Lecture 7

Training
Knowledge Base Query Agents
Outline

• **SEMPRE**: Seminal paper on synthesis-based training data acquisition
  • Synthesize 1 canonical sentence per query
  • Train with only manually paraphrased data on paraphrase data
• **Schema2QA**: Manual annotations / paraphrases
  • Synthesize with 900 templates
  • Train with synthesized & manually paraphrased data on real data
• **AUTOQA** – Automatically synthesized + a few shot
  • Synthesize with 900 templates
  • Automatic annotation and paraphrases
  • Train with synthesized + auto-paraphrase data + few-shot
SEMPRE: Building a Semantic Parser Overnight
[Wang et al. 15]

• Pioneered the concept of data synthesis for database queries based on formal query grammar
  • Use domain-independent grammars to generate one canonical training sample per query:
    natural language + (correct) annotation
  • Ask crowdsourced workers to paraphrase the sentences
  • Train and test only with paraphrase data
• Question answering agent overnight
  • Ask crowdsourced workers to paraphrase overnight for new domains!
  • No need to get real users; no need to annotate by hand!
Experimental Results

- Testing on withheld paraphased data
- Train & evaluation: manual paraphrase
- 8 domains
- 26K examples

<table>
<thead>
<tr>
<th>Method</th>
<th>Calendar</th>
<th>Blocks</th>
<th>Housing</th>
<th>Restaurants</th>
<th>Publications</th>
<th>Recipes</th>
<th>Social</th>
<th>Basketball</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL</td>
<td>74.4</td>
<td>41.9</td>
<td>54.0</td>
<td>75.9</td>
<td>59.0</td>
<td>70.8</td>
<td>48.2</td>
<td>46.3</td>
<td>58.8</td>
</tr>
<tr>
<td>NoLex</td>
<td>25.0</td>
<td>35.3</td>
<td>51.9</td>
<td>64.6</td>
<td>50.6</td>
<td>32.3</td>
<td>15.3</td>
<td>19.4</td>
<td>36.8</td>
</tr>
<tr>
<td>NoPPDB</td>
<td>73.2</td>
<td>41.4</td>
<td>54.5</td>
<td>73.8</td>
<td>56.5</td>
<td>68.1</td>
<td>43.6</td>
<td>44.5</td>
<td>56.9</td>
</tr>
<tr>
<td>Baseline</td>
<td>17.3</td>
<td>27.7</td>
<td>45.9</td>
<td>61.3</td>
<td>46.7</td>
<td>26.3</td>
<td>9.7</td>
<td>15.6</td>
<td>31.3</td>
</tr>
</tbody>
</table>
Our Experience to Apply “Overnight” Approach

• Very poor accuracy
  • Tried to improve the model for months.
• Problem 1:
  • Many errors in paraphrasing by crowdworkers
    • Especially when the synthesized sentences were clunky
  • Solution: Correct annotations with
    • Basic simple automatic techniques
    • Manual checking (by crowdworkers)
Our Experience to Apply “Overnight” Approach

• Very poor accuracy
  • Tried to improve the model for months.
• Problem 2
  • Poor training data - not much variety
    • The workers are given the canonical English sentences. That bias the choice of words
    • They try to complete the jobs as quickly as possible.
• Solution:
  • Ask workers to provide several answers - improved somewhat

Quiz: What is the implication of testing on withheld paraphrased data?
Pros and Cons of Overnight

• Cost
  • Cheaper than manual annotation,
    Cost includes manual paraphrase cost + correction

• Coverage [Not complete]
  • Limited templates not covering the space of queries
  • Limited variety in natural language
    • Workers are biased by the synthetic sentence
    • Often only make minimum changes

• Correctness
  • Workers makes mistakes, especially when the synthetic sentence quality is poor

Key takeaway:
  Good for paraphrase test data
  Does not work on real input!
  Need to address the deficiencies
Outline

• SEMPRE: Seminal paper on synthesis-based training data acquisition
  • Synthesize 1 canonical sentence per query
  • Train with only manually paraphrased data on paraphrase data

