Context Management in Database Systems with Word Embeddings

Status Talk - Michael Günther

Supervisors: Prof. Wolfgang Lehner
Prof. Susanne Strahringer

09 November 2020
Motivation

Roles for the Schema

- **Roles** in database **schema** and **query** language

  → Manage complexity and change **inside** one **dataset**

Roles for the Data

- Model **semantic roles** of text **data** in the database in **global vector** space

  → Manage complexity and change **across datasets**

"current software systems have to cope with increased **complexity** and **changes**" \cite{JKVL14}
Motivation

Kaggle Survey 2017: What do data scientists at work?

Question: “At work, which kind of data do you typically work with? “

- Relational data: 65.5%
- Text data: 53.0%
- Image data: 18.1%
- Other: 10.3%
- Video data: 5.1%

https://www.kaggle.com/surveys/2017

8,024 responses
**Motivation**

*Kaggle Survey 2017: What do data scientists at work?*

Question: „At work, which kind of data do you typically work with?“

- Relational data: 47.7%
- Text data: 53.0%
- Image data: 18.1%
- Other: 10.3%
- Video data: 5.1%

https://www.kaggle.com/surveys/2017

---

**Database Systems** should provide infrastructure to analyze **text data**!

---

**DB Engines Popularity**

The most popular database management systems*

November 2020 | Score
--- | ---
1. Oracle | 1345
2. MySQL | 1242
3. Microsoft SQL Server | 1038
4. PostgreSQL | 555
5. MongoDB | 454

* https://db-engines.com/en/ Access 02.11.2020
At work, which kind of data do you typically work with?

SQL Capabilities

Many operations and functions to analyze numbers

Very limited text analysis capabilities

Database Systems should provide infrastructure to analyze text data!
Word Embeddings: State-of-the-Art Text Analysis

- Data Storage with textual data
- Extracted text data
- Language Model
  - Numerical Representation (Vectors)
  - Inception: [0.54, -0.71, 0.11, ...]
  - Shutter_Island: [0.31, -0.59, -0.08, ...]
  - ....
Numerical Representation (Vectors)

Inception: [0.54, -0.71, 0.11, ...]

Shutter Island: [0.31, -0.59, -0.08, ...]

State-of-the-art Language Models: Word Embeddings

Extract Weights as Pre-Trained Language Model

Deep Neuronal Network

Large Text corpora in natural language

Training on Dummy Task

Extracted text data

Data Storage with textual data

Extracted text data

Language Model

Numerical Representation (Vectors)
Word Embeddings: State-of-the-Art Text Analysis

Semantic Vector Space

- woman
- man
- king
- prince
- London
- Berlin
- England
- Germany
- dry
- drier
- driest
- wet
- wetter
- wettest

Inception: [0.54, -0.71, 0.11, ...]
Shutter_Island: [0.31, -0.59, -0.08, ...]

Data Storage with textual data

Extracted text data

Language Model

Numerical Representation (Vectors)
Word Embeddings: State-of-the-Art Text Analysis

Data Storage with textual data

Extracted text data

Language Model

Applications of Word Embeddings

AI Applications

Data Integration and Exploration

Similarity Search Tasks

Inception: [0.54, -0.71, 0.11, ...]

Shutter_Island: [0.31, -0.59, -0.08, ...]

Numerical Representation (Vectors)
Word Embeddings in Analytical Systems

AI Applications

- Utilize *implicitly* encoded knowledge from large text corpora
- Capture *semantic similarity* of text values
Word Embeddings in Analytical Systems

**AI Applications**
- Utilize *implicitly* encoded knowledge from large text corpora
- Capture *semantic similarity* of text values

**Data Management**
- Semantic text *similarity queries*
- Data exploration
- Data integration
Word Embeddings in Analytical Systems

**AI Applications**
- Utilize implicitly encoded knowledge from large text corpora
- Capture semantic similarity of text values

**Data Management**
- Semantic text similarity queries
- Data exploration
- Data integration

**Information Retrieval**
- Semantic search
- Query expansion
- Multi-lingual search
Objectives of Integration of Word Embeddings in Database Systems

- Utilize implicitly encoded knowledge from large text corpora
- Capture semantic similarity of text values

Data Management
- Semantic text similarity queries
- Data exploration
- Data integration

Information Retrieval
- Semantic search
- Query expansion
- Multi-lingual search

Database systems should handle word embeddings as first-class citizens to support those tasks!
Objectives of Integration of Word Embeddings in Database Systems

- Utilize implicitly encoded knowledge from large text corpora
- Capture semantic similarity of text values

Database systems should handle word embeddings as first-class citizens to support those tasks!

Functionality

Adaptive Modeling
Objectives of Integration of Word Embeddings in Database Systems

Database systems should handle word embeddings as first-class citizens to support those tasks!

Data Management

- Utilize implicitly encoded knowledge from large text corpora
- Capture semantic similarity of text values

Functionality

- Efficient retrieval of embeddings
- Efficient vector calculations

Adaptive Modeling

- Adaptive modelling
- Adaptive embedding calculations (e.g. context-aware similarity search)
Objectives of Integration of Word Embeddings in Database Systems

Database systems should handle word embeddings as first-class citizens to support those tasks!

Data Management

- Utilize implicitly encoded knowledge from large text corpora
- Capture semantic similarity of text values

Information Retrieval

- Semantic search
- Query expansion
- Multi-lingual search

Database systems should handle word embeddings as first-class citizens to support those tasks!

Functionality

- Efficient retrieval of embeddings
- Efficient vector calculations

Adaptive Modeling

- Adaptive modeling
- Adaptive embedding calculations (e.g. context-aware similarity search)

- Optimize Embeddings → Improve Similarity Metrics → Improve ML model
- Context-specific representations
- Model relations like role types in ER-Model

Adaptive Modeling

- Efficient retrieval of embeddings
- Efficient vector calculations

Functionality

Context-aware Adaptation

- Efficient retrieval of embeddings
- Efficient vector calculations

Adaptive Modeling

- Adaptive modeling
- Adaptive embedding calculations (e.g. context-aware similarity search)

- Optimize Embeddings → Improve Similarity Metrics → Improve ML model
- Context-specific representations
- Model relations like role types in ER-Model

Functionality

Context-aware Adaptation

- Efficient retrieval of embeddings
- Efficient vector calculations

Adaptive Modeling

- Adaptive modeling
- Adaptive embedding calculations (e.g. context-aware similarity search)

- Optimize Embeddings → Improve Similarity Metrics → Improve ML model
- Context-specific representations
- Model relations like role types in ER-Model
Objectives of Integration of Word Embeddings in Database Systems

Database systems should handle word embeddings as first-class citizens to support those tasks!

**Data Management**
- Utilize implicitly encoded knowledge from large text corpora
- Capture sematic similarity of text values

**Information Retrieval**
- Semantic search
- Query expansion
- Multi-lingual search

**Database systems** should handle word embeddings as first-class citizens to support those tasks!

**Functionality**
- Efficient retrieval of embeddings
- Efficient vector calculations
- Optimize Embeddings
  - Improve Similarity Metrics
  - Improve ML model
- Represent Schema Terms
- Improve Table Understanding

**Adaptive Modeling**
- Adaptive modelling
- Adaptive embedding calculations (e.g. context-aware similarity search)
- Context-specific representations
- Model relations like role types in ER-Model
- Model relations between schema and instance text data (e.g. role-play relations)
Objectives

1. Data Management

2. Context-Aware Adaptation

3. Representation of Schema Information

Applications of Word Embeddings in Database Systems
Objectives

1. Data Management

2. Context-Aware Adaptation

3. Representation of Schema Information

Applications of Word Embeddings in Database Systems
Data Management for Word Embeddings

SQL Query

SELECT * [...]

