Say the word, and You'll be Free: Methods and Techniques for Natural Language Database Interfaces

Altigran da Silva
BDRI@UFAM
Manaus/Brazil

DOING@MaDICS
5 July 2021

Topics

• NLIDB: Motivations, Challenges, Limitations, Demands and Opportunities
• Visions: Data versus Language
• Data-Centric Systems (DCS):
  • NALIR, TEMPLAR, ATHENA and ATHENA++
• Language-Centric Systems (LCS)
  • Background: Seq2Seq Models and Embeddings
  • Seq2SQL and DBPal
• Queries over Multiple Tables
• Conclusions, Remarks, Developments and References
• Hands-on: implementation of an NLIDB in Python

Formal Query Languages are still Hard

• Non-technical and casual users are overwhelmed by technical issues from formal query languages
• SQL was initially developed for executive people to use
• Reality: even trained users face problems to write correct queries [Bowen@WITS’04]
• Users must be aware of:
  • The details on the schema of each DB
  • The semantics of each DB element mentioned in the query
  • The ways for joining information in the DB
  • The syntax of the query language

Natural Language Interfaces for DBs

• Allow casual users to access information stored in DBs using queries expressed in natural language
• The “philosopher’s stone” of DB interfaces [Codd@IFIP’1974]

“...the only way to encourage the casual user to interact with a database system is to allow free use of their native language.”

Edgar Frank “Ted” Codd
Creator of the Relational Model
Example [Affolter@VLDBJ’19]

Show me all movies with the actor Brad Pitt.

What are the movies with the actor Brad Pitt?

```
SELECT m.title
FROM Movie m JOIN Starring s ON s.movieId = m.id
JOIN Actor a ON a.actorId = s.actorId
JOIN Person p ON p.id = a.actorId
WHERE p.FirstName = "Brad" AND p.LastName = "Pitt"
```

NL Interfaces for DBs – Challenges

• Requirements
  • Understand the user’s intention or information needs when formulating the query.
  • Correctly represent this intention in a structured query language
  • Ultimately, it would imply in solving the general problem of Natural Language Understanding (NLU)
    • NLU is hypothetically one of the AI-Hard problems
  • Thus, all current systems are necessarily approximated and limited, considering the Codd’s statement

NL Interfaces for DBs – Limitations

• All existing methods use general pre-processing techniques
• Each one is based on assumptions on the NL queries they support
• Many times, these hypothesis are not explicit
• Examples of techniques/resources
  • Rules
  • Lexicons, ontologies, dictionaries, etc.
  • Training
  • User interaction
  • Logs

NL Interfaces for DBs – Evaluation

• There is no full understanding of how good techniques really are
• It is unknown how applicable they would be to real world situations
• Different studies, based on different datasets
• Often have limitations and assumptions, implicitly hidden in the context or datasets.
• Some evaluation metrics are commonly used, but they are quite simplistic and do not adequately represent the quality of results.
**NL Interfaces for DBs – Evaluation**

- Benchmark queries for NLDB qualitative evaluation [Affolter@VLDBJ’19]

<table>
<thead>
<tr>
<th>NL Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
</tr>
<tr>
<td>Q4</td>
</tr>
<tr>
<td>Q5</td>
</tr>
<tr>
<td>Q6</td>
</tr>
<tr>
<td>Q7</td>
</tr>
<tr>
<td>Q8</td>
</tr>
<tr>
<td>Q9</td>
</tr>
<tr>
<td>Q10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Likely Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join + String-based Selection</td>
</tr>
<tr>
<td>Join + Range-based Selection</td>
</tr>
<tr>
<td>Join + Date-based Selection</td>
</tr>
<tr>
<td>Join + Aggregation/Ordering</td>
</tr>
<tr>
<td>Join + Union</td>
</tr>
<tr>
<td>Concept/Subjectivity</td>
</tr>
<tr>
<td>Join + Aggregation</td>
</tr>
<tr>
<td>Join + Negation-based Selection</td>
</tr>
<tr>
<td>Join + Subquery</td>
</tr>
<tr>
<td>Join + Subquery</td>
</tr>
</tbody>
</table>

**Demand and Opportunities**

- **Demands**
  - Popularization of IR Systems - Search Engines: Users become used to explore by themselves
  - Data Scientist, Data Journalists
  - Democratization of access to online DBs for casual users
  - Massive use of conversational interfaces