• Schema2QA: Manual annotations / paraphrases
  • Synthesize with 900 templates
  • Train with synthesized & manually paraphrased data on real data

• AUTOQA – Automatically synthesized + a few shot
  • Synthesize with 900 templates
  • Automatic annotation and paraphrases
  • Train with synthesized + auto-paraphrase data + few-shot
Schema2QA

• Approach: generate question-answering agent from a schema
  • Use many more templates to cover the long tail of compound queries with variety
  • Then ask humans to paraphrase
  • Use more templates on (900)

1. Canonical templates:
   Ensure coverage with a template for every operator
   The templates compose to create compound queries

2. Grammar templates: covering grammar book variations

3. Domain-based templates: use developer-supplied field annotations
1. Canonical Templates: Cover all queries
Composable templates for every relational operator

<table>
<thead>
<tr>
<th>Operation</th>
<th>English Template</th>
<th>ThingTalk</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>table with fname equal to value</td>
<td>table, fname = value</td>
<td>restaurants with rating equal to 3</td>
</tr>
<tr>
<td>Projection</td>
<td>the fname of table</td>
<td>[fname] of table</td>
<td>the cuisine of restaurants</td>
</tr>
<tr>
<td>Subquery</td>
<td>the table1 of table2</td>
<td>table1,in_array(id,any(table1 of table2))</td>
<td>reviews of restaurant X</td>
</tr>
<tr>
<td>Join</td>
<td>table1 with their table2</td>
<td>table join table2</td>
<td>restaurants with their reviews</td>
</tr>
<tr>
<td>Aggregate</td>
<td>the number of table</td>
<td>count (table)</td>
<td>The number of restaurants</td>
</tr>
<tr>
<td>Aggregate &amp; Group by</td>
<td>the op fname in table</td>
<td>op (fname of table)</td>
<td>The average rating of restaurants</td>
</tr>
<tr>
<td>Aggregate &amp; Group by</td>
<td>the op fname1 in table in each fname2</td>
<td>op (fname1 of table by fname2)</td>
<td>The average rating of restaurants</td>
</tr>
<tr>
<td>Ranking</td>
<td>the n table with the min fname</td>
<td>sort (fname asc of table)[1:n]</td>
<td>the 3 restaurants with the min rating</td>
</tr>
<tr>
<td>Quantifier</td>
<td>table1 with table2</td>
<td>table1, contains(table2, any(table2))</td>
<td>restaurants with review with ...</td>
</tr>
<tr>
<td>Quantifier</td>
<td>table1 with no table2</td>
<td>table1, !contains(table2, any(table2))</td>
<td>restaurants with no review with ...</td>
</tr>
<tr>
<td>Row-wise function</td>
<td>the distance of table from location</td>
<td>[distance(geo, location)] of table</td>
<td>The distance of restaurants from here</td>
</tr>
<tr>
<td>Row-wise function</td>
<td>the number of(fname in table)</td>
<td>[count(fname)] of table</td>
<td>The number of reviews in restaurants</td>
</tr>
</tbody>
</table>
## 2. Standard Variation in NL
From English grammar books

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>Lighter, heavier</td>
</tr>
<tr>
<td>Height</td>
<td>Taller, shorter</td>
</tr>
<tr>
<td>Age</td>
<td>Older, younger</td>
</tr>
<tr>
<td>Length</td>
<td>Shorter, longer</td>
</tr>
<tr>
<td>Size</td>
<td>Smaller, bigger</td>
</tr>
<tr>
<td>Price</td>
<td>Cheaper, more expensive</td>
</tr>
<tr>
<td>Speed</td>
<td>Slower, faster</td>
</tr>
<tr>
<td>Temperature</td>
<td>Colder, hotter</td>
</tr>
<tr>
<td>Time</td>
<td>Earlier, before, later, after</td>
</tr>
<tr>
<td>Duration</td>
<td>Shorter, longer</td>
</tr>
<tr>
<td>Distance</td>
<td>Closer, nearer, farther, more distant</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sentence Purpose</th>
<th>Example Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declarative</td>
<td>I am looking for …</td>
</tr>
<tr>
<td>Imperative</td>
<td>Search for …</td>
</tr>
<tr>
<td>Interrogative</td>
<td>What is …</td>
</tr>
<tr>
<td>Exclamatory</td>
<td>—</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject Type</th>
<th>Interrogative Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>Who</td>
</tr>
<tr>
<td>Object</td>
<td>What</td>
</tr>
<tr>
<td>Time</td>
<td>When</td>
</tr>
<tr>
<td>Location</td>
<td>Where</td>
</tr>
</tbody>
</table>
NL: Connectives

