# First TAB - Context Management in Database Systems with Word Embeddings

Michael Günther (Michael.Guenther@tu-dresden.de)

Technische Universität Dresden, Institute of System Architecture, Database Systems Group

## 1 Introduction

In complex adaptive systems data integration plays an important role. Large organizations usually store data in a lot of different databases with different large schemes. Storing the data in one common scheme is due to the segmentation of such organizations in many loosely connected departments often not possible [16]. However, there are situations in which data from different schemes has to be integrated to solve certain tasks. Since the schemes can change independently it is highly desirable to be able to automatically integrate data from different sources. However, such data integration tasks typically require a lot of manual effort [17]. Since the volume of data which has to be managed growth and software systems evolving more frequently, the demand for more automated data integration solutions increases. Furthermore, data integration has to be supported by tools for data discovery and data exploration which allow the user to observe the coherences in the data. A key challenge in data integration is the matching process of data from different sources which describe the same entity called entity resolution. Since ids are usually not consistent across different datasets, entity resolution particularly focuses on the observation of textual values (e.g. labels, descriptions, ...). However, relational database management systems provide only very few possibilities to compare textual values. Furthermore, SQL operations for comparison are not able to conceive their semantic information. Thus, we want to integrate operations in relational database systems to semantically compare text values and observe relations between them which are not already covered by the relational schema.

Recently developed word embedding techniques like word2vec [13] allow to extract semantic information about words from natural language texts and have been shown their usefulness for a wide range of NLP [19, 15] and information retrieval tasks [14]. The basic idea behind the famous word2vec model is that the meaning of a word can be derived from the contexts it occurs in. The model emits vectors for each token in a training corpus which have a high cosine similarity for word vectors of semantic or syntactic similar words. Furthermore, it has been shown, that such word embeddings can be used to solve so-called analogy queries by applying simple algebraic operations [12]. In this way, semantic relationships such as word analogies, gender-inflections, or geographical relationships can be easily recovered.

Embeddings can be trained on texts of a specific domain. In this way, word embeddings provide a deeper understanding of this domains. They can also be facilitated to gather domain information which is useful to integrate different information sources. For instance [7] have shown that taxonomic information can be retrieved and [18] used word embeddings for ontology learning. In the first place they support data integration tasks by making textual values semantically comparable which are not syntactical equal. Moreover, word embedding operations enable capabilities for semantic comparison. This can be utilized for information retrieval and data exploration. Despite this, word embeddings have been shown to be useful for a variety of machine learning task, especially in entity resolution [5] which is one of the major problems in automated data integration.

In this thesis, it should be investigated how word embeddings can be integrated in relational database systems. For this purpose, new operations for unstructured text values should be provided. Furthermore, techniques should be provided to combine the knowledge in the relational database with the information encoded in the word embeddings to enable inference based on combinations of logical and inductive reasoning.

## 2 Problems and Objectives

By now we have identified four major challenges in the process of integrating word embeddings in database systems for context management and data integration. The first challenge is the integration of word embeddings and some basic operations on the vectors in relational database systems. The second challenge is to provide sufficient performance of word embedding operations. It should be possible to apply them on large datasets. Moreover, they should be applicable to real-time applications. In order to apply more complex classification and inference tasks, it should be possible to combine the information encoded in word embeddings with the relational information stored in the database. This should be achieved by encoding both information sources in a common representation which allows more high-level classification tasks. As a fourth challenge, it should be investigated how such word embedding database system could be applied to complex tasks in fields of data integration and context management.

*Integration of Word Embeddings* In order to make the knowledge encoded in word embeddings available to database systems the functionalities of a traditional RDBMS should be extended by operations to calculate the similarity of text values according to the similarity of respective word vectors. The word vector datasets should be maintained in the database system itself. It should be possible to choose between different datasets during runtime in order to change the context the similarity measurement should take into consideration. Furthermore, different word embedding datasets provide the possibility to compare the similarity of text values in different languages. In [11] it has been shown that the meaning of words also differs across different linguistic registers. Words can have different meanings in different domains. Thus, it is suggested to choose a word

embedding dataset according to the domain of the text values which should be compared. In this way similarity queries provided by word embedding operations differ from other approaches where only one specific notion of similarity is implemented. Beside similarity operations word embeddings have been shown to be able to solve analogy queries like the one in Figure 2 where the relationship between the movie "Godfather" and its director "Francis Ford Coppola" is used to derive the director of the movie Inceptions which results in "Christopher Nolan". The system should provide operations to solve such queries.

