \]
Property Vector (P2Vec)

A Brief Procedure for P2Vec Calculation
Ensemble: HNN + P2Vec

I. Train a classifier
   • with the input of HNN features (intermediate output) and P2Vec
   • using e.g., Multiple Layer Perception (MLP) and Logistic Regression (LR)

II. Average the predicted score by
    • a classifier trained with the input of P2Vec using e.g., MLP and LR
    • the HNN
Experiment

• Data
  • KG: DBpedia
  • Table sets: Limaye (8 classes, 114 columns), Efthymiou (31 classes, 620 columns), T2Dv2 (37 classes, 411 columns)

• Settings
  • Transfer: T2D-Tr (70% of T2Dv2) for training; Limaye, Efthymiou and T2D-Te (30% of T2Dv2) for testing
  • Independent: for each table set, 70% for training; 30% for testing

• Evaluation
  • Performance of HNN, P2Vec, ensemble approaches
  • Overall accuracy in comparison with baselines
Overall results (accuracy)

- HNN + P2Vec > CNNs used in ColNet, in both independent and transfer settings
- HNN + P2Vec > lexical matching based baselines (Lookup-Vote and T2K Match), in the independent setting
- Transfer setting:
  - HNN + P2Vec > Lookup-Vote, T2K Match on Limaye
  - Lookup-Vote > HNN + P2Vec > T2K Match on Efthymiou
  - A good compromise is using 10% of local data plus transferred data from training

<table>
<thead>
<tr>
<th>Methods (Training Data)</th>
<th>T2D-Te</th>
<th>Limaye</th>
<th>Efthymiou</th>
</tr>
</thead>
<tbody>
<tr>
<td>HNN + P2Vec (T2D-Tr)</td>
<td>0.966</td>
<td>0.746</td>
<td>0.650</td>
</tr>
<tr>
<td>HNN + P2vec (Local-70%)</td>
<td>0.968</td>
<td>0.907</td>
<td>0.865</td>
</tr>
<tr>
<td>HNN + P2vec (T2D-Tr + Local-10%)</td>
<td>-</td>
<td>0.907</td>
<td>0.697</td>
</tr>
<tr>
<td>Lookup-Vote</td>
<td>0.835</td>
<td>0.868</td>
<td>0.827</td>
</tr>
<tr>
<td>T2K Match</td>
<td>0.772</td>
<td>0.807</td>
<td>0.612</td>
</tr>
<tr>
<td>ColNet (T2D-Tr)</td>
<td>0.947</td>
<td>0.597</td>
<td>0.619</td>
</tr>
<tr>
<td>ColNet (Local-70%)</td>
<td>0.912</td>
<td>0.912</td>
<td>0.813</td>
</tr>
</tbody>
</table>
For more

• Papers

• More project information:
  • https://github.com/alan-turing-institute/SemAIDA/
  • https://www.turing.ac.uk/research/research-projects/artificial-intelligence-data-analytics-aida

• Semantic Web Challenge on Tabular Data to Knowledge Graph Matching
  • ISWC 2019
  • With large synthetic and real data for all three tasks
  • http://www.cs.ox.ac.uk/isg/challenges/sem-tab/
Knowledge Graph Correction
Problem

- Triple assertions with literal objects (entity mentions)
  - e.g., `<Yangtze_River, passesArea, “three gorges district”>`;
  - Such literal objects **fail to capture the semantics** like type and linkage;
  - Quite common:
    - in KGs from wiki like DBpedia and zhishi.me
    - in KGs from tables like LinkedGeoData
    - when multiple KGs are aligned

- Triple assertions with erroneous objects
  - e.g., `<Sergio_Agüero, playsFor, Manchester_United>` (Agüero actually plays for Manchester City)
  - such errors are often caused by semantic or lexical confusions
Solution

• Step #1: Detection
  • Downstream applications
  • Logic rules and constraints (e.g., SPARQL query template [Kontokostas et al. WWW 2014])
  • Machine learning methods e.g., semantic embedding, outlier detection and supervised learning

• Step #2: Correction
  • Find an entity from the KB for substitution,
    • “three gorges district” to Three_Gorges_Reservoir_Region
    • <Sergio_Agüero, playsFor, Manchester_United> to <Sergio_Agüero, playsFor, Manchester_City>

• Step #3: Canonicalization
  • Create a new entity for substitution if no existing entity can be found in Step #2

Type of the entity substitute is useful.
Nutshell for Typing

• A **knowledge based learning** framework for literal object classification

• **BiRNNs** + **attention** for exploring the contextual semantics – literal and triple

• The role of **constraints**
  • High quality negative sampling
    • Get negative samples (entities) from the sibling classes of a target class
    • Ensure that the entity cannot be inferred
  • Hierarchical prediction (utilizing disjointness)
    • Place: 0.9, Professor: 0.8 → adopt Place;
    • Place: 0.9, Town: 0.8 → adopt both Place and Town
Framework

For each literal, predict types, lookup entities

Literal Typing & Canonicalization
“Three Gorges District” – Location and BodyOfWater (Types)

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

Training

Sampling
Sample: (triple $<s,p,l>$, label)
Particular & General samples
Sample refinement

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

Candidate Classes $C_{PM}$

Query $E_M$

Literal Matching

Query

Classes $C_M$

Classes $C_P$

Property Objects $E_P$

$<$Subject, Property, Literal$>$’s

$\rightarrow$, passesArea, “Three Gorges District”,
$\rightarrow$, passesArea, “London, England”
Attention Recurrent Neural Network

$\mathbf{f}(s, p, l)$

- **Word Vectors**
  - "Yangtze" "River"
  - "passes" "Area"
  - "Three" "Gorges" "District"
  - Subject $s$
  - Predicate $p$
  - Literal $l$

- **Bidirectional RNNs**

- **Attention Layer**

- **FC Layer + Logistic Regression**

- **Query**
  - $\mathbf{x}_t, t \in [1, T]$
  - $\mathbf{h}_t, t \in [T, 1]$
  - $\mathbf{\bar{h}}_t, t \in [1, T]$

- **Word Attentions**
  - $u_w$
  - $\alpha_t, t \in [1, T]$
Conclusion and Outlook
Discussion and Outlook

• Semantic table annotation and KG boosting
  • Column type matching
  • Property matching
  • Robust feature learning

• KG correction
  • Detection
  • Correction and canonicalization
  • Heterogeneous KG (alignment of multiple sources)

• KG & Machine learning
  • Machine learning explanation
  • Integration of learning and reasoning (neural-symbolic)
• Main contributors:
  • Ian Horrocks (University of Oxford)
  • Ernesto Jimenez-Ruiz (City, University of London & University of Oslo)
  • Charles Sutton (The University of Edinburgh & Google)
  • Maximilian Pflueger (Ph.D student, University of Oxford)

• Partners and Sponsor:
  • SIRUS, University of Oslo
  • Alan Turing Institute, London
  • Samsung Research, London (new)
  • Siemens, München (new)
Thanks for Your Attentions!