Context Management in Database Systems with Word Embeddings

Michael Günther
Introduction

Language learning methods

Word embedding operations

Inception:

\[
\begin{bmatrix}
0.54, -0.71, 0.11, \\
\vdots
\end{bmatrix}
\]

Shutter_Island:

\[
\begin{bmatrix}
0.31, -0.59, -0.08, \\
\vdots
\end{bmatrix}
\]

300 dimensions

3M vectors

Word embedding dataset

Text corpora in natural language
Introduction

Word Embedding Operations within SQL:

```
SELECT m.title AS movie, 
    t.word AS similar_to 
FROM movies AS m, most_similar(m.title, 
    (SELECT title FROM movies)) AS t
```

Execution of most_similar operation:

<table>
<thead>
<tr>
<th>movie</th>
<th>similar_to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>Shutter Island</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Introduction

Contribution of Word Embedding to Database Systems

- Integrate external data sources: unstructured data (text in natural language)
- New operations for unstructured text values in the database
  - Semantic Similarity Operator → similarity matches instead of exact match
  - Analogy Operator: analyze semantic relations between text values

Some Applications

- Support data integration tasks by making textual values semantically comparable
- Data exploration
- Integrate simple information retrieval capabilities with minimal indexing effort

```
SELECT m.title AS movie
t.word AS similar_to
FROM movies AS m, most_similar(m.title, (SELECT title FROM movies)) AS t
```
Word Embeddings

- Mapping: Tokens $\rightarrow$ Vectors
- Vectors model semantic as well as syntactic relations between tokens.
- Distributed Hypothesis: Similar words have a similar distribution in the language (Harris, Z. (1954). “Distributional structure”)

Pre-trained word embedding datasets

- contain usually a few million vectors
- Dimensionality of the vectors: 200-300

Application Areas

- NLP techniques, Sentiment Analysis, Machine Translation, Information Retrieval, Visualizations, ...

Word2Vec Model

Source: https://www.tensorflow.org/tutorials/representation/word2vec
Word Embeddings: Operations

Quantify Similarity

- Cosine similarity between vectors:
  \[ \text{sim}_{\text{cos}}(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} \]

Analogies

- Analogy Queries: \( a - b \approx c - ? \)
  e.g. man – woman ≈ king - ? → queen
- Pair-Direction: \( \text{arg max}_{d \in V \setminus \{a,b,c\}} (\text{sim}_{\text{cos}}(a - b, c - d)) \)
- 3CosAdd:
  \[
  \text{arg max}_{d \in V \setminus \{a,b,c\}} (\text{sim}_{\text{cos}}(d, c) - \text{sim}_{\text{cos}}(d, a)) + \text{sim}_{\text{cos}}(d, b) = \text{arg max}_{d \in V \setminus \{a,b,c\}} \text{sim}_{\text{cos}}(d, c - a + b)
  \]

Relation Plot: man – woman
Source: [https://nlp.stanford.edu/projects/glove/](https://nlp.stanford.edu/projects/glove/)
Last access: 08.03.2018
Topics for Thesis

Integration of Word Embeddings in Relational Database Systems

High Performance Word Embedding Operations

Combine Relational Data and Word Embeddings

Application of Word Embedding Database System: Data Integration and Context Management
System Architecture

Fast word EmbedDings in Database sYstem

Basis

- PostgreSQL database system → Open source, Extensibility

Word Embedding Operations

- Implemented as User-Defined-Functions (UDFs)
  → Can be used in SQL queries
  → Search methods implemented efficiently in C
  → Interfaces implemented in PL/pgSQL

Index structures

- Word embeddings stored in relations
- Additional index data also stored in database relations
- Multiple WE datasets allow domain specific notion of similarity
WE Operations for Database System

Some Example Queries

kNN Queries*

```sql
SELECT m.title, t.term, t.score
FROM movies AS kNN(m.title, 3) AS t
ORDER BY m.title ASC, t.score DESC
→ Godfather | {Scarface, Goodfellas, Untouchables}
```

Analogy Queries

```sql
SELECT analogy_3cosadd(
  'Godfather', 'Francis_Ford_Coppola', m.title)
FROM movies AS m
Inception → Christopher Nolan
```