*Performance* An important subroutine of most of the word embeddings is the *kNN-Search* operation which goal is it to find the $k$ most similar vectors to a given vector. In this way, for instance, a most similar operation can be realized and analogy queries can be solved. A simple kNN query is shown in Figure 2. It determines for every movie in a movie relation the three most similar ones. However, the complexity of this operation is linear in the number of vectors and dimensions. Thus, running the kNN operation on sets of text values becomes unfeasible especially for use cases where real-time query execution is required. The execution of the kNN operation on every element of a set of query vectors resolves to a so-called kNN-Join operation. The challenge is thus to integrate a kNN-Join operation in the database system which is able to execute queries on high dimensional data. The index strategy should allow dynamic updates. Inserts are for instance useful since vectors for text values consisting of multiple tokens can be generated by averaging word vectors [3]. Thus, it is desirable to add such vectors to the word embedding index if they are added to a database table.

*Combination of Relational Data and Word Embeddings* Running SQL Queries with word embedding operations enables the user to combine the knowledge encoded in word embeddings with traditional analytical functions provided by the database. However, there are a few problems involved with this process. One problem is the polysemy of words. Having a token like "Apple" which has multiple senses (fruit vs. company) word embedding queries can deliver wrong results in case the sense of the text value in the database refers to a different meaning than the meaning encoded in the word embedding. Thus, the system has to detect if the senses concur, based on the embedding data and the relational data in the database system. Another problem is that in many cases the word embedding dataset might not contain a vector for every text value in the database. Some named entities might not occur in the text corpus used for training the word embeddings as frequent as necessary to produce high-quality embeddings. In such cases, it might be reasonable to use the relational data and their relation to the word embedding data to find suitable embeddings for missing tokens which could be imputed in the embedding dataset. Moreover, it might be useful to combine word embeddings and relational data to obtain embeddings encoding information of both information sources. We assume there are many possible use cases for embeddings of data from relational databases. Embeddings for relational data could be interesting to perform machine learning on

relational database systems more easily. Most of the machine learning algorithms have problems with the relational structure and presume that the data consists of a single table [4]. Moreover, data is often transformed into a continuous vector representation to run classifiers. Therefore, maintaining embeddings for the relational data would make it a lot easier to train classifiers on the data. While there are plenty of methods developed to train embeddings for graph data [8], there is only limited work done for training embeddings for relational data. One approach from [2] can be used for training embeddings on relational data to provide word embedding operations for text values in the database. In my thesis, an approach should be provided to generate embeddings which combine word embeddings with the knowledge from relational data. Retrofitting approaches like [6] try to adapt word embeddings based on graph data. By [5] this approach has been adapted for tabular data in a way that embeddings could be generated for values where no word embeddings exist. However, relations between tables are not considered. In the context of database systems, it is also interesting how to handle inserts and updates.

*Applications in Data Integration* In order to show the benefits of the approach, we should aim to provide applications in complex data integration scenarios. In such a scenario should be investigated how embeddings help to support data integration and context management. Based on our observation the database integration should be adapted to support those applications.

## 3 Current State of the Thesis

We integrated word embedding operations in a PostgreSQL database system which led to a publication on the SIGMOD 2018 [9]. In addition, we developed an index structure to improve the performance of approximated kNN-Join operations in the database system (not published yet). An overview of this work is given in Section 3.1. Furthermore, we acquired financial support for hardware infrastructure from the Intel AI Academy Program. An interactive web demo provides the possibility to observe word embedding operations. This is described in Section 3.2 One can choose from a selection of different word embedding datasets, run queries on several database schemes and explore how this influences the results. Furthermore, the influence of the approximation techniques on execution time and precision can be observed.

### 3.1 Integration of Word Embeddings in Relational Database Systems

To exhibit the rich information stored in word embeddings and utilize it for use cases in relational database systems we developed a system called FREDDY (Fast woRd EmbedDings Database sYstems) which extends a relational database system based on PostgreSQL. A wide range of User Defined Functions (UDFs) forming the base for novel query types allowing the user to analyze structured

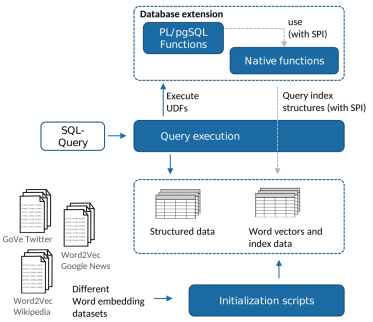



Fig. 1: FREDDY System Overview

```
SELECT m.title, t.term, t.score
FROM movies AS m,
kNN(m.title, 3) AS t
ORDER BY m.title ASC, t.score DESC
```

kNN Query

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

Analogy Query

Fig. 2: Simplified IMDB Query Examples

knowledge in the database relations together with huge, unstructured text corpora encoded as word embeddings. Supported by different index structures and approximation techniques these operations are able to perform fast similarity computations on high-dimensional vector spaces.