- **Opportunities:**
  - Technical maturity in NLP, ML & IR allow extracting text semantics with precision and efficiency

**Increasing Interest – Citations per Year**

- **LUNAR - 1973**
  - Progress in natural language understanding: an application to lunar geology [Woods@AFIPS’1973]

- **Precise - 2004**
  - Modern Natural Language Interfaces to Databases [Popescu@COLING94]

- **NALIR – 2014**
  - Constructing an Interactive Natural Language Interface for Relational Databases [Li@PVLDB’14]

- **NALIR – 2014 Best Paper VLDB**

**Increasing Interest – DB Community**

- **ICDE 2020:** 4 papers
- **SIGMOD 2020:** 5 papers, one tutorial
- **VLDB 2020:** 3 papers, one tutorial
- **ICDE 2021:** 4 papers
- **SIGMOD 2021:** 3 papers
- **VLDB 2021:** 1 paper so far
Visions: Data versus Language

Data-Centric Systems (DCS):
- Focus: Map references to DB elements occurring in the query
- Use rule-based techniques to map NL query words to SQL clauses
- Less dependent on the DB; More dependent on variations in NL queries

Language-Centric Systems (LCS):
- Focus: Instance of the automatic language translation problem
- Use Deep Learning models and algorithms
- More dependent on the DB; Less dependent on variations in NL queries

<table>
<thead>
<tr>
<th>Data-Centric</th>
<th>Language-Centric</th>
</tr>
</thead>
<tbody>
<tr>
<td>NaLIR</td>
<td>[Li@PVLDB'14]</td>
</tr>
<tr>
<td>SQLizer</td>
<td>[Yaghmazadeh,OOPSLA'17]</td>
</tr>
<tr>
<td>Templar</td>
<td>[Baik@ICDE'19]</td>
</tr>
<tr>
<td>ATHENA</td>
<td>[Saha@PVLDB'16]</td>
</tr>
<tr>
<td>ATHENA++</td>
<td>[Sen@PVLDB'20]</td>
</tr>
<tr>
<td>NSP</td>
<td>[Iyer@ACL'17]</td>
</tr>
<tr>
<td>Seq2SQL</td>
<td>[Zhong@CoRR'17]</td>
</tr>
<tr>
<td>SQLNet</td>
<td>[Xu@CoRR'17]</td>
</tr>
<tr>
<td>Coarse2Fine</td>
<td>[Lapata@ACL'18]</td>
</tr>
<tr>
<td>STAMP</td>
<td>[Zhou@ACL'18]</td>
</tr>
<tr>
<td>PT-MAML</td>
<td>[Huang@NAACL-HLT'18]</td>
</tr>
<tr>
<td>TypeSQL</td>
<td>[Yu@NAACL-HLT'18]</td>
</tr>
<tr>
<td>SyntaxSQLNet</td>
<td>[Yu@EMNLP'18]</td>
</tr>
<tr>
<td>GNN</td>
<td>[Guo@ACL'19]</td>
</tr>
<tr>
<td>IRNet</td>
<td>[Bogin@ACL'19]</td>
</tr>
<tr>
<td>Photon</td>
<td>[Zeng@CoRR'20]</td>
</tr>
<tr>
<td>DBPal</td>
<td>[Weir@SIGMOD'20]</td>
</tr>
</tbody>
</table>
NaLIR [Li@PVLDB’14]

- Natural Language Interface for Relational databases
- F. Li and H. V. Jagadish – DBGroup University of Michigan
- Best Paper VLDB, 2014
- Often used as a baseline in evaluation experiments with other DCS
- Original code and datasets: https://github.com/umich-dbgroup/NaLIR
- Python implementation by our group: http://fy.lu/cM9y
  - Jupyter notebook prepared by Genoveva and Javier Espinosa, thanks!