cuisine == "Italian" + rating == 5

restaurant that serves Italian cuisine and was rated 5 stars
restaurant with rating 5 and Italian cuisine
5-star restaurant that serves Italian cuisine
Italian restaurant with 5 stars
5-star Italian restaurant
3. Property-Level Templates
Based on POS (Part-of-Speech)

<table>
<thead>
<tr>
<th>POS</th>
<th>Annotation</th>
<th>Example Template</th>
<th>Example utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is-a Noun</td>
<td>alumni of &lt;value&gt;</td>
<td><em>table</em> who are [noun phrase] <em>value</em></td>
<td>people who are alumni of Stanford</td>
</tr>
<tr>
<td>Has-a Noun</td>
<td>a &lt;value&gt; degree</td>
<td><em>table</em> with a <em>value</em> [noun phrase]</td>
<td>people with a Stanford degree</td>
</tr>
<tr>
<td>Active verb</td>
<td>graduated from &lt;value&gt;</td>
<td><em>table</em> who [verb phrase] <em>value</em></td>
<td>people who graduated from Stanford</td>
</tr>
<tr>
<td>Passive verb</td>
<td>educated at &lt;value&gt;</td>
<td><em>table</em> [passive verb phrase] <em>value</em></td>
<td>people who educated at Stanford</td>
</tr>
<tr>
<td>Adjective</td>
<td>&lt;value&gt;</td>
<td><em>value</em> <em>table</em></td>
<td>Stanford people</td>
</tr>
<tr>
<td>Prepositional</td>
<td>from &lt;value&gt;</td>
<td><em>table</em> [prepositional phrase] <em>value</em></td>
<td>people from Stanford</td>
</tr>
</tbody>
</table>
Synthesis

• Use grammars + values to generate many different sentences

• Goal:
  • A sampling of all queries [impossible to do all]
  • Teach the system compositionality with many combinations (made possible by synthesis)
  • Teach the system entities by augmenting queries with values from the databases (restaurant names, etc)
## Template Syntax

**Target**  
The target non-terminal for the template;  
\$root is the top-level non-terminal for a command

**Expansion:**  
a list of literals  
or non-terminals to compose the target

**Semantic function:**  
Build the ThingTalk abstract syntax tree of the target

<table>
<thead>
<tr>
<th>Nonterminals</th>
<th>Natural Language</th>
<th>ThingTalk abstract syntax tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>$root</td>
<td>“show me” $filtered_table</td>
<td>=&gt; return $filtered_table;</td>
</tr>
<tr>
<td>$filtered_table</td>
<td>$table “who” $verb_filter</td>
<td>=&gt; addFilter(table, $verb_filter)</td>
</tr>
</tbody>
</table>

**Example:**

- **Step 1:** \$filtered_table  
  - people who went to Stanford

- **Step 2:** \$root  
  - Show me people who went to Stanford
Compounding Templates: e.g. Multiple Filters

