Web Science

Tran, Thi Ngoc Han
ngoc-han.thi.tran@stud.uni-hannover.de
Content

● Fun Facts: Automatic Trivia Fact Extraction from Wikipedia
  ○ Motivation
  ○ Method
  ○ Evaluation

● Automated Template Generation for Question Answering over Knowledge Graphs
  ○ Motivation
  ○ System Overview
  ○ Template Generation
  ○ Question Answering with Templates

● Discussion
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

What are Trivia Facts?
→ unimportant facts or details. Facts about people, events, etc. that are not well-known (*The Merriam-Webster*)
→ trivia-worthy
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

- Motivation
  - Trivia facts contributes to user experience around entity searches
  - Helps increase user engagement
  → automatically find trivia facts about entities from Wikipedia
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

● Method
  ○ Problem Formulation
    ■ Surprise
    ■ Cohesiveness
    ■ Tying it Together
  ○ Algorithm
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

● Problem Formulation
  ○ Surprise
    The similarity of an article $a$ to category $C$ as the average similarity between $a$ and articles of $C$

\[
\sigma(a, C) = \frac{1}{|C| - 1} \sum_{a \neq a' \in C} \sigma(a, a')
\]

Where article-article similarity by $\sigma(a, a')$

\[
surp(a, C) = \frac{1}{\sigma(a, C)}
\]
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

● Problem Formulation
  ○ Cohesiveness: The average similarity between pairs of articles from C

\[
\text{cohesive}(C') = \frac{1}{\binom{|C'|}{2}} \sum_{a \neq a'} \sigma(a, a')
\]

○ Tying it Together

\[
\text{trivia}(a, C') = \text{cohesive}(C') \cdot \text{surp}(a, C')
\]

\[
\text{trivia}(a, C') = \frac{\text{cohesive}(C')}{\sigma(a, C')}
\]

- ~1: the article typical for that category
- <1: the article more similar to other articles than the average
- >1: the article not similar to the category
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

- Problem Formulation - Example

Category: Democratic Party Presidents of the United States

Category: Grammy Award winners

Graph Properties:
- Category: Democratic Party Presidents of the United States
  - $\sigma = 0.601$
  - Cohesiveness = 0.619
  - Trivia = 1.03

- Category: Grammy Award winners
  - $\sigma = 0.241$
  - Cohesiveness = 0.399
  - Trivia = 1.651

Figure 3: Similarity graphs for two categories containing Barack Obama. Thicker edges are more similar. For visualization reasons, not all nodes and edges are shown.
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

Algorithm 1 Top Trivia algorithm

```
function TOPTRIVIA(inputArticle)
    for every category C of inputArticle do
        surprise ← SURPRISE(inputArticle, C)
        cohesiveness ← COHESIVENESS(C)
        C.trivia ← cohesiveness * surprise
    return category C with maximum trivia score
```

```
function SURPRISE(inputArticle, category)
    sum, count ← 0
    for every article a ≠ inputArticle in category C do
        similarity ← ARTICLESIMILARITY(inputArticle, a)
        sum ← sum + similarity
        count ← count + 1
    similarityToCategory ← sum/count
    return similarityToCategory
```

```
function ARTICLESIMILARITY(article1, article2)
    K ← 10
    T1 ← TopTFIDF(article1, K)
    T2 ← TopTFIDF(article2, K)
    similarity ← σ(article1, article2) using equation 3.1
    return similarity
```

```
function COHESIVENESS(category)
    sum, count ← 0
    for every pair of articles a1 ≠ a2 in category C do
        similarity ← ARTICLESIMILARITY(a1, a2)
        sum ← sum + similarity
        count ← count + 1
    cohesiveness ← sum/count
    return cohesiveness
```
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

● Evaluation:
  ○ Compare 4 algorithms
    ■ Wikipedia Trivia Miner (WTM):
      ● a ranking algorithm over Wikipedia sentences
      ● learns the notion of interestingness using domain-independent linguistic and entity based features
      ● the supervised ranking model is trained on existing user-generated trivia data available on the Web.
    ■ Top Trivia: The highest ranking category in this algorithm ranking
    ■ Middle-ranked Trivia: Using middle-of-the-pack ranked categories, as ranked by this algorithm
    ■ Bottom Trivia: Using the lowest-ranked categories by this algorithm
  ○ Dataset: list contains a diverse range of popular people, including politicians, sportspeople, scientists, actors, writers, singers, historical figures and other people of interest
  ○ Evaluation Study: use crowd-sourced work
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