NaLIR [Li@PVLDB’14] – Parsing

- Dependency parsing: task of finding syntactic dependencies in a sentence.
- Syntactic dependencies: asymmetric binary relationship between words
  - Includes grammatical roles (subject, object, determinative, modifier)
- Results in a syntactic dependency tree
- Uses the well-know Stanford Parser [Marneffe@LREC’06]
NaLIR [Li@PVLDB’14] – Parsing

- Given the above two observations, instead of explaining the linguistic parse tree from the database’s meta-data and data, respectively, which entirely depend on the database being queried.
- Often, the words/phrases they mapped to. The identification of select node, operator node, name node, and value nodes correspond to different tokens.
- In the mapping process, some nodes may fail to be mapped. In contrast, name nodes and value nodes correspond to the meta-data and data, respectively.
- The translation of the parses into SQL statements is straightforward since the structure of the EDM is identical to that of the SQL language. The translation is almost always exact. Sometimes there are multiple candidate valid parse trees for the query.

NaLIR [Li@PVLDB’14] – Node Mapper

- Identifies nodes that can be mapped to SQL components.
- Uses a table manually constructed that maps NL phrases to SQL clauses.
- Problems:
  - Some nodes are not mapped.
  - Some nodes have multiple mappings.

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Corresponding SQL Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Node (SN)</td>
<td>SQL keyword: SELECT</td>
</tr>
<tr>
<td>operator Node (ON)</td>
<td>an operator, e.g., &lt;, &gt;, =, contains</td>
</tr>
<tr>
<td>Function Node (FN)</td>
<td>an aggregation function, e.g., AVG</td>
</tr>
<tr>
<td>Name Node (NN)</td>
<td>a relation name or attribute name</td>
</tr>
<tr>
<td>Value Node (VN)</td>
<td>a value under an attribute</td>
</tr>
<tr>
<td>Quantifier Node (QN)</td>
<td>ALL, ANY, EACH</td>
</tr>
<tr>
<td>Logic Node (LN)</td>
<td>AND, OR, NOT</td>
</tr>
</tbody>
</table>

NaLIR [Li@PVLDB’14] – Tree Adjustor

- Analyzes the dependency tree with the mapped nodes.
- Generates Query Trees – candidate interpretations of the NL query.
- It can also adjust candidate query trees to make them syntactically valid considering the SQL language.
- Ranks the candidate query trees.
- The "best" of them leads to the SQL query.
- Adjustment and ranking based on a series of fixed heuristics.
NaLIR [Li@PVLDB’14] – Tree Adjustor

NaLIR [Li@PVLDB’14] – User Interaction

• Several problems can arise in the process
  • Parsing can generate spurious nodes from the query point of view
  • Mapping can fail or be ambiguous
  • Tree adjusting and ranking may fail
• In all these cases, the user is called to intervene
  • Perform adjustments and changes manually.

Experiments with the Microsoft Academic Search DB

<table>
<thead>
<tr>
<th>Queries</th>
<th>With User Interaction</th>
<th>No User Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td>34/34</td>
<td>24/32</td>
</tr>
<tr>
<td>Medium</td>
<td>34/34</td>
<td>23/34</td>
</tr>
<tr>
<td>Hard</td>
<td>20/30</td>
<td>15/32</td>
</tr>
</tbody>
</table>

NaLIR [Li@PVLDB’14] – User Interaction

Templar [Baik@ICDE’19]

• From the same group that created NaLIR
• Attempt to decrease user dependency
• Proposes relying on information mined from a query log
• Uses optimization techniques to improve:
  • Mapping of words in the NL query to DB elements
  • Join path generation
We provide a novel ontology-driven algorithm that generates a ranked list of interpretations, and their corresponding OQL queries. Moreover, ATHENA attains 87.2%, 88.3%, and 88.9% respective experimental study on three workloads. ATHENA achieves 100% precision on a geographical (GEO) and an academic (ACAD) dataset.

ATHENA [Saha@PVLDB’16] – Fase 1

- Phase 1: NL query Interpretation
- Maps each query word to ontology elements to which it may refer
  - Ex: “Mirian” mapped to Article.Author and Event.Coordinator
- Mapping combinations yield various interpretations of the LN query
- Each combination corresponds to a tree in the ontology graph
  - Interpretation Trees or iTree
- Finding these trees is a variation of the Steiner Tree Problem
  - An NP-Complete problem

ATHENA [Saha@PVLDB’16] – Example

Show me restricted stock investments in Alibaba since 2012 by investor and year

ATHENA [Saha@PVLDB’16] – Example (2)