<table>
<thead>
<tr>
<th>filtered table (1)</th>
<th>filtered table (2)</th>
<th>filtered table (1) &amp; (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alumni of Stanford</td>
<td>employee of Apple</td>
<td>alumni of Stanford who are employee of Apple</td>
</tr>
<tr>
<td>people who are alumni of Stanford</td>
<td>Apple as their employer</td>
<td>alumni of Stanford who have Apple as their employer</td>
</tr>
<tr>
<td>people with a Stanford degree</td>
<td>works for Apple</td>
<td>alumni of Stanford who works for Apple</td>
</tr>
<tr>
<td>people who have a Stanford degree</td>
<td>employed by Apple</td>
<td>alumni of Stanford employed by Apple</td>
</tr>
<tr>
<td>people who graduated from Stanford</td>
<td></td>
<td>people who are alumni of Stanford and have Apple as their employer</td>
</tr>
<tr>
<td>people educated at Stanford</td>
<td></td>
<td>people who are alumni of Stanford and works for Apple</td>
</tr>
<tr>
<td>people who were educated at Stanford</td>
<td></td>
<td>people who are alumni of Stanford and are employed by Apple</td>
</tr>
<tr>
<td>Stanford people</td>
<td></td>
<td>employee of Apple with a Stanford degree</td>
</tr>
<tr>
<td>people from Stanford</td>
<td></td>
<td>People with a Stanford degree who have Apple as their employer</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>People with a Stanford degree that works for Apple</td>
</tr>
</tbody>
</table>

Semantic functions (ThingTalk) are composed correspondingly.
Template to Add <search> To Get Full Questions

<table>
<thead>
<tr>
<th>filtered table</th>
</tr>
</thead>
<tbody>
<tr>
<td>alumni of Stanford</td>
</tr>
<tr>
<td>people who are alumni of Stanford</td>
</tr>
<tr>
<td>people with a Stanford degree</td>
</tr>
<tr>
<td>people who have a Stanford degree</td>
</tr>
<tr>
<td>people who graduated from Stanford</td>
</tr>
<tr>
<td>people educated at Stanford</td>
</tr>
<tr>
<td>people who were educated at Stanford</td>
</tr>
<tr>
<td>Stanford people</td>
</tr>
<tr>
<td>people from Stanford</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<generic verb for search> <filtered table>

Semantic functions (ThingTalk) are composed correspondingly
Neural Semantic Parser Model

• Pre-trained BERT encoder
• LSTM decoder

(Experiment done before Bart was available)

Schema2QA: High-Quality and Low-Cost Q&A Agents for the Structured Web
Silei Xu, Giovanni Campagna, Jian Li, and Monica S. Lam
In Proceedings of the 29th ACM International Conference on Information and Knowledge Management, October 2020
Schema2QA [Xu 2020a]: Evaluate on Realistic Data

- Schema from schema.org: Restaurant, people, movies, books, music
- Values from: Yelp, linkedIn, IMDB, Goodreads, Last.fm
- Evaluation/Test: much more realistic user input [Not paraphrases]
  - Crowdsourcing workers shown the properties of the schema
  - Over 2/3 of the questions have at least 2 properties in them
- Open ontology: Contains values unseen in training

- questions
- annotate
- restaurant

name
cuisine
reviews
...
## Dataset Sizes

<table>
<thead>
<tr>
<th></th>
<th>Restaurant</th>
<th>People</th>
<th>Movie</th>
<th>Book</th>
<th>Music</th>
<th>Hotel</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># Properties</strong></td>
<td>25</td>
<td>13</td>
<td>16</td>
<td>15</td>
<td>19</td>
<td>18</td>
<td>17.7</td>
</tr>
<tr>
<td><strong>Human Annotations</strong></td>
<td>122</td>
<td>95</td>
<td>111</td>
<td>96</td>
<td>103</td>
<td>83</td>
<td>101.7</td>
</tr>
<tr>
<td><strong>Synthetic</strong></td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
</tr>
<tr>
<td><strong>Human Paraphrase</strong></td>
<td>6.4K</td>
<td>7.1K</td>
<td>3.8K</td>
<td>3.9K</td>
<td>3.6K</td>
<td>3.3K</td>
<td>4.7K</td>
</tr>
<tr>
<td><strong>Dev</strong></td>
<td>528</td>
<td>499</td>
<td>389</td>
<td>362</td>
<td>326</td>
<td>433</td>
<td>424.5</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td>524</td>
<td>500</td>
<td>413</td>
<td>410</td>
<td>288</td>
<td>528</td>
<td>443.8</td>
</tr>
</tbody>
</table>
Evaluation Result