- Evaluation:
  - Evaluation Study: use crowd-sourced work
    - The workers were presented with the fact and asked to express their level of agreement with the following statements:
      - Trivia-worthiness: “This is a good trivia fact".
      - Surprise: “This fact is surprising".
      - Personal knowledge: “I knew this fact before reading it here"
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

- Evaluation

**Figure 4:** Majority opinion about facts being trivia-worthy, by algorithm
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

- Evaluation

**Figure 5:** Majority opinion about facts being surprising, by algorithm

**Figure 6:** Majority opinion about personal knowledge of facts, by algorithm
Fun Facts: Automatic Trivia Fact Extraction from Wikipedia

- Evaluation:
  - Bounced immediately out of the site (under 5 seconds)
    - Bottom Trivia: 52% of users
    - WTM: 47% of users
    - Top Trivia: 37% of users
  - Average time on the site for users who did not bounce
    - Bottom Trivia: 30.7 seconds
    - WTM: 43.1 seconds
    - Top Trivia: 48.5 seconds
Discussion
Content

● Fun Facts: Automatic Trivia Fact Extraction from Wikipedia
  ○ Motivation
  ○ Method
  ○ Evaluation

● Automated Template Generation for Question Answering over Knowledge Graphs
  ○ Motivation
  ○ System Overview
  ○ Template Generation
  ○ Question Answering with Templates

● Discussion
Automated Template Generation for Question Answering over Knowledge Graphs

- **Motivation**
  - Templates play an important role in *Question Answering* over Knowledge Graph
  - Prior works rely on Hand-crafted templates/rules with limited coverage

  → QUINT system
    - Automatically learns utterance-query template from user questions paired with their answers
    - Able to answer complex questions
Automated Template Generation for Question Answering over Knowledge Graphs

- System Overview

```
Template Generation

Role-aligned (Utterance Template, Query Template)

Generalization (Sec. 3.5)

Role-aligned (Utterance, Query)

ILP Alignment (Sec. 3.4)

(Utterance, Backbone Query)

Backbone query & Typing (Sec. 3.1 & 3.2)

(Utterance, Answer)

Question Answering

Template Matching & Instantiation (Sec. 4.1)

Query Ranking (Sec. 4.2)

Top-1 Query Candidate

Answer

New Utterance
```
Automated Template Generation for Question Answering over Knowledge Graphs

- Template Generation - Example
  - Backbone Query Construction
    - Annotate utterance $u$ with named entities using an “off-the-shelf named entity recognition and disambiguation system”
    - For each answer $a$, find the smallest connected subgraph of the KG containing above entities and a
      “Which actress played character $\text{[[Amy Squirrel | AmySquirrel]]}$ on $\text{[[Bad Teacher | BadTeacher]]}$?”

Figure 1: Example KG fragment.

Figure 4: Backbone query $q$. 
Automated Template Generation for Question Answering over Knowledge Graphs

- Template Generation - Example
  - Backbone Query Construction
  - Capturing Answer Types
Automated Template Generation for Question Answering over Knowledge Graphs

- **Template Generation - Example**
  - Backbone Query Construction
  - Capturing Answer Types
  - Utterance-Query Alignment
    - Use Integer Linear Programming (ILP) for alignment -> choose the correct type constraint

![Diagram of Backbone Query with Types](image)

**Figure 5:** Backbone query $\hat{q}$ with types.

![Diagram of Aligned Utterance Query Pair](image)

**Figure 7:** Aligned utterance query pair $(u, q, m)$. $m$ is indicated by shared $\text{ent}$, $\text{pred}$, and $\text{type}$ annotations (e.g., “played on” is aligned with $\text{cast.actor}$).
Automated Template Generation for Question Answering over Knowledge Graphs