Show me restricted stock investments in Alibaba since 2012 by investor and year
inner and outer query blocks. As shown in Figure 3, for query inner queries, and hence decides that

```
FROM
```

illustrate the workflow of A

Show me restricted stock investments in Alibaba since 2012 by investor and year

ATHENA [Saha@PVLDB’16] – Example (3)

ATHENA [Saha@PVLDB’16] – Fase 2

- Phase 2: Structured Query Generation
  - Relies on an Ontology-to-Database Mapping (MDG)
  - Describes how elements of the ontology are mapped to the DB elements
    - (concepts, property, relationships) => (tables, views, columns, FKs)
  - The "best" iTrees transformed into queries according to the MDS
  - Ontology Query Language (OQL):
    - Intermediate language used to allow logical independence
    - Query Translator: OQL to SQL
      - Other QLs can be used
  - ATHENA provides a ranked list of the queries
    - The user can choose the most appropriate

ATHENA++ [Sen@PVLDB’20]

- Extends ATHENA to cover complex nested queries
- The original query is partitioned into possible nested queries, according to a predefined taxonomy.

Language-Centric Systems (LCS)
Language-Centric Systems (LCS)

- Emerged mainly from the NLP community
- Main advantage: rely on machine learning instead of fixed rules
  - E.g., trained to recognize: “major cities” ⇒ “city.population > 150,000”
- Explore state-of-the-art Deep Learning techniques
- Specifically: sequence conversion method - Seq2Seq
- Challenge: Training
  - Needed for each target database
  - May involve queries and instances
  - Sometimes costly and error-prone

Sequence-to-Sequence Models (Seq2Seq)

- A Neural Network Model
- Transforms a sequence of elements into another sequence of elements
- Includes two networks: an Encoder (COD) and a Decoder (DEC)
  - COD: takes an input sequence and maps to an n-dimensional vector
  - DEC: takes the vector and transforms it into an output sequence.
- Most typical application example is machine translation

Coder (COD) and Decoder (DEC)

- Imagine COD and DEC as translators, each one speaking two languages.
- The first language is the mother tongue, which differs between the two
- For example, Portuguese and French
- The second is an imaginary language that the two speak
  - This correspond to the n-dimensional vector
- To translate Portuguese into French
  - The COD converts a Portuguese phrase into the imaginary language
  - As the DEC is able to read the imaginary language, it can translate the phrase into French.

Coder (COD) and Decoder (DEC) (2)

- Suppose that, initially, neither COD nor DEC are very fluent in the imaginary language.
- So that they can learn, we train them with several examples
  - This corresponds to the model training
- Usually implemented with Recurrent Neural Networks (RNN)
- Alternatives: LSTMs, Bi-LSTMs, GRU, transformers, ...
- Stacked nets can be used.
- Top-layer output states are the final representation
10.2 Encoder-Decoder Networks

Abstracts away from the specifics of machine translation and illustrates a basic encoder-decoder architecture. The elements of the network on the left process the input sequence and comprise the encoder, the entire purpose of which is to generate a contextualized representation of the input. In this network, this representation is embodied in the final hidden state of the encoder, $h_n$, which in turn feeds into the first hidden state of the decoder. The decoder network on the right takes this state and autoregressively generates a sequence of outputs.

Figure 10.3 Basic RNN-based encoder-decoder architecture. The final hidden state of the encoder RNN serves as the context for the decoder in its role as $h_0$ in the decoder RNN.

This basic architecture is consistent with the original applications of neural models to machine translation. However, it embodies a number of design choices that are less than optimal. Among the major ones are that the encoder and the decoder are assumed to have the same internal structure (RNNs in this case), that the final state of the encoder is the only context available to the decoder, and finally that this context is only available to the decoder as its initial hidden state. Abstracting away from these choices, we can say that encoder-decoder networks consist of three components:

1. An encoder that accepts an input sequence, $x_n$, and generates a corresponding sequence of contextualized representations, $h_n$.
2. A context vector, $c$, which is a function of $h_n$ and conveys the essence of the input to the decoder.
3. A decoder, which accepts $c$ as input and generates an arbitrary length sequence of hidden states $h_m$, from which a corresponding sequence of output states $y_m$ can be obtained.

Figure 10.4 illustrates this abstracted architecture. Let's now explore some of the possibilities for each of the components.