Similarity Queries

```sql
SELECT keyword
FROM keywords
ORDER BY cosine_similarity('comedy', keyword)
→ comedy, sitcom, dramedy, comic, satire, ...
```

* Function calls simplified
Topics for Thesis

Integration of Word Embeddings in Relational Database Systems

High Performance Word Embedding Operations

Combine Relational Data and Word Embeddings

Application of Word Embedding Database System: Data Integration and Context Management
K Nearest Neighbors

**Definition**

\[ k\text{NN}(x) = \arg \min_{\{p_1, \ldots, p_k\} \in P[k]} \sum_{n=1}^{k} ||x - p_n|| \quad x \in \mathbb{R}^D, P \subset \mathbb{R}^D \]

**Quadratic Euclidian Distances**

- Proportional to cosine distances (inverted cosine similarity):

\[ d_{eukl}(x, y)^2 = 2 \cdot d_{cos}(x, y) \]

**Naïve Algorithm**

- Determine all distances to vectors \( p \in P \)
  \( \rightarrow \) Select vectors with lowest distances (kNN)
- Complexity: \( O(|P| \cdot D) \)
Vector Quantization

**Objective**
- Transform vectorial data into **compact** representation

**Quantization**
- Cluster Data – kMeans
- **Objective**: Obtain a set of centroids which minimize the distortion:
  \[ \text{Minimize } d(y, q(y)) \]

**Quantization Function**
- Def.: \( q: y \in \mathbb{R}^D \rightarrow C, C \subset \mathbb{R}^D \)

Assign every vector to its nearest neighbor out of a set of centroids

Represent vector by centroid id

Inception  Shutter Island  Forrest Gump

\[ q(y) = c_{10} \rightarrow \text{id: 10} \]
Product Quantization

**Idea**
- More accurate quantization by using **multiple** quantizers for **subvectors**
- Compact representation of vectors by **centroid id sequence**
  → **Low computation time** for distances

**Product Quantization**

1. Split vector $\mathbf{y}$ in subvectors $u_1(\mathbf{y}), \ldots, u_m(\mathbf{y})$
2. Apply quantizers $q_1, \ldots, q_m$
   where $q_i : \mathbb{R}^d \rightarrow \{c_1, \ldots, c_k\}$
3. Represent product quantization as 
   id sequence

$$
\mathbf{y} = \{y_1, \ldots, y_d, y_{d+1}, \ldots, y_{2d}, \ldots, y_{(n-d)+1}, \ldots, y_n\}
$$

**Product Quantization:**

$$
qu_1(u_1(\mathbf{y})), q_2(u_2(\mathbf{y})), \ldots, q_m(u_m(\mathbf{y}))
$$

→ **Representation as sequence of centroid ids**

$$
Seq = \{1, \ldots, |C_k|\}^m
$$
Product Quantization Search

Fast Computation of kNN via Pre-Processing

- Approximated Distances: \( \hat{d}(x, y) = \sqrt{\sum_j d(u_j(x), q_j(u_j(y)))^2} \)

- Precompute: \( d \left( u_j(x), q_j(u_j(y)) \right)^2 \)

Query Sub Vector  Centroid of Quantizer

Computation of square distances resolves to a sum of \( m \) precomputed distances
Inverted Indexing

**Idea**
Accelerate kNN-Search by non-exhaustive search schema

**Preprocessing**
Additional Coarse Quantizer $q_c$ is applied to the complete vectors
→ Assigns vectors to partitions

**kNN Calculation**
- Coarse quantization of query vector is calculated → $id(c_1), ..., id(c_\omega)$ ....$\omega$ determine number of candidates
- Partitions with PID $\in \{id(c_1), ..., id(c_\omega)\}$ are retrieved
- Distances Calculation → kNN
SELECT kNN(movies.title, 3)
FROM movies

SELECT m1.title, m2.title
FROM movies m1 kNN-Join(3)movies m2
ON m1.title ~ m2.title
WHERE m1.year = ... AND m2.genre IN (…)