- Restaurants
- People
- Movies
- Books
- Music
- Hotels
- Average

Baseline: Templates only (No manual annotations) vs. Templates + manual annotation & paraphrases

- Restaurants
- People
- Movies
- Books
- Music
- Hotels
- Average
## Comparison with SEMPRE

<table>
<thead>
<tr>
<th>SEMPRE (Overnight paper)</th>
<th>Genie Schema2QA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Manual:</strong> Annotate properties (same POS)</td>
<td><strong>Manual:</strong> Annotate properties (different POS)</td>
</tr>
<tr>
<td><strong>Automatic:</strong> Grammar-based synthesis</td>
<td><strong>Automatic:</strong> Grammar-based synthesis</td>
</tr>
<tr>
<td>(canonical only)</td>
<td>(with 900 templates)</td>
</tr>
<tr>
<td><strong>Manual:</strong> Paraphrase synthesized sentences</td>
<td><strong>Manual:</strong> Paraphrase 2% of synthesized sentences</td>
</tr>
<tr>
<td><strong>Train with only paraphrased sentences</strong></td>
<td><strong>Train with synthesized + few-shot paraphrased data</strong></td>
</tr>
</tbody>
</table>
Results: Comparison with SEMPRE

Evaluated and tested on crowdsourced questions (Not paraphrases)

Table 7: Query accuracy of BERT-LSTM trained with only data synthesized from SEMPRE and Schema2QA templates.

<table>
<thead>
<tr>
<th></th>
<th>Restaurants</th>
<th>People</th>
<th>Movies</th>
<th>Books</th>
<th>Music</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEMPRE</td>
<td>3.5%</td>
<td>1.6%</td>
<td>4.8%</td>
<td>2.2%</td>
<td>0.7%</td>
<td>2.6%</td>
</tr>
<tr>
<td>Schema2QA</td>
<td>63.2%</td>
<td>66.8%</td>
<td>63.2%</td>
<td>49.8%</td>
<td>57.3%</td>
<td>60.1%</td>
</tr>
</tbody>
</table>

Manual annotations of fields included
No paraphrases
Quiz

• Template-based generation: 900 templates
  • Is it worth the work?
  • Do we need to repeat for every language?
Outline

• SEMPRE: Seminal paper on synthesis-based training data acquisition
  • Synthesize 1 canonical sentence per query
  • Train with only manually paraphrased data on paraphrase data
• Schema2QA: Manual annotations / paraphrases
  • Synthesize with 900 templates
  • Train with synthesized & manually paraphrased data on real data
• AUTOQA – Automatically synthesized + a few shot
  • Synthesize with 900 templates
  • Automatic annotation and paraphrases
  • Train with synthesized + auto-paraphrase data + few-shot
Can We Fully Automate the Process

• Automatic field annotation?

• Automatic paraphrases?
Auto-Annator

*alumniOf* property in *people* table

<table>
<thead>
<tr>
<th>POS</th>
<th>Annotation</th>
<th>Example Template</th>
<th>Example utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is-a Noun</td>
<td>alumni of &lt;value&gt;</td>
<td><em>table</em> who are [noun phrase] <em>value</em></td>
<td>people who are alumni of Stanford</td>
</tr>
<tr>
<td>Has-a Noun</td>
<td>a &lt;value&gt; degree</td>
<td><em>table</em> with a <em>value</em> [noun phrase]</td>
<td>people with a Stanford degree</td>
</tr>
<tr>
<td>Active verb</td>
<td>graduated from &lt;value&gt;</td>
<td><em>table</em> who [verb phrase] <em>value</em></td>
<td>people who graduated from Stanford</td>
</tr>
<tr>
<td>Passive verb</td>
<td>educated at &lt;value&gt;</td>
<td><em>table</em> [passive verb phrase] <em>value</em></td>
<td>people who educated at Stanford</td>
</tr>
<tr>
<td>Adjective</td>
<td>&lt;value&gt;</td>
<td><em>value</em> <em>table</em></td>
<td>Stanford people</td>
</tr>
<tr>
<td>Prepositional</td>
<td>from &lt;value&gt;</td>
<td><em>table</em> [prepositional phrase] <em>value</em></td>
<td>people from Stanford</td>
</tr>
</tbody>
</table>