*System Architecture* FREDDY's[1] system architecture is sketched in Figure 1. Multiple word embedding datasets can be added to the system. A script creates new relations for index structures which enable fast approximated kNN queries. To make use of these word embeddings within SQL we implemented the following User Defined Functions (UDFs) that operate on the index relations:

- *cosine_similarity(token varchar, token varchar)*: Quantifies the similarity between two tokens
- *kNN(token varchar, k int)*: Searches for the $k$ most similar tokens according to the input
- *kNN_in(token varchar, k int, output_set varchar[])*: like kNN but restricting the result to a defined set of output tokens (e.g. to obtain results corresponding to a column in a database relation)
- *knn_batch(query_set varchar[], k integer)*: kNN function that runs multiple queries in batches (e.g. for JOIN operations based on similarity)
- *analogy(token_1 varchar, token_2 varchar, token_3 varchar)*: Solves analogy queries using the *PairDirection*, *3CosADD* or *3CosMul* method [12]
- *analogy_in(token_1 varchar, token_2 varchar, token_3 varchar, output_set varchar[])* Analogy queries restricting the result set
- *groups(tokens varchar[], groups varchar[])* Assigns input tokens to groups specified by other tokens according to their similarity

Moreover, there are further UDFs which serve as helper functions, e.g. for the calculation of centroids for calculating exact cosine similarity values. Search functions like $kNN$ provide the possibility of an exact computation as well, but can

[1] https://github.com/guenthermi/postgres-word2vec

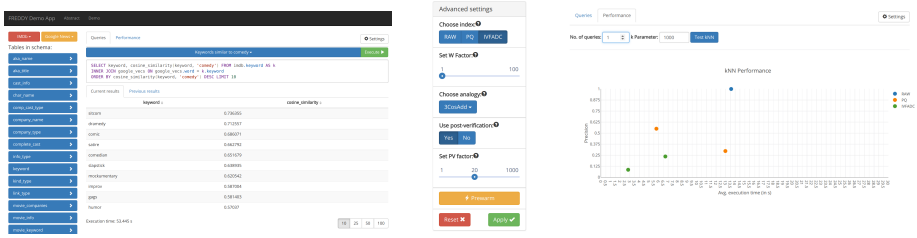
also be performed in an approximated manner to be applied on large input sets and tables. The UDFs for similarity calculations and search operations are implemented in C whereas interfaces are realized via the procedural script language PL/pgSQL. By using the PostgreSQL Server Programming Interface (SPI), the UDFs are able to run SQL commands inside the functions, e.g. to access the word vectors and index structures. All UDFs are bundled into a PostgreSQL extension.

*Performance* K nearest neighbor search (kNN-Search) is a universal data processing technique and a fundamental operation for word embeddings trained by word2vec or related approaches. The benefits of operations on dense vectors like word embeddings for analytical functionalities of RDBMSs motivate an integration of kNN-Joins. However, kNN-Search, as well as kNN-Joins, have barely been integrated into relational database systems so far. We develop an index structure for approximated kNN-Joins working well on high-dimensional data and provide an integration into PostgreSQL.
We identified some criteria which are specific for the application focus and should be fulfilled by the index structure and a respective search join operation:
**Minimization of index accesses:** Since the index data is stored in database relations the index accesses are performed by SQL queries. However performing large amounts of single SQL queries on a relational database systems could slow done the search performance. Thus, the join operation should minimize the amount of such index retrieval queries.
**High-dimensional data:** Word embeddings are vectors with typically more than 100 dimensions. Due to the curse of dimensionality, indexing such vectors is a hard problem. In order to provide sufficient performance, we decided to use approximated search techniques which can cope with this property.
**Adaptive kNN-Join algorithm:** Most of the approximated kNN operations are optimized to deal with a large number of target vectors from which they obtain the $k$ most similar ones. On the contrary, a kNN-Join operation might deal with smaller target sets, however, gets a large number of query vectors as an input. Our kNN-Join operation has to deal with both scenarios. However, it should not solve the problem by creating multiple index structure, since this leads to a higher demand for memory and longer insertion and update time.
**Different demands on precision and response time:** Regarding the approximation of the vector similarity, it might be relevant for a user to specify how much the approximated nearest neighbors should agree with the exact values. On the contrary, real-world systems need to comply certain latency constraints, e.g. for exploratory data processing, fast response times are crucial. Consequently, the approximated kNN-Join should provide features to configure such trade-offs. Providing these tunable trade-offs would also support query execution in an online aggregation manner, i.e. get estimates of a kNN-Join query as soon as the query is issued and steadily refine during its execution.