Structured data

Query Execution
Data Management for Word Embeddings

Storage Engine
- **Data structure** for word embeddings in RDBMS
- Updatable embeddings

Query Execution
- SQL Query: `SELECT * [...]`
- Structured data
- Word vectors and index data
- Word embedding datasets

- M1
- M2
- M3
Data Management for Word Embeddings

Storage Engine
- **Data structure** for word embeddings in RDBMS
- Updatable embeddings

Word Embedding Operations
- Similarity joins to support WE applications
- Derived operations (e.g. analogies)

Query Execution

```
SELECT *
```

SQL Query

Word Embedding Operations

Structured data

Word vectors and index data

M1

M2

M3

Word embedding datasets
Data Management for Word Embeddings

**Storage Engine**
- **Data structure** for word embeddings in RDBMS
- Updatable embeddings

**Word Embedding Operations**
- Similarity joins to support WE applications
- Derived operations (e.g. analogies)

**Efficiency**
- Updatable index structures
- Approximated operations

**Query Execution**

```
SELECT * [X]
```

SQL Query

Structured data

Word vectors and index data

Word embedding datasets
Data Management for Word Embeddings

**Storage Engine**
- **Data structure** for word embeddings in RDBMS
- Updatable embeddings

**Word Embedding Operations**
- Similarity joins to support WE applications
- Derived operations (e.g., analogies)

**Efficiency**
- Updatable index structures
- Approximated operations

**Adaptivity**
- Multiple WE models to represent text values
- Domain-specific recommendation

**Query Execution**
- SELECT *
- SQL Query
- Word Embedding Operations
- Structured data
- Word vectors and index data

**Recommendation**
- Evaluation results
- Multiple WE models
- M1, M2, M3
- Word embedding datasets
- M1, M2, M3
Cognitive Intelligence Queries


**Word Embedding Operations in Spark**
- Extend **Apache Spark** with WE operations

```sql
SELECT * ...
```

**Structured data**

**Query Execution**
Cognitive Intelligence Queries


Word Embedding Operations in Spark
- Extend Apache Spark with WE operations

Internal and External Word Embeddings
1. **Internal:** Serialize tables as sequences of text → apply word embedding model

Vector Generation
- Structured data
- Serialized text
- Internal word embedding datasets

Query Execution
- SQL Query
- Structured data
- Word vectors and index data

Cognitive Intelligence Queries


Word Embedding Operations in Spark
- Extend Apache Spark with WE operations

Internal and External Word Embeddings
1. **Internal**: Serialize tables as sequences of text → apply word embedding model
2. **External**: Additional embeddings trained on large text corpora are added.

---

**Vector Generation**
- External word embedding datasets
- Word vectors and index data

**Query Execution**
- SQL Query
  ```sql
  SELECT * [...]
  ```
Cognitive Intelligence Queries


**Word Embedding Operations in Spark**
- Extend **Apache Spark** with WE operations

**Internal and External Word Embeddings**
1. **Internal**: Serialize tables as sequences of text → apply word embedding model
2. **External**: Additional embeddings trained on large text corpora are added.
   - **Storing** word vectors in **designated tables**
Cognitive Intelligence Queries


**Word Embedding Operations in Spark**

- Extend *Apache Spark* with WE operations

**Internal and External Word Embeddings**

1. **Internal**: Serialize tables as sequences of text → apply word embedding model
2. **External**: Additional embeddings trained on large text corpora are added.
   - **Storing** word vectors in designated tables

**Additional User-Defined Functions**

- Use word embeddings in **UDFs** for semantic text queries
Cognitive Intelligence Queries


Word Embedding Operations in Spark
- Extend Apache Spark with WE operations

Internal and External Word Embeddings
1. **Internal:** Serialize tables as sequences of text → apply word embedding model
2. **External:** Additional embeddings trained on large text corpora are added
    - Storing word vectors in designated tables

Additional User-Defined Functions
- Use word embeddings in UDFs for semantic text queries

### WE Operations

**Similarity**

\[
sim_{\cos}(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}
\]

**Query Execution**

```
SELECT m1.title, m2.title
FROM movies AS m1, movies AS m2
WHERE similarity(m1, m2) > 0.5
```

**CI Operations**

Inception
Shutter_Island

Cognitive Intelligence Queries

Word Embedding Operations in Spark
- Extend Apache Spark with WE operations

Internal and External Word Embeddings
1. **Internal**: Serialize tables as sequences of text → apply word embedding
2. **External**: Additional embeddings trained on large text corpora are added
- Storing word vectors in designated tables

Additional User-Defined Functions
- Use word embeddings in UDFs for semantic text queries

**WE Operations**

**Similarity**

\[ \text{sim}_{\text{cos}}(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} \]

**Analogy**

```
SELECT analogy('Godfather', 'Francis_Ford_Coppola', m.title)
FROM movies AS m
```

```
SELECT m1.title, m2.title
FROM movies AS m1, movies AS m2
WHERE similarity(m1, m2) > 0.5
```

Query Execution

CI Operations

[Diagram showing word embeddings, analogy, and similarity calculations]
Cognitive Intelligence Queries

Word Embedding Operations in Spark
- Extend Apache Spark with WE operations

Internal and External Word Embeddings
1. **Internal**: Serialize tables as sequences of text → apply word embedding model
2. **External**: Additional embeddings trained on large text corpora are added
- Storing word vectors in designated tables

Additional User-Defined Functions
- Use word embeddings in UDFs for semantic text queries

**WE Operations**

**Similarity**

\[ \text{sim}_\text{cos}(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|} \]

**Analogy**

```
SELECT analogy('Godfather', 'Francis_Ford_Coppola', m.title)
FROM movies AS m
```

**Query Execution**

- Select data
- Word vectors and index data
- External word embeddings
- CI Operations
- Extend Spark with WE operations
- Internal and external embeddings
- Storing word vectors in designated tables
- Use word embeddings in UDFs for semantic text queries
Word Embedding Operations in Spark

- Extend Apache Spark with WE operations

Internal and External Word Embeddings

1. **Internal**: Serialize tables as sequences of text → apply word embedding model
2. **External**: Additional embeddings trained on large text corpora are added
   - Storing word embedding in tables
   - UDFs can access tables
   - But no RDBMS (Spark)

### Additional User-Defined Functions

- Use word embeddings in UDFs for semantic text queries

**Query Execution**

- SQL Query

### Cognitive Intelligence Queries

Cognitive Intelligence Queries


Word Embedding Operations in Spark

- Extend Apache Spark with WE operations

Internal and External Word Embeddings

1. **Internal**: Serialize tables as sequences of text → apply word embedding model
2. **External**: Additional embeddings trained on large text corpora are added.
   - **Storing** word vectors in designated tables