1. Generate canonical annotation based on name, and assign type by POS Tagger
2. Construct simple example sentences with templates
3. Paraphrase with a neural paraphrase model
4. Parse the paraphrases with POS-based parser to extract annotations
Auto-Annotator

- Use simple sentences: one property at a time
  - Less mistakes
  - Focus on the property to obtain more variety
- Always use real-world values
  - More context to the language model

Quiz: Why annotate at the POS level, and not just the whole sentence?
## Training

<table>
<thead>
<tr>
<th></th>
<th>Restaurant</th>
<th>People</th>
<th>Movie</th>
<th>Book</th>
<th>Music</th>
<th>Hotel</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># of Properties</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>13</td>
<td>16</td>
<td>15</td>
<td>19</td>
<td>18</td>
<td>17.7</td>
</tr>
<tr>
<td><strong>Schema2QA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manual Annotations</td>
<td>122</td>
<td>95</td>
<td>111</td>
<td>96</td>
<td>103</td>
<td>83</td>
<td>101.7</td>
</tr>
<tr>
<td>Synthetic</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
</tr>
<tr>
<td>Human Paraphrase</td>
<td>6.4K</td>
<td>7.1K</td>
<td>3.8K</td>
<td>3.9K</td>
<td>3.6K</td>
<td>3.3K</td>
<td>4.7K</td>
</tr>
<tr>
<td><strong>AutoQA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto Annotations</td>
<td>151</td>
<td>121</td>
<td>157</td>
<td>150</td>
<td>144</td>
<td>160</td>
<td>147.2</td>
</tr>
<tr>
<td>Synthetic</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
</tr>
</tbody>
</table>
Evaluation on Schema2QA Dataset

- Restaurants
- People
- Movies
- Books
- Music
- Hotels
- Average

- Templates Only
- With Auto-Annotator
- With Manual Annotations & Paraphrases
Evaluation on Schema2QA Dataset

Accuracy goes up by ~19%
Evaluation on Schema2QA Dataset

- Restaurants
- People
- Movies
- Books
- Music
- Hotels
- Average

- Templates Only
- With Auto-Annotator
- With Manual Annotations & Paraphrases
Automatic Training Data Generation

• Automatic annotators

• Automatic paraphrases
How to Auto-Paraphrase

• The task: Find paraphrases to the NL part of the training data
  • Example: NL: Could you find me a restaurant that serves American food?,
    TT: restaurant, cuisine = ‘American’
  • A paraphraser: an NL sentence \(\rightarrow\) an equivalent NL sentence
  • To get a paraphraser:
    Fine-tune a Seq2Seq model on a paraphrasing dataset
    • how hard can it be?
    • Quiz: What make good paraphrases for our goal?
Outline

1. The paraphrasing problem? How to evaluate?
2. How to get a paraphrase data set to train
3. How to ensure correctness and diversity
1. Paraphrase Problem

- A paraphrasing dataset
  - sentence pairs \((X_i, Y_i)\) where \(X_i\) and \(Y_i\) are paraphrases of each other
- What is the loss function in training?
  - Predict the next token in the gold sentence
  - Negative log likelihood
- What is the metric for evaluation?
BLEU Score: a Validation Metric

- **Bilingual Evaluation Understudy (2002)**
  - Compares machine generations to one or several human-written references
  - Computes a similarity score by matching n-grams of the generated text against the references:

\[
BLEU = \beta \prod_{n=1}^{k} p_n^{2^n}
\]