In order to satisfy the mentioned criteria, we developed an index structure base on product quantization [10] and inverted indexing like it was used in [1].

Several optimizations allow us to combine both techniques to an index structure which is suitable for our use case, whereas we see applications of those index structure beyond their use for word embedding similarity search. An evaluation of the system on different word embedding datasets against a baseline method has shown the benefits of such an integrated kNN-Join operation and the performance of the proposed approach.

### 3.2 Web Demonstration



(a) Query Interface (b) Configuration (c) Search Analysis View

Fig. 3: Demo Interface

A web application makes it possible to explore these novel query capabilities on different database schemes and a variety of word embeddings generated on different text corpora. From a systems perspective, the user is able to examine the impact of multiple approximation techniques and their parameters for similarity search on query execution time and precision. The main views and the configuration element are shown in Figure 3.

## References

1. Babenko, A., Lempitsky, V.: The inverted multi-index. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 3069–3076. IEEE (2012)
2. Bordawekar, R., Shmueli, O.: Enabling cognitive intelligence queries in relational databases using low-dimensional word embeddings. arXiv preprint arXiv:1603.07185 (2016)
3. Campr, M., Ježek, K.: Comparing semantic models for evaluating automatic document summarization. In: International Conference on Text, Speech, and Dialogue. pp. 252–260. Springer (2015)
4. Domingos, P.: Machine learning for data management: Problems and solutions. In: Proceedings of the 2018 International Conference on Management of Data. pp. 629–629. ACM (2018)

5. Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., Tang, N.: Distributed representations of tuples for entity resolution. Proceedings of the VLDB Endowment **11**(11), 1454–1467 (2018)
6. Faruqui, M., Dodge, J., Jauhar, S.K., Dyer, C., Hovy, E., Smith, N.A.: Retrofitting word vectors to semantic lexicons. arXiv preprint arXiv:1411.4166 (2014)
7. Fu, R., Guo, J., Qin, B., Che, W., Wang, H., Liu, T.: Learning semantic hierarchies via word embeddings. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1199–1209 (2014)
8. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems **151**, 78–94 (2018)
9. Günther, M.: Freddy: Fast word embeddings in database systems. In: Proceedings of the 2018 International Conference on Management of Data. pp. 1817–1819. ACM (2018)
10. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence **33**(1), 117–128 (2011)
11. Kutuzov, A., Kuzmenko, E., Marakasova, A.: Exploration of register-dependent lexical semantics using word embeddings. In: Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH). pp. 26–34 (2016)
12. Levy, O., Goldberg, Y.: Linguistic regularities in sparse and explicit word representations (2014)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
14. Nalisnick, E., Mitra, B., Craswell, N., Caruana, R.: Improving document ranking with dual word embeddings. In: Proceedings of the 25th International Conference Companion on World Wide Web. pp. 83–84. International World Wide Web Conferences Steering Committee (2016)
15. Sienčnik, S.K.: Adapting word2vec to named entity recognition. In: Proceedings of the 20th nordic conference of computational linguistics, nodalida 2015, may 11-13, 2015, vilnius, lithuania. pp. 239–243. No. 109, Linköping University Electronic Press (2015)
16. Stonebraker, M., Ilyas, I.F.: Data integration: The current status and the way forward. IEEE Data Eng. Bull. **41**(2), 3–9 (2018)
17. Thirumuruganathan, S., Parambath, S.A.P., Ouzzani, M., Tang, N., Joty, S.: Reuse and adaptation for entity resolution through transfer learning. arXiv preprint arXiv:1809.11084 (2018)
18. Wohlgenannt, G., Minic, F.: Using word2vec to build a simple ontology learning system. In: International Semantic Web Conference (Posters & Demos) (2016)
19. Xue, B., Fu, C., Shaobin, Z.: A study on sentiment computing and classification of sina weibo with word2vec. In: Big Data (BigData Congress), 2014 IEEE International Congress on. pp. 358–363. IEEE (2014)