Additional User-Defined Functions

- Use word embeddings in UDFs for semantic text queries

<table>
<thead>
<tr>
<th>Storage</th>
<th>WE Operations</th>
<th>Efficiency</th>
<th>Adaptivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cognitive Intelligence Queries


**Word Embedding Operations in Spark**

- Extend **Apache Spark** with WE operations

**Internal and External Word Embeddings**

1. **Internal**: Serialize tables as sequences of text → apply word embedding model
2. **External**: Additional embeddings trained on large text corpora are added.
   - **Storing** word vectors in designated tables

**Additional User-Defined Functions**

- Use word embeddings in UDFs for **semantic text queries**

---

<table>
<thead>
<tr>
<th>Storage</th>
<th>WE Operations</th>
<th>Efficiency</th>
<th>Adaptivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>(✓)</td>
<td></td>
</tr>
</tbody>
</table>

---

**Query Execution**

- SELECT * […]
- SQL Query
- CI Operations

- Some operations use **simple inverted indexing**
- No batch-wise execution

---

- Structured data
- Word vectors and index data
- External word embeddings
Word Embedding Operations in Spark
- Extend Apache Spark with WE operations

Internal and External Word Embeddings
1. Internal: Serialize tables as sequences of text → apply word embedding model
2. External: Additional embeddings trained on large text corpora are added.
   - Storing word vectors in designated tables

Additional User-Defined Functions
- Use word embeddings in UDFs for semantic text queries

<table>
<thead>
<tr>
<th>Storage</th>
<th>WE Operations</th>
<th>Efficiency</th>
<th>Adaptivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>(✓)</td>
<td>✘</td>
</tr>
</tbody>
</table>

Query Execution
- SQL Query
- CI Operations
- Structured data
- Word vectors and index data
- External word embeddings

No support for adaptive word embedding representation

Vector Extension to PostgreSQL

SELECT * […]

Query Execution

PASE: ANN Index Operations

Structured data


Vector Extension to PostgreSQL

- New vector type and similarity operations

```
SELECT * [...] 
```

SQL Query

Query Execution

PASE: ANN Index Operations

ANN Index Structures: IVFFlat

- Inception
- Shutter Island
- Forrest Gump

38
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)


Vector Extension to PostgreSQL

- New vector type and similarity operations

Add Index Structures for Approximated Nearest Neighbor Search

- Cluster vectors to reduce disk access
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)


Vector Extension to PostgreSQL

- New vector type and similarity operations

Add Index Structures for Approximated Nearest Neighbor Search

- Cluster vectors to reduce disk access
- Nearest neighbor (similarity) search:
  1. Determine nearest cluster centroid

Query Execution

```
SELECT * […]
```

PASE: ANN Index Operations

ANN Index Structures: IVFFlat
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)


Vector Extension to PostgreSQL

- New **vector type** and **similarity operations**

Add Index Structures for Approximated Nearest Neighbor Search

- Cluster vectors to reduce disk access
- **Nearest neighbor (similarity) search:**
  1. Determine nearest cluster centroid
  2. Read only vectors in the cluster

**Query Execution**

```
SELECT * [...] + PASE: ANN Index Operations
```

**ANN Index Structures: IVFFlat**

- Inception
- Shutter Island
- Forrest Gump
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)

Vector Extension to PostgreSQL

- New vector type and similarity operations

Add Index Structures for Approximated Nearest Neighbor Search

- Cluster vectors to reduce disk access
- Nearest neighbor (similarity) search:
  1. Determine nearest cluster centroid
  2. Read only vectors in the cluster
  3. Calculate similarity to vectors in the cluster and determine nearest vector
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)


Vector Extension to PostgreSQL
- New vector type and similarity operations

Add Index Structures for Approximated Nearest Neighbor Search
- Cluster vectors to reduce disk access
- Nearest neighbor (similarity) search:
  1. Determine nearest cluster centroid
  2. Read only vectors in the cluster
  3. Calculate similarity to vectors in the cluster and determine nearest vector
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)


Vector Extension to PostgreSQL

- New vector type and similarity operations

Add Index Structures for Approximated Nearest Neighbor Search

- Cluster vectors to reduce disk access
- Nearest neighbor search:
  1. Determine nearest cluster centroid
  2. Read only vectors in the cluster
  3. Calculate similarity to vectors in the cluster and determine nearest vector

Storage | Word vectors and index data | Efficiency | Adaptivity
--- | --- | --- | ---
✓ | | | |

Query Execution

SELECT * [...] SQL Query

PASE: ANN Index Operations

Structured data +

Word vectors and index data

Word embedding datasets
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)


Vector Extension to PostgreSQL

- New vector type and similarity operations

Add Index Structures for Approximated Nearest Neighbor Search

- Cluster vectors to reduce disk access
- Nearest neighbor search
  1. Determine nearest cluster centroid
  2. Read only vectors in the cluster
  3. Calculate similarity to vectors in the cluster and determine nearest vector

Storage, WE Operations, Efficiency, Adaptivity

<table>
<thead>
<tr>
<th>Storage</th>
<th>WE Operations</th>
<th>Efficiency</th>
<th>Adaptivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>(✓)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query Execution

```
SELECT * [...]  
```

SQL Query

PASE: ANN Index Operations

Structured data

Word vectors and index data

Word embedding datasets

kNN search is implemented

No specific embedding functions
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)


**Vector Extension to PostgreSQL**
- New vector type and similarity operations

**Add Index Structures for Approximated Nearest Neighbor Search**
- Cluster vectors to reduce disk access
- Nearest neighbor (similarity)
  1. Determine nearest cluster centroid
  2. Read only vectors in the cluster
  3. Calculate similarity to vectors in the cluster and determine nearest vector

---

**Storage**

<table>
<thead>
<tr>
<th>WE Operations</th>
<th>Efficiency</th>
<th>Adaptivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

---

**Query Execution**

- SQL Query
  ```sql
  SELECT * [...] 
  ```

- PASE: ANN Index Operations

Word vectors and index data

---

- Efficient kNN search
- Adaptive search space
- Updatable index
- No efficient kNN-Join

---

**Word embedding datasets**
PASE (PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension)


**Vector Extension to PostgreSQL**
- New vector type and similarity operations

**Add Index Structures for Approximated Nearest Neighbor Search**
- Cluster vectors to reduce disk access
- Nearest neighbor (similarity) search:
  1. Determine nearest cluster centroid
  2. Read only vectors in the cluster
  3. Calculate similarity to vectors in the cluster and determine nearest vector

---

**Query Execution**

```
SELECT * [...]
```

**PASE: ANN Index Operations**

**Storage**

<table>
<thead>
<tr>
<th>WE Operations</th>
<th>Efficiency</th>
<th>Adaptivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>

No support for adaptive word embedding representation

Word vectors and index data

Word embedding datasets
FREDDY: Fast Word Embeddings in Database Systems


SELECT *

SQL Query

Word-Embedding Operations

Structured data

Word vectors and index data

Recommendation

M1 Evaluation results

Evaluation results

Recommend Model

Publications

[Gün18] SIGMOD’18

M1 Word embedding datasets

M2 Word embedding datasets
FREDDY: Fast Word Embeddings in Database Systems

Highly Optimized kNN-Join

- Design operations as specific kNN-Join operations
- kNN-Join is highly optimized

**WE Operations**

High-Level WE Operations

- most similar()
- analogy()
- group()

