T. Grust et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2019),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 529

Explore FREDDY: Fast Word Embeddings in Database Systems

Michael Günther[1] Zdravko Yanakiev[1] Maik Thiele[1] Wolfgang Lehner[1]

Abstract: Word embeddings encode a lot of semantic as well as syntactic features and therefore are useful in many tasks especially in Natural Language Processing and Information Retrieval. FREDDY (Fast woRd EmbedDings Database sYstems), an extended PostgreSQL database system, allowing the user to analyze structured knowledge in the database relations together with unstructured text corpora encoded as word embedding by introducing novel operations for similarity calculation and analogy inference. Approximation techniques support these operations to perform fast similarity computations on high-dimensional vector spaces. This demo allows exploring the powerful query capabilities of FREDDY 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.

Keywords: word embeddings, relational database, k nearest neighbor queries

1 Introduction

Word2vec [Mi13], GloVe [PSM14] and fastText [Bo16], all well-known models to produce word embeddings, are powerful techniques to study the syntactic and semantic relations between words by representing them in a low-dimensional vector. By applying algebraic operations on these vectors semantic relationships such as word analogies, gender-inflections, or geographical relationships can be easily recovered [LG14]. Word embeddings are useful in many tasks in Natural Language Processing and Information Retrieval, such as text mining and classification, sentiment analysis, sentence completion, or dictionary construction. Some recent papers also showed the potential of word embeddings to enable AI capabilities in relational databases [BBS17] and data curation tasks [TTO18].
The goal of the demo is to show how word embeddings could be utilized to augment and enrich the SQL query capabilities, e.g. to compare columns according to their semantic relation or to group rows according to the similarity. For this purpose, we implemented a web application for FREDDY [Gü18, GTL19]. We use pre-trained word embedding models of large text corpora such as Wikipedia but also domain-specific word embeddings that we trained ourselves. By exploiting this external knowledge during query processing we are able to apply inductive reasoning on text values stored in database relations. This is done by

[1] Technische Universität Dresden, Institute of Systems Architecture, Dresden Database Systems Group, Nöthnitzer Straße 46, 01187 Dresden, firstname.lastname@tu-dresden.de

doi:10.18420/btw2019-38

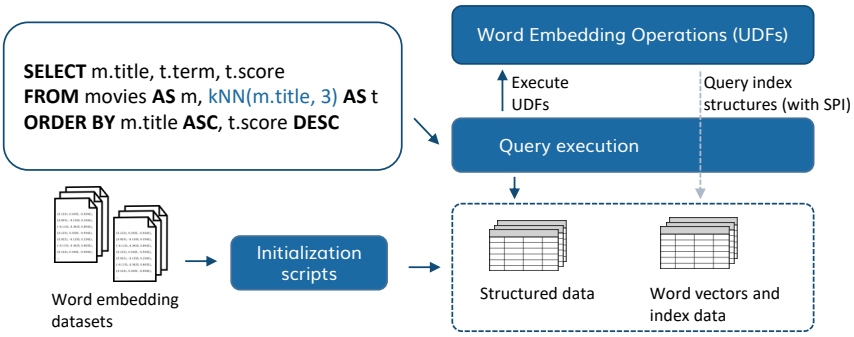


Fig. 1: System Overview

extending the standard SQL syntax of a PostgreSQL database system with novel functions that utilize the word embedding datasets to perform semantic similarity calculations. In this way, it is possible to search for the most similar terms, to group them into categories or to compare semantic relations between terms among each other, e.g. to answer so-called analogy queries. In the context of the IMDB[3], FREDDY enables novel query types such as shown in Figure 1, a query that returns the top-3 nearest neighbors (3NN) of each movie in IMDB. Given the movie "Godfather" as input this results in "Scarface", "Goodfellas" and "Untouchables". FREDDY also supports semantical grouping, e.g. assigning movies to genres such as *film noir* or *road movie*.
Since most word embedding operations perform *k nearest neighbor search (kNN)* they easily can become a bottleneck for cases where the kNN search is done hundreds of thousands times within an SQL query. Therefore, FREDDY provides fast approximation techniques that exploit the trade-off between execution time and precision.

2 FREDDY: System Overview

FREDDY's system architecture is sketched in Figure 1. For word embeddings datasets, which should be added, a script creates new relations for the different index structures. For exact distance computations, an additional relation is created storing terms and the respective normalized wordvec vector. In order to make use of these word embeddings within SQL we implemented the User Defined Functions (UDFs) in Figure 2 which operate on the index relations. 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 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.

[3] `https://www.imdb.com`

cosine_similarity(token varchar, token varchar): Quantifies the similarity between two tokens

analogy(token_1 varchar, token_2 varchar, token_3 varchar): Solves analogy queries using the *PairDirection*, *3CosADD* or *3CosMul* method [LG14]

analogy_in(token_1 varchar, token_2 varchar, token_3 varchar, output_set varchar[]): Analogy queries with restricted result set

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)

groups(tokens varchar[], groups varchar[]): Assigns input tokens to groups specified by other tokens according to their similarity

Fig. 2: Word Embedding Operations

3 Demonstration Outline

For this demonstration, we provide a web-client, different database schemas, various word embeddings and a selection of pre-defined queries[4]. Participants are also able to create their own queries. To gain detailed insights on how the different index structures and search functions perform and how their parameters affect result quality and query performance, we provide several widgets to select and adjust them.