- Template Generation - Example
  - Backbone Query Construction
  - Capturing Answer Types
  - Utterance-Query Alignment
  - Generalization to Templates
    - Remove the concrete labels of edges (predicates) and nodes (entities and types)
    - Keep the semantic alignment annotations

\[ q \]

\[ q_t \]
Automated Template Generation for Question Answering over Knowledge Graphs

- System Overview
Automated Template Generation for Question Answering over Knowledge Graphs

- Question Answering for a new utterance $u'$
  - Match it against all templates in repository
  - Rank the queries (due to multiple matching templates or due to ambiguity of phrases in the lexicon)
  - Adopt a learning-to-rank approach to rank the obtained queries and return the highest ranking query

![Diagram](image)

Figure 9: Template instantiation (using $t$ in Figure 8).
Automated Template Generation for Question Answering over Knowledge Graphs

- Answering Complex Questions (composed of multiple clauses)
  - Automated dependency parse rewriting: if there are
    - Coordinating conjunction or relative clause, and
    - Matches against our template repository result in less sub-questions than expected
  - Sub-question answering
    - Each match corresponds to a sub-question that can be answered independently
    - Keep the ranked-list of queries
  - Stitching
    - Return the answers resulting from the combination of queries that the sum of their scores is highest
Automated Template Generation for Question Answering over Knowledge Graphs

- Result
  - On WebQuestions and Free917

<table>
<thead>
<tr>
<th>Method</th>
<th>WebQuestions</th>
<th>Free917</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cai and Yates [10] (2013)</td>
<td>-</td>
<td>59.0</td>
</tr>
<tr>
<td>Berant et al. [4] (2013)</td>
<td>35.7</td>
<td>62.0</td>
</tr>
<tr>
<td>Kwiatkowski et al. [19] (2013)</td>
<td>-</td>
<td>68.0</td>
</tr>
<tr>
<td>Yao and Van Durme [46] (2014)</td>
<td>33.0</td>
<td>-</td>
</tr>
<tr>
<td>Berant and Liang [5] (2014)</td>
<td>39.9</td>
<td>68.6</td>
</tr>
<tr>
<td>Bao et al. [2] (2014)</td>
<td>37.5</td>
<td>-</td>
</tr>
<tr>
<td>Bordes et al. [8] (2014)</td>
<td>39.2</td>
<td>-</td>
</tr>
<tr>
<td>Yao [45] (2015)</td>
<td>44.3</td>
<td>-</td>
</tr>
<tr>
<td>Dong et al. [12] (2015)</td>
<td>40.8</td>
<td>-</td>
</tr>
<tr>
<td>Bast and Haussmann [3] (2015)</td>
<td>49.4</td>
<td>76.4</td>
</tr>
<tr>
<td>Berant and Liang [21] (2015)</td>
<td>49.7</td>
<td>-</td>
</tr>
<tr>
<td>Yih et al. [47] (2015)</td>
<td>52.5</td>
<td>-</td>
</tr>
<tr>
<td>Reddy et al. [28] (2016)</td>
<td>50.3</td>
<td>78.0</td>
</tr>
<tr>
<td><strong>This Work</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUINT-untyped</td>
<td>50.8</td>
<td>78.6</td>
</tr>
<tr>
<td>QUINT</td>
<td>51.0</td>
<td>72.8</td>
</tr>
</tbody>
</table>
Automated Template Generation for Question Answering over Knowledge Graphs

- Result
  - On ComplexQuestions

<table>
<thead>
<tr>
<th>Method</th>
<th>Average F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bast and Haussmann-basic</td>
<td>27.8</td>
</tr>
<tr>
<td>Bast and Haussmann++</td>
<td>46.7</td>
</tr>
<tr>
<td>QUINT</td>
<td>49.2</td>
</tr>
</tbody>
</table>

Table 7: Results on ComplexQuestions.
Automated Template Generation for Question Answering over Knowledge Graphs

- Limitations
  - No template matching
    - Incompleteness of predicate lexicon
    - Incorrect dependency parse trees and POS tag annotations
  - Wrong answers returned
    - Mistakes from NER/NED system
    - Missing entities in lexicon
    - Lack of any appropriate templates for some questions
Discussion