Foundation: Efficient kNN-Join

- Batch-wise execution
- Fast approximation
- Adaptive search space
- Updateable index

Query Execution

Word-Embedding Operations

FREDDY: Fast Word Embeddings in Database Systems


**Highly Optimized kNN-Join**
- Design operations as specific kNN-Join operations
- kNN-Join is highly optimized

**Dataset Selection**
- Maintain multiple word embedding models
- Model selection via SQL
- Operations automatically use selected model

[Diagram showing query execution, word-embedding operations, dataset selection, and recommendation]

**Recommendation**
- M1 Evaluation results
- M2 Word embedding datasets
- M3

[Graph showing evaluation results for different models]

[Publications]
- [ Gün18 ] SIGMOD’18
- [ GTL19 ] BTW’19
FREDDY: Fast Word Embeddings in Database Systems


**Highly Optimized kNN-Join**
- Design operations as specific kNN-Join operations
- kNN-Join is highly optimized

**Dataset Selection**
- Maintain multiple word embedding models
- Model selection via SQL
- Operations automatically use selected model

**Model Recommendation**
- Automatically build eval dataset from large tabular corpus
- Fine-granular evaluation of word embedding models
- Domain-specific model recommendation (e.g. for column, application domain, relation type, ... )
### Data Management - Conclusion

<table>
<thead>
<tr>
<th>Approach</th>
<th>Storage</th>
<th>WE Operations</th>
<th>Efficiency</th>
<th>Adaptivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASE</td>
<td>✓</td>
<td>(√)</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>CI Queries</td>
<td>✓</td>
<td>✓</td>
<td>(√)</td>
<td>✗</td>
</tr>
<tr>
<td>FREDDY</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Objectives

1. Data Management

2. Context-Aware Adaptation

3. Representation of Schema Information

Applications of Word Embeddings in Database Systems
Context-Aware Adaptation

<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th_Element</td>
<td>Luc Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
</tr>
</tbody>
</table>
Context-Aware Adaptation

Context-Specific Semantic of Text

Brazil (Country) VS Brazil (Sci-fi Movie)

<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th_Element</td>
<td>Luc Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
</tr>
</tbody>
</table>
Context-Aware Adaptation

Context-Specific Semantic of Text

Brazil (Country) vs Brazil (Sci-fi Movie)

Unknown / Rare Words

Rare words obtain poor embedding representation
Context-Aware Adaptation
Context-Aware Adaptation

**Expressiveness**

Support **columnar, row-wise, and foreign key relations**
Context-Aware Adaptation

Expressiveness
Support columnar, row-wise, and foreign key relations

Holistic
Combine knowledge from pre-training on textual corpora with relational knowledge

<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th_Element</td>
<td>Luc Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
</tr>
</tbody>
</table>

Understanding Relational Data

Adapted Embeddings

Movie Director Graph

Luc_Besson
5th_Element
Brazil
Valerian

Relational Data

Luc_Besson
5th_Element
Brazil
Valerian

Embeddings

Luc_Besson
5th_Element
Brazil
Valerian

Adapted Embeddings

d.Luc_Besson
m.5th_Element
Brazil
m.Valerian
m.Brazil
Context-Aware Adaptation

**Expressiveness**
Support *columnar, row-wise, and foreign key relations*

**Holistic**
Combine knowledge from *pre-training on textual corpora with relational knowledge*

**Online Updatable**
Get representations for *text values inserted* in the database *at a later point of time* without retraining

---

<table>
<thead>
<tr>
<th>Relational Data</th>
<th>Embeddings</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th_Element</td>
<td>Luc_Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry_Gilliam</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc_Besson</td>
</tr>
</tbody>
</table>

Adapted Embeddings:
- m.5th_Element
- m.Valerian
- m.Brazil
- d.Luc_Besson

---

**Table**

<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th_Element</td>
<td>Luc_Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry_Gilliam</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc_Besson</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Context-Aware Adaptation

**Expressiveness**

Support *columnar, row-wise, and foreign key* relations

**Holistic**

Combine knowledge from *pre-training on textual corpora with relational knowledge*

**Online Updatable**

Get representations for *text values inserted* in the database at a later point of time without retraining

**RDBMS Integration**

Automatically build representations from a RDBMS

---

RDBMS Integration

- Automatically build representations from a RDBMS

---

### Embeddings

- **Luc_Besson**
- **5th_Element**
- **Brazil**
- **Valerian**

### Relational Data

- **movie**
  - 5th_Element
  - Brazil
  - ... Valerian
  - ... 5th_Element

- **director**
  - ... Luc_Besson
  - ... Terry_Gilliam
  - ... Luc_Besson

### Adapted Embeddings

- **m.5th_Element**
- **m.Valerian**
- **m.Brazil**
- **d.Luc_Besson**
Joint Embedding Models: RC-Net

Joint Embedding Models: RC-Net

Joint Embedding Models: RC-Net


**RC Data Model**

- **Categorical Connections:** relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections:** relations between two entities (e.g. entities in the same row, foreign key relations in DB system)
Joint Embedding Models: RC-Net


**RC Data Model**

- **Categorical Connections**: relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections**: relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

**Joint Loss function**

- Loss function combined word embedding (skip-gram) loss with loss for categorical and relational connections

**Objective Function**

\[ J = \alpha E_r + \beta E_c - L \]
Joint Embedding Models: RC-Net


**RC Data Model**

- **Categorical Connections**: relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections**: relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

**Joint Loss function**

- Loss function combined word embedding (skip-gram) loss with loss for categorical and relational connections

**Objective Function**

\[ J = \alpha E_r + \beta E_c - L \]

- Relational loss
  \[ d(h + r, t) \]
  Minimize \( d(h + r, t) \)
Joint Embedding Models: RC-Net


**RC Data Model**

- **Categorical Connections**: relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections**: relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

**Joint Loss function**

- Loss function combined word embedding (skip-gram) loss with loss for categorical and relational connections

![Diagram of RC-Net model](image)

**Objective Function**

\[ J = \alpha E_r + \beta E_c - L \]

- **Relational loss**: Minimize \( d(h + r, t) \)
- **Categorical loss**: Minimize \( \sum_{e_1 \in C} \sum_{e_2 \in C} w \cdot d(e_2, e_2) \)
Joint Embedding Models: RC-Net


**RC Data Model**

- **Categorical Connections**: relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections**: relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

**Joint Loss function**

- Loss function combined word embedding (skip-gram) loss with loss for categorical and relational connections

![Joint Loss function diagram](image)

\[ J = \alpha E_r + \beta E_c - L \]

- **Relational loss**: \( d(h + r, t) \)
- **Categorical loss**: \( \sum_{e_1 \in C} \sum_{e_2 \in C} w \cdot d(e_2, e_2) \)
- **Word embedding loss**
Joint Embedding Models: RC-Net


**RC Data Model**
- **Categorical Connections**: relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections**: relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

**Joint Loss function**
- Loss function combined word embedding (skip-gram) loss with loss for categorical and relational connections
- Optimization with neural network

**Objective Function**

\[ J = \alpha E_r + \beta E_c - L \]

- **Relational loss**
  \[ \min d(h + r, t) \]
- **Categorical loss**
  \[ \sum_{e_1 \in C} \sum_{e_2 \in C} w \cdot d(e_2, e_2) \]
- **Word embedding loss**