### Word Embeddings

- Neural models manipulate vectors
- In the case of text, **embeddings** are vectors that represent **words**
  - They can also represent **sentences, documents, and even attributes of a table**!
- In the case of words: **word embeddings** – semantics inferred from context
- Example:
  - Predict the following word given a prefix: "When I got home, I forgot to feed the _____."
  - Suppose we see the following training sentence: "When I get home, I have to remember to feed the cat"
  - A traditional model can predict "cat" but not "dog"
  - A neural model can assign high probability also to "dog"
  - Considering that "cat" and "dog" have similar embeddings

### Pre-training and Word Embeddings

- Models assume the existence of previously generated **embeddings** for a large set of words.
- In many cases, the embeddings obtained with methods such as **word2vec** are sufficient to get good results.
- There are many more recent and powerful methods
- The process of generating word embeddings and its properties are itself a subject worth discussing.

### Distributional Hypothesis & Vector Semantics

- **Distributional Hypothesis (HD)**
  - Words with similar meanings tend to occur in similar contexts.
  - Formulated in the 1950's by several linguists
- **Vector Semantics**
  - Instantiates the HD, creating representations of the meaning of words, called **embeddings**, from their distributions in a corpus.
  - Used in NLP applications to exploit word semantics
  - Base for more powerful word representation (e.g., ELMo and BERT)
- **Representation Learning**: embeddings can be learned automatically from input texts
Distributional Hypothesis - Example

- What is Jambú?
- The word was seen in the following contexts:
  - "Jambú is delicious sautéed with garlic"
  - "Jambú is excellent on rice"
  - "... Jambú leaves with salty sauces..."
- Some of the words in the above texts were seen in contexts such as:
  - "... spinach sautéed with garlic over rice ..."
  - "... chard stems and leaves are delicious..."
  - "... collard greens and other salted vegetables..."

Adapted from Jurafsky & Martin, 2019

Vector Semantics

- Words represented as vectors or embedding in a multidimensional semantic space
- Allows to estimate the similarity between words (or sentences).
- Combines two intuitions: Distributional Hypothesis and representation of words as numerical vectors.
- There are several versions of vector semantics, each one defining the elements of the vectors in slightly different ways.
- In general, all of them are based on some form of weighted count of neighbor words

Word Embeddings Examples

- 2D projection of embeddings for a few words.
- Words with similar semantics are close in space
- The close words are not necessarily syntactically "similar"

Reproduced from Xun et al. KDD '17

Adapted from Jurafsky & Martin, 2019
**Word2Vec**

- Algorithm Skip-gram with negative Sampling [Mikolov@NIPS’13]
- Method for generating short and dense embeddings
- Including in the word2vec package and therefore is commonly called word2vec.
- Fast, efficient for training.
- Available online with code and pre-trained embeddings.
- Other popular methods:
  - GloVe [Pennington@EMNLP’14] e fastText [Bojanowski@TACL’17]

**Word2Vec – Intuition**

- Instead of counting how often each word occurs next to a word \( w \), train a binary classifier to calculate the probability of words occurring near \( w \).
- The embedding is formed from the *weights* of the learned classifiers.
- Revolutionary intuition: we can use the current text as an implicitly unsupervised training corpus for this classifier;
- A word \( v \) occurring near \( w \) acts as a positive example.
- Avoids the need for any type of manual labeling
- Proposed in the context of neural language models [Collobert@JMLR’11]

---

**Language-Centered System (LCS)**

- Use pre-trained embeddings to encode and represent:
  - Queries in Natural Language
  - Database schemas
  - Database Instances - all tuples with attribute values
- SQL query generated using Seq2Seq models
- All current systems are supervised
- Many consider that the DB contains a single table

