Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base

Scott Wen-tau Yih
Ming-Wei Chang, Xiaodong He, Jianfeng Gao
Question Answering with Knowledge Base

• Large-scale Knowledge Base
  • Properties of billions of entities
  • Plus relations among them

• Question Answering
  “What are the names of Obama’s daughters?”
\[ \lambda x. \text{parent}(\text{Obama}, x) \land \text{gender}(x, \text{Female}) \]
Search Engine → QA Engine

who was Katy Perry's husband

who was Tom Cruise's first wife

when did Minnesota become a state?

Scafell Pike

Map data ©2015 Google
Who is Justin Bieber’s sister?

Jazmyn Bieber
Who is Justin Bieber’s sister?

\[ \lambda x. \text{sibling_of(justin_bieber, } x) \land \text{gender}(x, \text{female}) \]
Key Challenges

• Language mismatch
  • Lots of ways to ask the same question
    “What was the date that Minnesota became a state?”
    “When was the state Minnesota created?”
    “Minnesota's date it entered the union?”
  • Need to map them to the predicate defined in KB
    location.dated_location.date_founded

• Large search space
  • Some Freebase entities have >160,000 immediate neighbors

• Compositionality
Staged Query Graph Generation

Basic idea

• Query graph
  • Resembles subgraphs of the knowledge base
  • Can be directly mapped to a logical form in λ-calculus

• Semantic parsing
  • A search problem that grows the graph through staged state-actions
Staged Query Graph Generation Addresses Key Challenges

- Language mismatch
  - Advanced entity linking
  - Relation matching via deep convolutional NN
- Large search space
  - Representation power of a parse controlled by staged search actions
  - Grounding partially the utterance during search
- Compositionality
  - Possible combinations limited by local subgraphs

52.5% $F_1$ (Accuracy) on WebQuestions
Outline

- Introduction
- Background
  - Graph knowledge base
  - Query graph
- Staged Query Graph Generation (Our Approach)
- Experiments
- Conclusion
Knowledge Base

- Triples of $\text{subj-pred-obj} (e_1, p, e_2)$
- Knowledge graph
  - Each entity is a node
  - Two related entities linked by a directed edge (predicate)
- CVT node
  - Compound value type
  - Encode $n$-ary relations
Who first voiced Meg on Family Guy?

\[ \lambda x. \exists y. \text{cast}(\text{FamilyGuy}, y) \land \text{actor}(y, x) \land \text{character}(y, \text{MegGriffin}) \]
Outline

• Introduction
• Background

• Staged Query Graph Generation (Our Approach)
  • Link topic entity
  • Identify core inferential chain
  • Augment constraints

• Experiments
• Conclusion
Staged Query Graph Generation

- A search problem with staged states and actions

Who first voiced Meg on Family Guy?

(1) **Link Topic Entity**
Who first voiced Meg on *Family Guy*?

(2) **Identify Core Inferential Chain**
Staged Query Graph Generation

Who first voiced Meg on Family Guy?

(3) Augment Constraints
Link Topic Entity

• An advanced entity linking system for short text

• Prepare surface-form lexicon $\mathcal{L}$ for entities in the KB
• Entity mention candidates: all consecutive word sequences in $\mathcal{L}$, scored by the statistical model
• Up to 10 top-ranked entities are considered as topic entity
Identify Core Inferential Chain

• Relationship between topic and answer (x) entities
• Explore two types of paths
  • Length 1 to non-CVT node
  • Length 2 where y can be grounded to CVT

Who first voiced Meg on Family Guy?

{cast-actor, writer-start, genre}
Relation Matching using Deep Convolutional Neural Networks (DSSM [Shen+ 14])

- Input is mapped to two $k$-dimensional vectors
- Probability is determined by softmax of their cosine similarity

$$P(R|P) = \frac{\exp(\cos(y_R, y_P))}{\sum_{R'} \exp(\cos(y_{R'}, y_P))}$$

who voiced meg on <e> cast-actor
Augment Constraints

- **Who** first voiced **Meg** on **Family Guy**?

\[ \lambda x. \exists y. \text{cast}(\text{FamilyGuy}, y) \land \text{actor}(y, x) \]

- One or more constraint nodes can be added to \( y \) or \( x \)
  - \( y \): Additional property of this event (e.g., \( \text{character}(y, \text{MegGriffin}) \))
  - \( x \): Additional property of the answer entity (e.g., \( \text{gender} \))

- Only subset of constraint nodes are considered
  - e.g., entities detected in the question (more detail in Appendix)
Who first voiced Meg on Family Guy?
Who first voiced Meg on *Family Guy*?
Learning Reward Function – Features

- Topic Entity
  - Entity linking scores
- Core Inferential Chain
  - Relation matching scores (NN models)
- Constraints: Keyword and entity matching
  - ConstraintEntityWord(“Meg Griffin”, q) = 0.5
  - ConstraintEntityInQuestion(“Meg Griffin”, q) = 1
- Overall
  - NumNodes(s) = 5
  - NumAnswers(s) = 1

\[ q = \text{Who first voiced Meg on Family Guy?} \]

\[ s = \]
Outline

• Introduction
• Background
• Staged Query Graph Generation (Our Approach)
• Experiments
  • Data & evaluation metric
  • Creating training data from Q/A pairs
  • Results
• Conclusion
WebQuestions Dataset [Berant+ 13]