- \(\beta\) is a function of the length of the generated text, to penalize short ones
- \(n\)-grams: Overlapping spans of \(n\)-words
- \(p_n\) is **\(n\)-gram precision**: (# matched \(n\)-grams) / (# \(n\)-grams in generated text)
  - a gram in the reference can be matched only once
- \(k\) is usually 4

Original Text: Could you find me a restaurant that serves American food?
Paraphrase 1: Please suggest an American diner.
Paraphrase 2: Could you find me a hotel?
2. Getting a Paraphrase Dataset

1. Two sentences are paraphrases if their representations are similar, according to some model (Lewis et al, 2020)

2. Two sentences that describe the same picture are paraphrases (Prakash et al, 2016)

MSCOCO dataset

- a dog makes a face while rolling on the ground.
- a brown and white dog laying on his back smiling.
- a dog is on it’s back on the grass with an open mouth.
- a dog laying on its back in the grass with its mouth open.
- a dog with its mouth open lays in the grass.
Getting a Paraphrase Dataset

3. Translations from the same sentence
   The ParaBank 2 Dataset (Hu et al, 2019)
   - Example
     French: L'homme est né libre, et partout il est dans les fers, Rousseau
     - Google Translations
       - Man is born free, but everywhere he is in chains
       - Man was born free, and everywhere he is in chains
     - Human Translation
       - Man is born free, and everywhere he is in shackles
       - People are born free, but they are in iron chains throughout
The ParaBank 2 Dataset

- Created using a bilingual (English-Czech) corpus of news, books, movie subtitles, etc.
- Given [English, Czech], English’ = translate to English (Czech)
  - English and English’ are paraphrases
- Lots of tricks to improve the grammaticality and diversity of machine translated sentences
- Scores ~90% on grammaticality
- 84% on semantic similarity of pairs, according to human judges
Our Paraphraser

• Fine-tune a BART pretrained seq2seq model with the ParaBank 2 dataset
BART Pre-trained Seq2Seq Model

Bert: Masked model, not generational

GPT: Next word prediction, not bidirectional

BART: Seq-2-seq, Denoising model, bidirectional, generational

Graphic courtesy of: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
Bart: Pretrained Model

• Denoising
  • Token masking
  • Token deletion
  • Text span masking: replace a span with one mask token
  • Sentence permutation
  • Document rotation: starting at a random position
• Model
  • 6-layer transformer each for encode and decode
  • BART-large: 12 layers
• Useful for downstream tasks
  • Question answering, entailment, summarization, response generation, translation
3. Diverse and accurate paraphrases

<table>
<thead>
<tr>
<th>Synthetic</th>
<th>Paraphrased</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search some cafeteria that have greater star</td>
<td>Search for a restaurant that has more than 3</td>
</tr>
<tr>
<td>than 3, and do not have smoking.</td>
<td>stars and doesn't smoke.</td>
</tr>
</tbody>
</table>

BART-based paraphraser
Self-Training

We need to filter out the noisy paraphrases

1. Train a parser with synthetic dataset
2. Generate potentially noisy paraphrases of the synthetic dataset
3. Use the parser from (1) to parse the paraphrases of (2)
4. Remove all paraphrases where the new parse does not match the label
5. Add filtered paraphrases to the training set
6. Repeat

![Diagram showing the process of self-training with an auto-paraphraser, semantic parser, and paraphrase filter. The synthetic dataset flows through the components, with arrows indicating the direction of data processing.]
Genie Summary

- Database schema and values
  - Auto-annotator
    - Paraphraser
    - POS-based annotation extraction
    - Attribute annotations
  - Template-based data synthesizer
    - English-grammar-based comprehensive templates
    - Auto-paraphraser
      - Paraphraser
      - Semantic parser
        - Paraphrases
        - Logical forms
      - Paraphrase filter
        - Paraphrases + original logical forms

Stanford University
Quiz

• Why bother with self-training if we only accept paraphrases that are already parsed correctly?

• Do we need to filter noise on property-level paraphrases?