**LCS – Benchmarks**

- Many LCS are focused on a specific benchmark

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>DBs and Tables</th>
<th>Queries</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>WikiSQL [Zhong@CoRR’17]</td>
<td>26,531 tables extracted from Wikipedia HTML tables</td>
<td>80,654 &lt;NL,SQL&gt; pairs No joins Labeled with MeC Turk</td>
<td>Seq2SQL [Zhong@CoRR’17] SQLNet [Xu@CoRR’17] Coarse2Fine [Lapata@ACL’18] STAMP [Zhou@ACL’18] PT-MAML [Huang@NAACL-HLT’18] TypeSQL@Yu@NAACL-HLT’18</td>
</tr>
<tr>
<td>Spider [Yu@EMNLP’18]</td>
<td>200 DBs 12B domains ~5 tables/DB</td>
<td>10.181 NL and 5.693 SQL include Joins Labeled by 11 Grads</td>
<td>SyntaxSQLNet@Yu@EMNLP’18 GNN [Guo@ACL’19] IRNet@Bogin@ACL’19</td>
</tr>
</tbody>
</table>
**Seq2SQL [Zhong@CoRR’17]**
- Developed at Salesforce Research
- Introduces WikiSQL Benchmark - used for training and testing
- Takes advantage of the inherent structure of SQL queries
  - Encodes the NL query and a target table
  - Predicts each part of the SQL query separately

**Seq2SQL [Zhong@CoRR’17] – Inference**
- Aggregation operations
  - An RNN encodes NL query
  - 4 possible outputs: COUNT, MIN, MAX or NONE
- Columns in SELECT (Projections)
  - An RNN encodes combinations of the query and each column
- WHERE Clause (Selection Predicates)
  - An RNN encodes the query, each column and subset of the SQL vocabulary
  - Pointer Network: Output Vocabulary is made up of input words
- Does not support joins

**Seq2SQL [Zhong@CoRR’17] – Training**
- Given the columns of the table, for each NL query, generates a candidate SQL query that runs on the DB
- The result of the execution is used as a reward to train a reinforcement learning algorithm

**Seq2SQL [Zhong@CoRR’17] – Training (2)**

How to Talk to Your Database, by Victor Zhong (https://blog.einstein.ai/how-to-talk-to-your-database/
**Seq2SQL [Zhong@CoRR’17] – Similar Systems**

- Many other similar systems also encode the DB at the input and decode the output using pointer networks
- Some also assume a SQL (Slot-Filling) template:
  - SQLNet [Xu@CoRR’17], Coarse2Fine [Lapata@ACL’18]
  - TypeSQL[Yu@NAACL-HLT’18]
- Others decode the SQL query as a sequence of words
  - STAMP [Zhou@ACL’18], PT-MAML [Huang@NAACL-HLT’18]
- Others decode the SQL query into a syntax tree
  - IRL [Bogin@ACL’19], GNN [Guo@ACL’19], SyntaxSQLNet [Yu@EMNLP’18]

**DBPal [Weir@SIGMOD’20]**

- Johns Hopkins Univ. ,TU Darmstadt e Brown Univ.
- **Seq2Seq + attention** mechanisms
- Focus on using a limited volume of training data
- Generates synthetic training examples
  - Technique known in ML as data augmentation
  - Uses templates and paraphrase
- Improves overall translation precision
- Increases robustness to language variations

**DBPal [Weir@SIGMOD’20] – Output Vocabulary**

- Schema elements in output vocabulary, not input
  - This vocabulary also includes SQL keywords and constant values
- Narrower vocabulary than usual in Seq2Seq: reduces complexity
  - DEC: Chooses words from this vocabulary to generate the SQL query as the resulting sequence

**Consequences**

- Model specializes in target BD
- Can only process a query if it contains vocabulary words
- Model needs to be trained for each new database

**DBPal [Weir@SIGMOD’20] – Training**

- Relies on multiple SQL query templates
  - For each template, there is 1 or more NL templates
- Training generator:
  - Instantiate NL templates with schema elements.
  - NL slots filled with words/phrases from a manually constructed dictionary.

---

**Details of the model architecture and the hyperparameters of our training pipeline in detail, we discuss an optimiza-**

**June 23–28, USA**

**Table, column, and attribute**(s)

**SelectPhrase**

**FromPhrase**

**WherePhrase**

**Filter** slots are

**Parameter**

**Generator**

**Augmentation**

**Lenormalizer**

**Natural Translator**

**DBMS**

**Post-processor**

**Generated training set:**

- What are cities whose state is New Mexico?  
  - SELECT city FROM cities WHERE state = 'New Mexico'
- Show me states with a population over 5 million:  
  - SELECT state FROM cities WHERE population > 5000000
**DBPal [Weir@SIGMOD’20] – Training (2)**