- What character did Natalie Portman play in Star Wars? ⇒ Padme Amidala
- What currency do you use in Costa Rica? ⇒ Costa Rican colon
- What did Obama study in school? ⇒ political science
- What do Michelle Obama do for a living? ⇒ writer, lawyer
- What killed Sammy Davis Jr? ⇒ throat cancer

[Examples from Berant]

- 5,810 questions crawled from Google Suggest API and answered using Amazon MTurk
  - 3,778 training, 2,032 testing
  - A question may have multiple answers ⇒ using Avg. F1 (≈ accuracy)
Creating Training Data from Q/A Pairs

Relation Matching (Identifying Core Inferential Chain)

- List all the length 1 & 2 paths from any potential topic entity
- Treat any inferential chain resulting in $F_1 \geq 0.5$ to create positive pairs

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Inferential Chain</th>
</tr>
</thead>
<tbody>
<tr>
<td>what was &lt;e&gt; known for</td>
<td>people.person.profession</td>
</tr>
<tr>
<td>what kind of government does &lt;e&gt; have</td>
<td>location.country.form_of_government</td>
</tr>
<tr>
<td>what year were the &lt;e&gt; established</td>
<td>sports.sports_team.founded</td>
</tr>
<tr>
<td>what city was &lt;e&gt; born in</td>
<td>people.person.place_of_birth</td>
</tr>
<tr>
<td>what did &lt;e&gt; die from</td>
<td>people.deceased_person.cause_of_death</td>
</tr>
<tr>
<td>who married &lt;e&gt;</td>
<td>people.person.spouse_s</td>
</tr>
<tr>
<td></td>
<td>people.marriage.spouse</td>
</tr>
</tbody>
</table>
Creating Training Data from Q/A Pairs

Reward Function $\gamma$

- Apply the same best-first search procedure to training data
- Use the $F_1$ score of the query graph as the reward function
- For each question, create 4,000 candidate query graphs
  - All positive ($F_1 > 0$) examples
  - Randomly selected negative examples
Avg. F1 (Accuracy) on WebQuestions Test Set

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yao-14</td>
<td>33</td>
</tr>
<tr>
<td>Berant-13</td>
<td>35.7</td>
</tr>
<tr>
<td>Bao-14</td>
<td>37.5</td>
</tr>
<tr>
<td>Bordes-14b</td>
<td>39.2</td>
</tr>
<tr>
<td>Berant-14</td>
<td>39.9</td>
</tr>
<tr>
<td>Yang-14</td>
<td>41.3</td>
</tr>
<tr>
<td>Yao-15</td>
<td>44.3</td>
</tr>
<tr>
<td>Wang-14</td>
<td>45.3</td>
</tr>
<tr>
<td>Yih-15</td>
<td>52.5</td>
</tr>
</tbody>
</table>
## Contribution from Entity Linking

- Statistics of entity linking results on training set questions

<table>
<thead>
<tr>
<th>Method</th>
<th>#Entities</th>
<th>Covered Ques.</th>
<th>Labeled Ent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freebase API</td>
<td>19,485</td>
<td>98.8%</td>
<td>81.2%</td>
</tr>
<tr>
<td>Yang &amp; Chang, ACL-15</td>
<td>9,147</td>
<td>99.8%</td>
<td>87.8%</td>
</tr>
</tbody>
</table>

- $F_1$ drops from 52.5% to 48.4% when using Freebase API
Contribution from Relation Matching

- $F_1$ score of query graphs that have only a core inferential chain: 49.6 (vs. 52.5 full system)

- Questions from search engine users are short & simple
  - 1,888 (50%) training questions can be answered exactly ($F_1 = 1$)
  - Even if the correct parse requires more constraints, the less constrained graph still gets a partial score
Error Analysis

A random sample of 100 incorrectly answered questions

- Label issues (34%)
  - Label error (2%)
    - Incomplete labels (17%, e.g., “What songs did Bob Dylan write?”)
    - Acceptable answers (15%, e.g., “Time in China” vs. “UTC+8”)
- Incorrect entity linking (8%)
- Incorrect inferential chain (35%)
- Incorrect/Missing constraints (23%)
Conclusions (1/2)

A new framework for semantic parsing of questions

• Query graph
  • Meaning representation that can be directly mapped to logical form, using predicates in target KB

• Semantic parsing
  • Query graph generation as staged search problem

• New state-of-the-art on WebQuestions (52.5 $F_1$)
  • Advanced entity linking
  • Convolutional NN for relation matching
Conclusions (2/2)

• Future Work
  • Improve the current system
    • Matching relations more accurately
    • Handling constraints in a more principled way
  • Joint structured-output prediction model (e.g., SEARN [Daumé III 06])
  • Extend the query graph to represent more complicated questions

• Data & Resource
  • Sent2Vec (DSSM) http://aka.ms/sent2vec
  • System output http://aka.ms/codalab-webq
  • Intermediate files (e.g., entity linking, model files, training data, etc.) will be released soon http://aka.ms/stagg