DBPal: A Learned Natural Language Interface for Databases using Distant Supervision

P. Ajie Utama, Nathaniel Weir, Carsten Binnig, Ugur Cetintemel
Brown University, TU Darmstadt
Motivation

“who is the flu patient with highest age”

SELECT *
FROM patients
WHERE patients.age = (
    SELECT MAX( patients.age )
    FROM patients
    WHERE patients.diagnosis = "flu"
)
AND patients.diagnosis = "flu"
LIMIT 1

NL Interfaces to Databases (NLIDBs) enable a natural and concise way for naive users to explore data
NLIDB as Machine Translation

- Sequence-to-Sequence (seq2seq) neural model has shown remarkable success on
  - Language Translation, e.g. English to French, English to Czech
  - Semantic Parsing, e.g. English to λ-Calculus
Challenges:

● Training set bottleneck: Annotation is expensive!
  ○ Crowdsourcing is imperfect and time-consuming
  ○ Need new data for every new domain

● Natural language is infinitely more expressive than SQL
  ○ Many NL utterances map to a single SQL query, need to be able to handle them all

<table>
<thead>
<tr>
<th>Natural Language</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>what is the count of patients where diagnosis is flu ?</td>
<td>SQL: SELECT COUNT(*) FROM patients WHERE diagnosis='flu'</td>
</tr>
<tr>
<td>what is the total sum of patients where diagnosis is being influenza ?</td>
<td>what is the count of patients diagnosed with flu ?</td>
</tr>
<tr>
<td>how many patients with flu are there ?</td>
<td>count flu-diagnosed patients</td>
</tr>
</tbody>
</table>
Goal

Create a domain-agnostic, linguistically robust neural translation model that a naive user can plug into any new database schema
Our Approach: Synthetic Data Generation

**Input**
- DB Schema

**Generate using templates**
- Cover variety of SQL

**Automatic paraphrasing and noising**
- Cover variety of NL

**Output**
- Augmented NL / SQL pairs

**Distant Supervision:** Automatically generating and labelling (potentially noisy) training data rather than manually handcrafting it.
Template-based Generation

### Example: Patients Data

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carsten</td>
<td>39</td>
<td>Flu</td>
</tr>
<tr>
<td>Melanie</td>
<td>28</td>
<td>Diabetes</td>
</tr>
<tr>
<td>Fred</td>
<td>14</td>
<td>Flu</td>
</tr>
</tbody>
</table>

#### SQL:

```
SELECT <att>
FROM <table>
WHERE <conds>
```

#### NL:

Show me the <att>(s) of <table> with <conds>

#### Slot Filling:

```
SELECT age
FROM patients
WHERE diagnosis = 'flu'
```

Show me the age of patients with diagnosis equal to flu

#### Automatic Paraphrasing:

Display the age of inpatients where diagnosis is flu

#### Word Dropping:

Show me the age of patients with diagnosis equal to flu

---

Millions of NL/SQL Pairs Produced
Framework Summary:

**Schema**
- SQL / NL template pairs
- Slotfilling Lexicon

**Generator**
- "What are cities whose state is California"
  - SELECT name FROM cities WHERE state = 'California'
- "Show me average of population of cities for each state"
  - SELECT AVG(population) FROM cities GROUP BY state

**Augmentations**

**Lemmatizer**

**Inference time**
- NL query
- Seq2Seq
- SQL Queries

**Templatizer**

**RDMBS**
Experimental Evaluation:

Evaluated Systems

- NaLIR: Rule-based NLIDB (Li & Jagadish, Best Paper VLDB 2015)
  - Supervised learning; requires manual annotation for new database schema
- DBPal: Our Framework (Utama et al. 2018)
  - Distant supervision; minimal manual effort for new database schema

Accuracy comparison between DBPal and other baselines on the two benchmark datasets

<table>
<thead>
<tr>
<th></th>
<th>Patients</th>
<th>GeoQuery</th>
</tr>
</thead>
<tbody>
<tr>
<td>NaLIR (w/o feedback)</td>
<td>15.60%</td>
<td>7.14%</td>
</tr>
<tr>
<td>NaLIR (w feedback)</td>
<td>21.42%</td>
<td>N/A</td>
</tr>
<tr>
<td>NSP</td>
<td>N/A</td>
<td>83.9%</td>
</tr>
<tr>
<td>DBPal (w/o augmentation)</td>
<td>74.80%</td>
<td>38.60%</td>
</tr>
<tr>
<td>DBPal (full pipeline)</td>
<td>75.93%</td>
<td>55.40%</td>
</tr>
</tbody>
</table>
Ongoing Work

- Dialogue-based query refinement
- Complex query identification and handling
- Handling unseen words and phrases more effectively
- Leverage POS tagging/syntactic parsing as auxiliary task to increase compositionality
DBPal in Action:

- Real time translation system user interface:

```sql
SELECT patients.diagnosis FROM patients WHERE patients.gender = 'female' AND patients.age = 50
```

http://localhost:8888/#/patients
Model Architecture

Encoder

Decoder

GRU
GRU
... GRU
GRU
GRU
... GRU
GRU
GRU

Layer 2

Layer 1

<s>
SELECT city.name INTEGER@0

show
cities
INTEGER@0

=