Embedding dataset
Joint Embedding Models: RC-Net


RC Data Model

- **Categorical Connections:** relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections:** relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

Joint Loss function

- Loss function combined word embedding (skip-gram) loss with loss for categorical and relational connections
- Optimization with neural network

Expressivness  Holistic  Updateable  RDBMS Integration

(✓)

Objective Function

\[ J = \alpha E_r + \beta E_c - L \]

Relational Data

- Terry_Gilliam
- Luc_Besson
- Valerian
- 5th_Element

Relational loss

Minimize \( d(h + r, t) \)

Categorical loss

\[ \sum_{e_1 \in C} \sum_{e_2 \in C} w \cdot d(e_2, e_2) \]

Word embedding loss

Embedding dataset
Joint Embedding Models: RC-Net


**RC Data Model**

- **Categorical Connections**: relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections**: relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

**Joint Loss function**

- Loss function combines word embedding (skip-gram) loss with categorical and relational connections
- Optimization with neural network

**Objective Function**

\[ J = \alpha E_r + \beta E_c - L \]

**Expressiveness**
- Holistic
- Updateable
- RDBMS Integration

<table>
<thead>
<tr>
<th></th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
</tr>
</thead>
</table>

**Relational Data**

- Terry_Gilliam
- Luc_Besson
- Valerian
- 5th_Element

**Relational Expression**

- Minimize \( d(h + r, t) \)

**Categorical Expression**

- Minimize \( \sum_{e_1 \in C} \sum_{e_2 \in C} w \cdot d(e_2, e_2) \)

**Word Embedding Loss**

- Embedding dataset
Joint Embedding Models: RC-Net

**RC Data Model**
- **Categorical Connections**: relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections**: relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

**Joint Loss function**
- Loss function combined word embedding (skip-gram) loss with loss for categorical and relational connections
- Optimization with neural network

**Expressivness** | Holistic | Updateable | RDBMS Integration
---|---|---|---
✓ | ✓ | ✘ |
Joint Embedding Models: RC-Net


**RC Data Model**

- **Categorical Connections**: relations that group entities into categories (e.g. columns in DB system)
- **Relational Connections**: relations between two entities (e.g. entities in the same row, foreign key relations in DB system)

**Joint Loss function**

- Loss function combined word embedding (skip-gram) loss with loss for categorical and relational connections
- Optimization with neural network

<table>
<thead>
<tr>
<th>Expressiveness</th>
<th>Holistic</th>
<th>Updateable</th>
<th>RDBMS Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Objective Function**

$$J = \alpha E_r + \beta E_c - L$$

**No integration** (designed for knowledge graphs)

**Word embedding loss**

$$\min \{ \alpha (r + r', c) - L, \Sigma e_1 \in C \Sigma e_2 \in C w \cdot d(e_2, e_2) \}$$

**Integration**

- ✓
- ✓
- ✘
- ✘
Retrofitting Word Vectors


**Retrofitting**

Based on an **objective function** pre-trained **word embeddings** are **re-fined** using **structured data**

![Diagram](attachment:image.png)
Retrofitting Word Vectors


Retrofitting

Based on an **objective function** pre-trained word embeddings are re-fined using structured data.

**Objective Function**

- Retrofitting \([FDJ+14]\)
- Functional Retrofitting \([LMP18]\)
- Counterfitting \([MST+16]\)
- Post-Processing \([dLCE^{NiWV16}]\)

**Relational Data**

- Terry_Gilliam
- Luc_Besson
- Brazil
- Valerian
- 5th_Element

**Embedding dataset**

---

Retrofitting Word Vectors


**Retrofitting**

Based on an **objective function** pre-trained **word embeddings** are re-fined using **structured data**

**Objective Function**

- Minimize convex function $J$ by an iterative algorithm

![Diagram of retrofitting process](image-url)

Objective Function

$$J = \sum_{i=1}^{n} \left[ \alpha \| v_i - v_i' \| + \sum_{(v_i,v_j) \in E} \beta \| v_i - v_j \| \right]$$

- Pre-trained Vector
- ○ Retrofitted Vector

Retrofitted embedding dataset
Retrofitting Word Vectors


Retrofitting

Based on an objective function pre-trained word embeddings are re-fined using structured data

Objective Function

- Minimize convex function $J$ by an iterative algorithm
- Treat embeddings:
  - similar to pre-trained embeddings

$$J = \sum_{i=1}^{n} \alpha \|v_i - v_i'\| + \sum_{(v_i, v_j) \in E} \beta \|v_i - v_j\|$$

Retrofitted embedding dataset

Retrofitting Word Vectors


Retrofitting

Based on an **objective function** pre-trained **word embeddings** are **re-fined** using **structured data**

**Objective Function**

- Minimize convex function $J$ by an iterative algorithm
- Treat embeddings:
  - similar to pre-trained embeddings
  - similar to embeddings of related text values

$$J = \sum_{i=1}^{n} \alpha \|v_i - v_i'\| + \sum_{(v_i, v_j) \in E} \beta \|v_i - v_j\|$$

Pre-trained Vector

Retrofitted Vector

Retrofitted embedding dataset
Retrofitting Word Vectors

Based on an **objective function** pre-trained **word embeddings** are **re-fined** using **structured** data

**Objective Function**

- Minimize convex function $J$ by an iterative algorithm
- Treat embeddings:
  - similar to pre-trained embeddings
  - similar to embeddings of related text values
- Either connected or not
- Strong coupling to syntax

Expressivness | Static | Updateable | RDBMS Integration
---|---|---|---
X | | |
Retrofitting Word Vectors

Based on an objective function pre-trained word embeddings are re-fined using structured data

Objective Function

- Minimize convex function $J$ by an iterative algorithm
- Treat embeddings:
  - similar to pre-trained embeddings
  - similar to embeddings of related text values

Combines global textual and local relational knowledge

<table>
<thead>
<tr>
<th>Expressivness</th>
<th>Holistic</th>
<th>Updateable</th>
<th>RDBMS Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Retrofitted embedding dataset

Retrofitting Word Vectors

Based on an **objective function** pre-trained word embeddings are re-fined using structured data.