- **Example Template**
  - SQL: SELECT (Att)(s) FROM (Table) WHERE (Filter)
  - LN: (SelectPhrase) the (Att)(s) (FromPhrase) (Table)(s) (Filter)
- **Example of an instantiated template**
  - SQL: SELECT name FROM patient WHERE age=20
  - LN: Show me the name of all patients with age 20
- **Currently, there are about 100 templates**
- **Typical training sets**
  - DBs with a single table: ~1 MM <NL,SQL> template pairs
  - DBs with more tables: ~2 to 3 MM <NL,SQL> template pairs
- **Augmentation**: Generation of synthetic pairs
  - **Goal**: covering a broad spectrum of linguistic variations for the same SQL query.
  - **Add pairs <NL,SQL> with linguistic variations in NL**
  - **Words and subphrases randomly exchanged in NL queries**, using paraphrases provided by the Paraphrase Database (PPDB)
    - Show patients names with age @AGE => Display patients names with age @AGE.
  - **Lemmatization**: normalize words in <NL, SQL>
    - E.g.: "cars" and "car’s" replaced by "car".
    - Also applied in run time

**DBPal [Weir@SIGMOD’20] – Execution**

- **Neural Translator**: translates the query
- **Outcome from training**.
- **Query constants replaced by markers (placeholders)**
- **Makes the query independent from the DB state used in the training**.
- **Then, the lemmatizer is applied**
- **Postprocessor**: Replaces markers with constants
- **The query can run in the DBMS**

**Queries over Multiple Tables**
References to DB

will smith films

Query Match 1

set of tuples
matching will smith

set of tuples
matching films

set of tuples
matching films

set of tuples
matching will smith

Lathe Overview

In this chapter, we present an overview of our system Lathe for generating SQL queries given a keyword query with references to the database schema.

For this, we use the sample movie database we illustrate in Figure 3.1:

Consider that a user inputs the keyword query

```
will smith
```

We begin by presenting a simple example of the task carried out by the method.

References to DB

will smith films

Query Match 2

set of tuples
matching will smith

set of tuples
matching films

set of tuples
matching will smith

set of tuples
matching films

Figure 3.1: A sample movie database taken from IMDB

```
Internet Movie Database https://www.imdb.com/interfaces/
```
Candidate Networks Generated

will smith films

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>ID</th>
<th>Person_ID</th>
<th>Movie_ID</th>
<th>ID</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Will Smith</td>
<td>25</td>
<td>1</td>
<td>6</td>
<td></td>
<td>Men in Black</td>
<td>1997</td>
</tr>
<tr>
<td>1</td>
<td>Will Smith</td>
<td>26</td>
<td>1</td>
<td>7</td>
<td></td>
<td>I am Legend</td>
<td>2007</td>
</tr>
</tbody>
</table>

Candidate Network Generation

- Combinatorial Problem: answers must include all references minimally, i.e., without redundancies
- Example: Mondial Database (CIA Factbook)
  - 28 tables, 17,115 tuples, 104 FKs
- For the query "South East" :
  - 208 possible Query Matches
  - 105 possible Candidate Networks
    - Up to 10 tables involved

Candidate Network Generation - Approaches

- Problem raised in the context of the DISCOVER system [Hristidis@VLDB'02], pioneer work in keyword queries over DBs
- A series of works produced in our group improved the efficiency in the generation process and the quality of the Candidate Networks
- Efficient Generation of Candidate Networks
  - [Oliveira@ICDE'18] and [Oliveira@TKDE'20]
- Ranking of Candidate Networks
  - [Oliveira@ICDE'15] and [Oliveira@TKDE'20]

Conclusion and Remarks
What was not covered here …

- Systems for keyword queries in relational BDs
  - [Yu@IDE'B10] : A little old survey
  - [Affolter@VLDBJ'19] : Much more recent. It also covers various DCS

- Experimental Results
  - [Kim@PVLDB'20]: Excellent recent survey with experimental results of several NLIDBs with various benchmarks.

- Applications in Conversational and Dialogue Systems
  - [Ozcan@SIGMOD'20]: Tutorial at SIGMOD 2020.
    - Authors from the ATHENA/ATHENA++ group at IBM. It also covers several NLIDBs.