Query Interface View The user can choose between different database schemes (1): IMDB, DBLP[5] and Discogs[6] music data. In addition, there is a drop-down menu to select different word embedding datasets from a selection including word2vec vectors trained on Goole News articles and Wikipedia as well as vectors trained with GloVe[PSM14] on Common Crawl[7] data. Executing the same query multiple times using different word embeddings leads to different result sets. In general, it is recommended to choose a word embedding dataset which is trained on a related topic according to the database schema. In the text field at (2), the query can be created manually or the users get inspired by one of the pre-defined example queries provided by the drop-down menu above. If a query is executed its result and the response time appears at (3). The demonstrator also keeps track of the previous query and its result. They can be retrieved and compared with the current ones using the tab menu above the result table. In a sidebar (see Figure 3b) the users can choose between the different index structures for similarity search and different analogy query types.

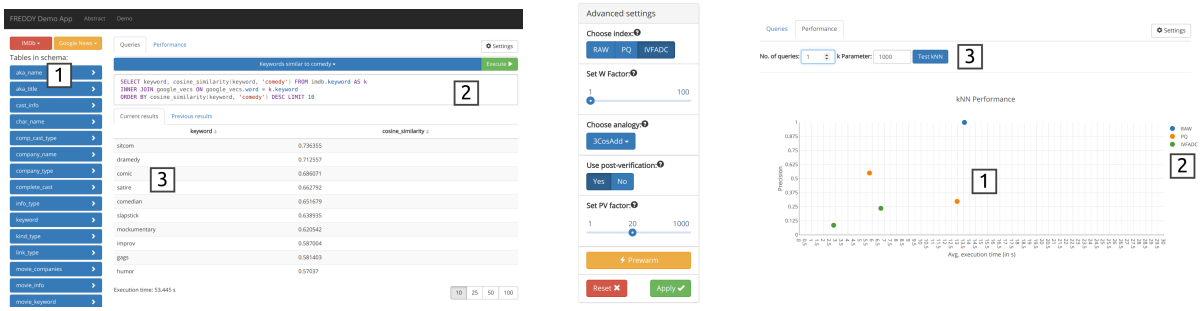
Performance View In a second view illustrated in Figure 3c, the demo user can perform time and precision measurements for kNN and analogy queries using different configurations and compare the results by employing different plots (1). Different visual features (e.g. color, size, ...) encode the index and search parameters. The notations are declared in the legend at (2). To obtain reliable measurements the queries are executed multiple times and the average values for the response time and the precision are obtained. At (3) the number of queries that should be executed and the neighborhood k are specified. The configuration of the search function is defined in the sidebar (Figure 3b) just as in the query view.

[4] Screencast on our FREDDY website `https://wwwdb.inf.tu-dresden.de/research-projects/freddy/`

[5] `https://dblp.uni-trier.de`

[6] `https://www.discogs.com`

[7] `http://commoncrawl.org/`



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

Fig. 3: The web-interface of FREDDY

Acknowledgments

This work is funded by the German Research Foundation (DFG) within the Research Training Group "Role-based Software Infrastructures for continuous-context-sensitive Systems" (GRK 1907) and by the Intel® AI Research.

References

[BBS17] Bordawekar, Rajesh; Bandyopadhyay, Bortik; Shmueli, Oded: Cognitive Database: A Step towards Endowing Relational Databases with Artificial Intelligence Capabilities. CoRR, abs/1712.07199, 2017.

[Bo16] Bojanowski, Piotr; Grave, Edouard; Joulin, Armand; Mikolov, Tomas: Enriching Word Vectors with Subword Information. CoRR, abs/1607.04606, 2016.

[GTL19] Günther, Michael; Thiele, Maik; Lehner, Wolfgang: Fast Approximated Nearest Neighbor Joins For Relational Database Systems. Datenbanksysteme für Business, Technologie und Web (BTW 2019), 2019.

[Gü18] Günther, Michael: FREDDY: Fast Word Embeddings in Database Systems. In: Proc. of the 2018 International Conference on Management of Data. ACM, pp. 1817–1819, 2018.

[LG14] Levy, Omer; Goldberg, Yoav: Linguistic regularities in sparse and explicit word representations. In: Proc. of the 18th conference on computational natural language learning. pp. 171–180, 2014.

[Mi13] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S; Dean, Jeff: Distributed Representations of Words and Phrases and their Compositionality. In: Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc., 2013.

[PSM14] Pennington, Jeffrey; Socher, Richard; Manning, Christopher D.: GloVe: Global Vectors for Word Representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543, 2014.

[TTO18] Thirumuruganathan, Saravanan; Tang, Nan; Ouzzani, Mourad: Data Curation with Deep Learning [Vision]: Towards Self Driving Data Curation. CoRR, abs/1803.01384, 2018.