**Objective Function**

- Minimize convex function $J$ by an iterative algorithm.
- Treat embeddings:
  - similar to pre-trained embeddings
  - similar to embeddings of related text values

\[
J = \sum_{i=1}^{n} \left( \alpha \| v_i - v'_i \| \right) + \sum_{(v_i, v_j) \in E} \beta \| v_i - v_j \|
\]

- Relatively fast training
- Online updates could be implemented

<table>
<thead>
<tr>
<th>Expressivness</th>
<th>Holistic</th>
<th>Updateable</th>
<th>RDBMS Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✓</td>
<td>(✓)</td>
<td></td>
</tr>
</tbody>
</table>

Retrofitted embedding dataset

Retrofitting Word Vectors


**Retrofitting**

Based on an **objective function** pre-trained **word embeddings** are **re-fined** using **structured data**

**Objective Function**

- Minimize convex function $J$ by an iterative algorithm
- Treat embeddings:
  - similar to pre-trained embeddings
  - similar to embeddings of related text values

<table>
<thead>
<tr>
<th>Expressiveness</th>
<th>Holistic</th>
<th>Updateable</th>
<th>RDBMS Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

No RDBMS integration

Retrofitted embedding dataset
**Database Retrofitting**

Retrofitting specialized for **database** systems

---

**Objective Function**

---

**Publications**

- [GTL20] EDBT'20
- [GOTL20] CIKM'20

---

---
RETRO: Relation Retrofitting

Database Retrofitting
Retrofitting specialized for database systems

Objective Function
- Minimize convex function $\Psi(W)$ by an iterative algorithm that updates the matrix $W$
- Distinct embeddings for same text value in multiple columns

Objective Function

$$\Psi(W) = \sum_{i=1}^{n} \left[ \alpha_i \|v_i - v_i'\|^2 + \beta_i \Psi_C(v_i, W) + \Psi_R(v_i, W) \right]$$

RETROfitted embedding dataset

Publications
- [GTL20] EDBT'20
- [GOTL20] CIKM'20

RETRO: Relation Retrofitting

Database Retrofitting

Retrofitting specialized for database systems

Objective Function

- Minimize convex function $\Psi(W)$ by an iterative algorithm that updates the matrix $W$
- Distinct embeddings for same text value in multiple columns
- Treat embeddings:
  - similar to pre-trained ones

$$\Psi(W) = \sum_{i=1}^{n} [\alpha_i \|v_i - v'_i\|^2 + \beta_i \Psi_C(v_i, W) + \Psi_R(v_i, W)]$$

Publications

- [GTL20] EDBT'20
- [GOTL20] CIKM'20
### RETRO: Relation Retrofitting


#### Database Retrofitting

Retrofitting specialized for database systems

#### Objective Function

- **Minimize** convex function $\Psi(W)$ by an iterative algorithm that updates the matrix $W$
- **Distinct** embeddings for same text value in multiple columns
- Treat embeddings:
  - similar to pre-trained ones
  - in a column to be similar

$$\Psi(W) = \sum_{i=1}^{n} [\alpha_i \|v_i - v'_i\|^2 + \beta_i \Psi_C(v_i, W) + \Psi_R(v_i, W)]$$
RETRO: Relation Retrofitting


Database Retrofitting

Retrofitting specialized for database systems

Objective Function

- Minimize convex function $\Psi(W)$ by an iterative algorithm that updates the matrix $W$
- Distinct embeddings for same text value in multiple columns
- Treat embeddings:
  - similar to pre-trained ones
  - in a column to be similar
  - similar to embeddings of related text values

\[
\Psi(W) = \sum_{i=1}^{n} [\alpha_i \|v_i - v'_i\|^2 + \beta_i \Psi_C(v_i, W) + \Psi_R(v_i, W)]
\]

Publications

- [GTL20] EDBT'20
- [GOTL20] CIKM'20
RETRO: Relation Retrofitting


Database Retrofitting

Retrofitting specialized for database systems

Objective Function

- Minimize convex function $\Psi(W)$ by an iterative algorithm that updates the matrix $W$
- Distinct embeddings for same text value in multiple columns
- Treat embeddings:
  - similar to pre-trained ones
  - in a column to be similar
  - similar to embeddings of related text values

RDBMS Integration

- Automatic relation extraction
- Extra algorithm for Online updates

RETRO: Relation Retrofitting

**RETRO: Relation Retrofitting**


**Database Retrofitting**

- Minimization of objective function through an iterative algorithm.
- Distinct embeddings for the same text values in multiple columns.
- Treatment of embeddings:
  - similar to pre-trained ones in a column.
  - similar to embeddings of related text values.

**RDBMS Integration**

- Automatic relation extraction.
- Extra algorithm for online updates.

### Objective Function

\[
\Psi(W) = \sum_{i=1}^{n}[\alpha_i \|v_i - v'_i\|^2 + \beta_i \Psi_C(v_i, W) + \Psi_R(v_i, W)]
\]

**Evaluation Results on ML Task**

- Pre-trained Embeddings
- Original Retrofitting
- Node Embeddings

**Embeddings**

- Luc_Besson
- 5th_Element
- Brazil
- Valerian

**Relational Data**

- Terry_Gilliam
- Luc_Besson
- Valerian
- 5th_Element

**Publications**

- [GTL20] EDBT’20
- [GOTL20] CIKM’20
## Context-Aware Adaptation - Conclusion

<table>
<thead>
<tr>
<th>Approach</th>
<th>Expressiveness</th>
<th>Holistic</th>
<th>Updateable</th>
<th>RDBMS Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC-Net</td>
<td>✓</td>
<td>✓</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Retrofitting</td>
<td>✘</td>
<td>✓</td>
<td>(✓)</td>
<td>✘</td>
</tr>
<tr>
<td>RETRO Framework</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Objectives

1. Data Management

2. Context-Aware Adaptation

3. Representation of Schema Information

Applications of Word Embeddings in Database Systems
Representation of Schema Information in DB Systems

Limitations of Word Embeddings for Tables

<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
<th>release_date</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Element</td>
<td>Luc Besson</td>
<td>7 May 1997</td>
<td>...</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
<td>20 Feb 1985</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
<td>17 Jul 2017</td>
<td>...</td>
</tr>
</tbody>
</table>
Representation of Schema Information in DB Systems

Limitations of Word Embeddings for Tables

- **Word embeddings** represent **tokens** → **text values** in DB are more **diverse**

<table>
<thead>
<tr>
<th>director</th>
<th>release_date</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Besson</td>
<td>7 May 1997</td>
<td>...</td>
</tr>
<tr>
<td>Gilliam</td>
<td>20 Feb 1985</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Besson</td>
<td>17 Jul 2017</td>
<td>...</td>
</tr>
</tbody>
</table>

**More diverse text values**
- Multi words
- Titles
- Abbreviations
- Dates

- Short texts (e.g. comments)
### Representation of Schema Information in DB Systems

#### Limitations of Word Embeddings for Tables

- **Word embeddings** represent tokens → text values in DB are more diverse
- No separation between schema and instance data → instance-of, role-play relations are poorly represented

<table>
<thead>
<tr>
<th>Director</th>
<th>Release Date</th>
<th>Instance Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luc Besson</td>
<td>7 May 1997</td>
<td>Invasion of the ...</td>
</tr>
<tr>
<td>Terry Gilliam</td>
<td>20 Feb 1985</td>
<td>Brazil</td>
</tr>
<tr>
<td>Luc Besson</td>
<td>17 Jul 2017</td>
<td>Valerian</td>
</tr>
</tbody>
</table>

---

**More diverse text values**
- Titles
- Multi words
- Abbreviations
- Short texts (e.g. comments)
- Dates

**Schema and Instance Values**

Observation in popular Word Embedding Dataset (Google News word2vec):

\[
\text{sim}_\text{cos}(\text{Angela\_Merkel, moon}) > \text{sim}_\text{cos}(\text{Angela\_Merkel, person})
\]
Representation of Schema Information in DB Systems

Limitations of Word Embeddings for Tables

- Word embeddings represent tokens → text values in DB are more diverse
- No separation between schema and instance data → instance-of, role-play relations are poorly represented

Model to pre-train embeddings on tables to improve table understanding

More diverse text values
- Titles
- Multi words
- Abbreviations
- Short texts (e.g. comments)
- Dates