• Can we skip property-level paraphrases?
Outline

1. Why is natural language (English) so hard?
2. Idea 1: Domain-Independent Templates
3. Idea 2: Automatic Property-Level Annotation
4. Neural Paraphrasing
5. Idea 3: Automatic Sentence-Level Paraphraser
6. Evaluation
### Schema2QA: Training Set

<table>
<thead>
<tr>
<th></th>
<th>Restaurant</th>
<th>People</th>
<th>Movie</th>
<th>Book</th>
<th>Music</th>
<th>Hotel</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># of Properties</strong></td>
<td>25</td>
<td>13</td>
<td>16</td>
<td>15</td>
<td>19</td>
<td>18</td>
<td>17.7</td>
</tr>
<tr>
<td><strong>Schema2QA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Synthetic</strong></td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
</tr>
<tr>
<td><strong>Human Paraphrase</strong></td>
<td>6.4K</td>
<td>7.1K</td>
<td>3.8K</td>
<td>3.9K</td>
<td>3.6K</td>
<td>3.3K</td>
<td>4.7K</td>
</tr>
<tr>
<td><strong>AutoQA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong># of Annotations</strong></td>
<td>151</td>
<td>121</td>
<td>157</td>
<td>150</td>
<td>144</td>
<td>160</td>
<td>147.2</td>
</tr>
<tr>
<td><strong>Synthetic</strong></td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
<td>270K</td>
</tr>
<tr>
<td><strong>Auto paraphrase</strong></td>
<td>281K</td>
<td>299K</td>
<td>331K</td>
<td>212K</td>
<td>341K</td>
<td>285K</td>
<td>292K</td>
</tr>
</tbody>
</table>

- Training takes about 3 hours using V100 for 30K iterations
Evaluation Result

- Restaurants: 20%
- People: 20%
- Movies: 40%
- Books: 60%
- Music: 80%
- Hotels: 100%
- Average: 73.33%
Evaluation Result

Accuracy goes up by ~19%
Evaluation Result

Accuracy goes down by ~10%!
Evaluation Result

- Templates Only
- With Auto-Annotator
- With Auto-Annotator + Naive Paraphraser
- With Auto-Annotator + Auto-Paraphraser

Accuracy goes up by ~8%
Evaluation Result

There is a ~6% gap.
Change the BERT-LSTM to Fine-Tuning BART

Auto-BART matches performance with human data on BERT-LSTM
Auto-Annotator & Auto-Paraphraser are Complementary

- Auto-annotator
  - phrase-level
  - generic

- Auto-paraphraser
  - sentence-level
  - value-specific

Restaurants  People  Movies  Books  Music  Hotels  Average

Stanford University
Quiz

- Now we can automate everything, should we generate as much data as possible?
Quiz

• Now we can automate everything, how much data should we generate?
  • Accuracy grows logarithmically with the amount of data
  • [Oren et al 2021] On Schema2QA dataset, a carefully sampled dataset with 5K examples can achieve comparable accuracy (83.4%) with a model trained with 1M examples (85%)
  • Find the sweet point that balances accuracy and computation cost!
Quiz

• Is the performance good enough?

• How do we improve the performance?
Conclusions

• Importance to test with real data
• Fully automatic tool: Schema → Question semantic parser
• Data synthesis
  1. Property-level paraphrases to extract POS
  2. Domain-independent templates (900)
  3. Sentence-level paraphrases with noise-filtering with self-training
     • Paraphraser: Fine-tuned on BART with the ParaBank 2 Dataset
     • Self-Training: Use model $i$ to label more data to train model $i+1$
References

• [Wang 2015] Building a Semantic Parser Overnight
• [Su 2017]: Cross-domain Semantic Parsing via Paraphrasing
• [Xu 2020a] Schema2QA: High-Quality and Low-Cost Q&A Agents for the Structured Web
• [Xu 2020b] AutoQA: From Databases To Q&A Semantic Parsers With Only Synthetic Training Data
• [Marion 2021] Structured Context and High-Coverage Grammar for Conversational Question Answering over Knowledge Graphs