Some Further Developments

- Database Exploration - Tool for Data Scientists and Analyst
  - Doctors, biologists, financial analysts, lawyers, marketing staff, ...
  - Old proposal [Dar@VLDB'98], but only recently carried out.
  - Examples: SODA [Blunschi@VLDB'12] and ATHENA [Lei@IDE'B18]

- Natural language as inter-model Lingua Franca
  - Polystores [Duggan@SIGREC'15]: Federations of DBs with multiple data models
  - Data Lakes : centralized repository of raw or minimally cured data available to perform analytical activities [Terrizzano@CIDR'15]
  - Idea explored with keyword queries at INRIA [Hadda@CoRR'20]

- Somewhat surprising connection with the schema evolution problem
  - More "relaxed" queries are less vulnerable to changes in the DB schema
  - Idea explored in LESSQL [Afonso@SANER'20] developed by our group.

Thanks to …

- The Divine Wisdom
- Mirian, Genoveva and Anne-Lyse for the kind invitation
- You all for attending ... I am honored
- UFAM, Institute of Computing, Graduate Program in Informatics
- The Database and Information Retrieval Group
- JustoBraz and Méliuz
  - Research Support and access to real problems that matter
- CAPES, CNpq and FAPEAM
  - Research Support
- Paulo Martins, Lucas Citolin, Brandell Ferreira,
- SAMSUNG: Support to Paulo Martins

References – Surveys and Tutorial

- [Ozcan@SIGMOD'20] Fatma Ozcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, Vasiliis Efthymiou: State of the Art and Open Challenges in Natural Language Interfaces to Data. SIGMOD Conference 2020: 2629-2636
  - IBM Tutorial - same authors of ATHENA
Further References (1)

- [Blunschi@VLDB'12] Lukas Blunschi, Claudio Jossen, Donald Kossmann, Kurt Stockinger: SODA: Generating SQL for Business Users. Proc. VLDB Endow. 5(10): 932-943 (2012) SODA

Further References (2)

- [Codd@IFIP'74] E. F. Codd: Seven Steps to Rendezvous with the Casual User. IFIP Working Conference Data Base Management 1974: 179-200

Further References (3)

Further References (4)

- [Li@PVDB'14] F. Li and H. V. Jagadish. Constructing an interactive natural language interface for relational databases. PVLDB, 8(1):73–84, 2014. NAULR

Further References (5)

- [Mikolov@NIPS'13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregor S. Corrado, Jeffrey Dean: Distributed Representations of Words and Phrases and their Compositionality. NIPS 2013: 3111-3119
- [Oliveira@ICDE'15] Pericles de Oliveira, Altigran Soares da Silva, Edelho Silva de Moura: Ranking Candidate Networks of relations to improve keyword search over relational databases. ICDE 2015: 399-410
- [Oliveira@ICDE'18] Pericles de Oliveira, Altigran Soares da Silva, Edelho Silva de Moura; Rosiane Rodrigues: Match-Based Candidate Network Generation for Keyword Queries over Relational Databases. ICDE 2018: 1344-1347
- [Oliveira@TKDE'20] Pericles de Oliveira, Altigran Soares da Silva, Edelho Silva de Moura; Rosiane Rodrigues, "Efficient Match-Based Candidate Network Generation for Keyword Queries over Relational Databases," in IEEE Transactions on Knowledge and Data Engineering,
- [Ozcan@SIGMOD'20] Fatma Ozcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, Vasillis Efthymiou: State of the Art and Open Challenges in Natural Language Interfaces to Data. SIGMOD Conference 2020: 2629-2636 Tutorial IBM
- [Pennington@EMNLP'14] Jeffrey Pennington, Richar Socher, Christopher D. Manning: Glove: Global Vectors for Word Representation. EMNLP 2014: 1532-1543

Further References (6)


Further References (7)

- [Wei@SIGMOD'20] Nathaniel Wei, Prasetya Utama, Alex Galakatos, Andrew Croffy, Amir Ikhiche, Shekar Ramaswamy, Robin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, Ugur Çetintemel, Carsten Binning: DBPAL: A Fully Pluggable NLSQL Training Pipeline. SIGMOD Conference 2020: 2347-2361 DBPAL
- [Xu@CoRR'17] X. Xu, C. Liu, and D. Song. Spider: A large scale human-labeled dataset for complex and cross domain semantic parsing and text-to-sql tasks. In EMNLP, pages 1532-1543
Further References (8)