Schema and Instance Values
Observation in popular Word Embedding Dataset: (Google News word2vec)

\[ \text{sim}_{\cos}(\text{Angela Mekel}, \text{moon}) > \text{sim}_{\cos}(\text{Angela Mekel}, \text{person}) \]
Representation of Schema Information in DB Systems

Table corpus → Training → Embedding Model

<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Element</td>
<td>Luc Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
</tr>
</tbody>
</table>

**Embeddings**

<table>
<thead>
<tr>
<th>movie</th>
<th>Brazil</th>
</tr>
</thead>
<tbody>
<tr>
<td>director</td>
<td>Terry Gilliam</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Representation of Schema Information in DB Systems

Schema-Aware

- Capability to model **category information**
- Recognize **instance-of** and **role-play relations**
- Differentiate between schema and instance text values
Representation of Schema Information in DB Systems

**Schema-Aware**
- Capability to model **category information**
- Recognize **instance-of and role-play relations**
- Differentiate between **schema and instance text values**

**Tabular Input**
- Allow training on **text values in a structured schema** (e.g. Web tables, CSV files)

---

**Table corpus**

<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Element</td>
<td>Luc Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
</tr>
</tbody>
</table>

**Embeddings**

<table>
<thead>
<tr>
<th>movie</th>
<th>Brazil</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Training**

**Embedding Model**
Representation of Schema Information in DB Systems

**Schema-Aware**
- Capability to model category information
- Recognize instance-of and role-play relations
- Differentiate between schema and instance text values

**Tabular Input**
- Allow training on text values in a structured schema (e.g. Web tables, CSV files)

**Flexibility**
- Model any terms (e.g. out-of-vocabulary terms) → Model is useful for tables not in the corpus
- Capable to model any text values on cell-level in databases (e.g. single and multi words)
EmbDI: Embeddings for Relational Datasets


<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Element</td>
<td>Luc Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
</tr>
</tbody>
</table>
EmbDI: Embeddings for Relational Datasets

Construction of Tripartite Graph with Nodes for:
- Records (tuples)
- Text values
- Attributes (columns)

EmbDI: Embeddings for Relational Datasets

Construction of Tripartite Graph with Nodes for:
- Records (tuples)
- Text values
- Attributes (columns)

Construction of Sequences
- Random walks
- For each node the same number of walks

EmbDI: Embeddings for Relational Datasets

Construction of Tripartite Graph with Nodes for:
- Records (tuples)
- Text values
- Attributes (columns)

Construction of Sequences
- Random walks
- For each node the same number of walks

Apply Word Embedding Model
- Multiple embedding models applicable

EmbDI: Embeddings for Relational Datasets

Construction of Tripartite Graph with Nodes for:
- Records (tuples)
- Text values
- Attributes (columns)

Construction of Sequences:
- Random walks
  - For each node the same number of walks

Apply Word Embedding Model:
- Multiple embedding models applicable

Embeddings for tuples and columns
- No textual bonding, e.g. no relation between attributes with same name

Schema-Aware Tabular Input Flexible

[✓]


Tripartite Graph

Random Walks

Training

Embedding Model
EmbDI: Embeddings for Relational Datasets


Construction of Tripartite Graph with Nodes for:
- Records (tuples)
- Text values
- Attributes (columns)

Construction of Sequences
- Random walks
- For each node the same number of walks

Apply Word Embedding Model
- Multiple embedding models applicable

Trainable on any kind of tabular data

<table>
<thead>
<tr>
<th>Schema-Aware</th>
<th>Tabular Input</th>
<th>Flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Random Walks
- $r_1$, $r_2$, ..., $r_n$
EmbDI: Embeddings for Relational Datasets


Construction of Tripartite Graph with Nodes for:

- Records (tuples)
- Text values
- Attributes (columns)

Construction of Sequences

- Random walks
- For each node the same number of walks

Apply Word Embedding Model

- Multiple embedding models applicable

<table>
<thead>
<tr>
<th>Schema-Aware</th>
<th>Tabular Input</th>
<th>Flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>

Model is specific for the database it is trained on.

Training → Embedding Model
Web Table Embeddings


<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Element</td>
<td>Luc Besson</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
</tr>
</tbody>
</table>
# Web Table Embeddings


<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Element</td>
<td>Luc Besson</td>
</tr>
<tr>
<td>Terry Gilliam</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
</tr>
</tbody>
</table>

- **Web Table Embeddings**
  - TabVec$^{[GGS18]}$
  - Table2Vec$^{[ZZB19]}$
  - EM Model$^{[GRE+17]}$
  - TaBERT$^{[YNYR20]}$
Web Table Embeddings

[“Entity Matching on Web Tables: a Table Embeddings approach for Blocking.” EDBT. 2017.]

Preprocessing of Table

- Headers and instance values are marked
- Normalization based on the detected data type, e.g. replace numbers with “$” sign

<table>
<thead>
<tr>
<th>h:movie</th>
<th>h:director</th>
</tr>
</thead>
<tbody>
<tr>
<td>v:5th_Element</td>
<td>v:Luc_Besson</td>
</tr>
<tr>
<td>v:Brazil</td>
<td>v:Terry_Gilliam</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>v:Valerian</td>
<td>v:Luc_Besson</td>
</tr>
</tbody>
</table>

Preprocessed
Web Table Embeddings


Preprocessing of Table
- Headers and instance values are marked
- Normalization based on the detected data type, e.g. replace numbers with "$" sign

Construct Sequences from Web Tables
- Attributes model: header row

![Preprocessed Table Image](image-url)
Web Table Embeddings


Preprocessing of Table
- Headers and instance values are marked
- Normalization based on the detected data type, e.g. replace numbers with „$“ sign

Construct Sequences from Web Tables
- Attributes model: header row
- Entities model: subject column
Web Table Embeddings

Preprocessing of Table
- Headers and instance values are marked
- Normalization based on the detected data type, e.g. replace numbers with "$" sign

Construct Sequences from Web Tables
- Attributes model: header row
- Entities model: subject column
- Triples: sequence of <entity, header, value> triples

Web Table Embeddings

Preprocessing of Table

- Headers and instance values are marked
- Normalization based on the detected data type, e.g. replace numbers with "$" sign

Construct Sequences from Web Tables

- Attributes model: header row
- Entities model: subject column
- Triples: sequence of <entity, header, value> triples

[Image of preprocessed table]

- h:movie
- v:5th_Element
- v:Brazil
- v:Valerian

- h:director
- v:Luc_Besson
- v:Terry_Gilliam
- v:Brazil...

- v:5th_Element
- v:Luc_Besson
- v:Brazil...