<table>
<thead>
<tr>
<th>m_id</th>
<th>title</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scarface</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>Untouchables</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>Goodfellas</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>Ben Hur</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10^6</td>
<td>Godfather</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>1972</td>
<td>[0.21; 0.58; ...; -0.77]</td>
</tr>
<tr>
<td>Brando</td>
<td>[-0.46; 0.25; ...; 0.44]</td>
</tr>
<tr>
<td>Ben Hur</td>
<td>[0.76; 0.33; ...; 0.91]</td>
</tr>
<tr>
<td>Copolla</td>
<td>[0.76; 0.48; ...; -0.51]</td>
</tr>
<tr>
<td>...</td>
<td>[-0.46; 0.53; ...; 0.85]</td>
</tr>
<tr>
<td>Untouchables</td>
<td>[0.86; -0.22; ...; 0.12]</td>
</tr>
</tbody>
</table>

kNN(R × T) = \{(r, t) | t ∈ kNN(r, T), r ∈ R\}
kNN-Joins in RDBMS

**Challenges**

- **Batch-wise kNN Search** for large query sets → reduce interface and retrieval times
- **High Dimensional Data** → Previous Work mainly focuses on low-dimensional data (e.g. geographical data)
- **Adaptive kNN-Join Algorithm** → Different cardinalities of join operands → Only one index for all vectors
- **Different Demands on Precision and Response Time**

\[
kNN(R \Join T) = \{(r, t) | t \in kNN(r, T), r \in R\}
\]
kNN-Joins in RDBMS – Adaptive Algorithm

(1) Preprocessing
- Precompute distance values
- Adaptive to target set size

(2) Query Construction
- Determine $\omega$ (Number of Part.)
- Estimate number of retrieved entries (Probabilistic Approach)

(3) Data Retrieval
- Retrieve kNN candidates

(4) Distance Calculation
- User can adjust trade-off between precision and response time
Evaluation

Evaluation Setup
- Time and precisions measurements for randomly sampled word vectors

Dataset
- 3M 300-dimensional word vectors

Parameters
- Number of query vectors: 5,000
- Number of target vectors: 50,000
- K: 5
- Selectivity of the filter $\alpha$ (high values are less selective)
- Different sizes of candidate sets for post verification
Current Status

Integrate Word Embedding Operations in PostgreSQL:

Improve Performance for underlying kNN-Join Operations

Additional Grants
- Intel AI Academy Program

Web Demo
- Interactive exploration of word embedding queries, Influence of different Datasets, Influence of approximation on results quality and runtime
Topics for Thesis

- Integration of Word Embeddings in Relational Database Systems
- High Performance Word Embedding Operations
- Combine Relational Data and Word Embeddings
- Application of Word Embedding Database System: Data Integration and Context Management
How to combine both information sources?

Open Challenges

1. Equality Checking
   Determine if database entries are represented in the word embedding dataset

2. Alignment
   Make structured relational knowledge comparable with knowledge encoded in word embeddings

3. Word Embedding Imputation
   Integrate Missing Entities in Word Embedding Dataset
Check if entities are represented in the word embedding dataset

**Strategy**

1. Extract relations between text values
2. Determine which entries belong to a common concept (hyponym hypernym relations)
3. Solve the outlier problem:
   - Which word does not belong to the others?

**Open Questions**

- How to determine the number of outliers?
- How to obtain the relations?
  - Information encoded in ER-Model
  - External domain knowledge
Alignment of both information sources

Objective
- Enrich Word Embeddings by introducing structural relations in the embeddings
- Populate Embeddings: Add vectors for missing text values

Algorithm
1. Extract structural information
2. Init new embeddings with word embeddings
3. Retrofit embeddings with Learning Algorithm

Applications
- Machine learning on databases with complex schemes
- Entity Resolution
Topics for Thesis

Integration of Word Embeddings in Relational Database Systems

High Performance Word Embedding Operations

Combine Relational Data and Word Embeddings

Application of Word Embedding Database System: Data Integration and Context Management
Applications

Entity Resolution
- Find Rows in tables of different schemes which refer to the same entity
  → Word embeddings can be used to identify semantic similar rows based on text values

Identify Semantic Links
- Identify semantic relations in the datasets and between different datasets
Integration of Word Embeddings in Relational Database Systems

High Performance Word Embedding Operations

Combine Relational Data and Word Embeddings

Application of Word Embedding Database System: Data Integration and Context Management