[Diagram of training and embedding model]

Web Table Embeddings

Preprocessing of Table
- Headers and instance values are marked
- Normalization based on the detected data type, e.g. replace numbers with "$" sign

Construct Sequences from Web Tables
- Attributes model: header row
- Entities model: subject column
- Triples: sequence of <entity, header, value> triples
- Embeddings for values and headers
- However, sequences might be improper for tabular data

Schema-Aware Tabular Input | Flexible
---|---
✓ | 

![Diagram showing preprocessing and construction of sequences from web tables]

Web Table Embeddings


Preprocessing of Table

- Headers and instance values are marked
- Normalization based on the detected data type, e.g. replace numbers with "$" sign

Construct Sequences from Web Tables

- Attributes model: header row
- Entities model: subject column
- Triples: sequence of <entity, header, value> triple

Schema-Aware | Tabular Input | Flexible
---|---|---
✓ | ✓ | -

Trainable on any kind of tabular data
Web Table Embeddings


Preprocessing of Table
- Headers and instance values are marked
- Normalization based on the detected data type, e.g. replace numbers with "$" sign

Construct Sequences from Web Tables
- Attributes model: header row
- Entities model: subject column
- Triples: sequence of <entity, header, value> triples

<table>
<thead>
<tr>
<th>Schema-Aware</th>
<th>Tabular Input</th>
<th>Flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✘</td>
</tr>
</tbody>
</table>
### Pre-trained Web Table Embeddings

<table>
<thead>
<tr>
<th>movie</th>
<th>director</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Element</td>
<td>Luc Besson</td>
<td>...</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry Gilliam</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc Besson</td>
<td>...</td>
</tr>
</tbody>
</table>

**Publications**

<table>
<thead>
<tr>
<th>ECIR'21</th>
<th>(submitted)</th>
</tr>
</thead>
</table>
Pre-trained Web Table Embeddings

Preprocessing of Table

- **Normalization** based on the detected data type, e.g. replace numbers with "$" sign
- **Headers** and **instance values** are **encoded** with **different alphabet**

<table>
<thead>
<tr>
<th>Μονοε</th>
<th>Διρεκτορ</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th_Element</td>
<td>Luc_Besson</td>
<td>...</td>
</tr>
<tr>
<td>Brazil</td>
<td>Terry_Gilliam</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Valerian</td>
<td>Luc_Besson</td>
<td>Brazil</td>
</tr>
</tbody>
</table>

Preprocessed

Publications

<table>
<thead>
<tr>
<th>ECIR'21</th>
</tr>
</thead>
<tbody>
<tr>
<td>(submitted)</td>
</tr>
</tbody>
</table>
Pre-trained Web Table Embeddings

Preprocessing of Table
- **Normalization** based on the detected data type, e.g. replace numbers with "$" sign
- **Headers** and **instance** values are **encoded** with different alphabet

Construct Bipartite Graph
- Process all tables to create **graph of header ↔ instance data** relations
Pre-trained Web Table Embeddings

Preprocessing of Table

- **Normalization** based on the detected data type, e.g. replace numbers with "$" sign
- **Headers** and instance values are encoded with different alphabet

Construct Bipartite Graph

- Process all tables to create graph of header ↔ instance data relations

Random Sequences

- Construct random walks or other types of sequences on graph
Pre-trained Web Table Embeddings

**Preprocessing of Table**
- **Normalization** based on the detected data type, e.g. replace numbers with "$" sign
- **Headers** and **instance** values are **encoded** with different alphabet

**Construct Bipartite Graph**
- Process all tables to create graph of **header** ←** instance data** relations

**Random Sequences**
- Construct **random walks** or other types of **sequences** on graph

**Apply fastText Model**
- Generate **n-gram embedding** model
Pre-trained Web Table Embeddings

Preprocessing of Table
- **Normalization** based on the detected data type, e.g. replace numbers with "$" sign
- **Headers** and instance values are encoded with different alphabet

Construct Bipartite Graph
- Process all tables to create graph of header ↔ instance data relations

Random Sequences
- Construct random walks or other types of sequences on graph

Apply fastText Model
- Generate n-gram embedding model
Pre-trained Web Table Embeddings

Evaluation Results on Layout Identification Task

- DWTC: Baseline
- F: fastText Word Embedding Model
- DWTC+W: Ensemble Model

Attributes
- μοιε
- διρεκτορ

Random Sequences
- 5th_Element
- Brazil
- Valerian
- Terry_Gilliam
- Luc_Besson

Publications
- ECIR'21 (submitted)
### Representation of Schema Information - Conclusion

<table>
<thead>
<tr>
<th>Approach</th>
<th>Schema-Aware</th>
<th>Tabular Input</th>
<th>Flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmbDI</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
</tr>
<tr>
<td>EM Web Table Embeddings</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
</tr>
<tr>
<td>fastText Web Table Embeddings</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Conclusion

1. Data Management

2. Context-Aware Adaptation

3. Representation of Schema Information

Applications of Word Embeddings in Database Systems
Conclusion

1. Data Management

2. Context-Aware Adaptation

3. Representation of Schema Information

Applications of Word Embeddings in Database Systems
Conclusion

1. Data Management

- Storage
- WE Operations
- Efficiency
- Adaptivity

2. Context-Aware Adaptation

- Expressiveness
- Holistic
- Updateable
- RDBMS Integration

3. Representation of Schema Information

Applications of Word Embeddings in Database Systems
Conclusion

1. Data Management

Storage | WE Operations | Efficiency | Adativity

2. Context-Aware Adaptation

Expressiveness | Holistic
Updateable | RDBMS Integration

3. Representation of Schema Information

Schema-Aware | Tabular Input
Flexible

Applications of Word Embeddings in Database Systems
1. Data Management

2. Context-Aware Adaptation

3. Representation of Schema Information

Applications of Word Embeddings in Database Systems
Conclusion - Publications

1. Data Management
   - Storage
   - WE Operations
   - Adaptivity

   - FREDDY (SIGMOD'18)
   - kNN-Joins + Demo (BTW'19)
   - FacetE (BTW'19)
   - DBTest'20

2. Context-Aware Adaptation
   - Expressivness
   - Holistic
   - Updateable
   - RDBMS Integration

3. Representation of Schema Information
   - Schema-Aware
   - Tabular Input
   - Flexible

Applications of Word Embeddings in Database Systems
Conclusion - Publications

1. Data Management
   - Storage
   - WE Operations
   - Adaptness

2. Context-Aware Adaptation
   - Expressiveness
   - Holistic
   - Updateable
   - RDBMS Integration

3. Representation of Schema Information
   - Schema-Aware
   - Tabular Input
   - Flexible

Applications of Word Embeddings in Database Systems

- FREDDY
  - SIGMOD'18
- kNN-Joins + Demo
  - BTW'19
- FacetE
  - DBTest'20

- RETRO + Demo
  - EDBT'20
- RETRO-Online Updates+GNNs
  - CIKM'20

- Web Table Embeddings
  - Under Review
  - ECIR'21
Conclusion – Future Work

Open Research Aspects

- Integrate RETRO Online-Updates into RDBMS
- Further investigate Web Table Embeddings and their Applications

Thesis Writing Process

- Submission: September 2021
- Defense: November 2021

Applications of Word Embeddings in Database Systems
Thank you for your Attention! – Question Round

1. Data Management
   - Storage
   - WE Operations
   - Adaptivity

   FREDDY
   SIGMOD’18

   kNN-Joins + Demo
   BTW’19

   FacetE
   DBTest’20

2. Context-Aware Adaptation
   - Expressivness
   - Holistic
   - Updateable
   - RDBMS Integration

3. Representation of Schema Information
   - Schema-Aware
   - Tabular Input
   - Flexible

   RETRO + Demo
   EDBT’20
   RETRO-Online Updates+GNNs
   CIKM’20

Applications of Word Embeddings in Database Systems

Web Table Embeddings
Under Review ECIR’21

Web Table
Embeddings
References


References


References


