RetroLive: Analysis of Relational Retrofitted Word Embeddings

Michael Günther
Database Systems Group, Technische Universität Dresden
Dresden, Germany
Michael.Guenther@tu-dresden.de

Maik Thiele
Database Systems Group, Technische Universität Dresden
Dresden, Germany
Maik.Thiele@tu-dresden.de

Erik Nikulski
Database Systems Group, Technische Universität Dresden
Dresden, Germany
Erik.Nikulski@tu-dresden.de

Wolfgang Lehner
Database Systems Group, Technische Universität Dresden
Dresden, Germany
Wolfgang.Lehner@tu-dresden.de

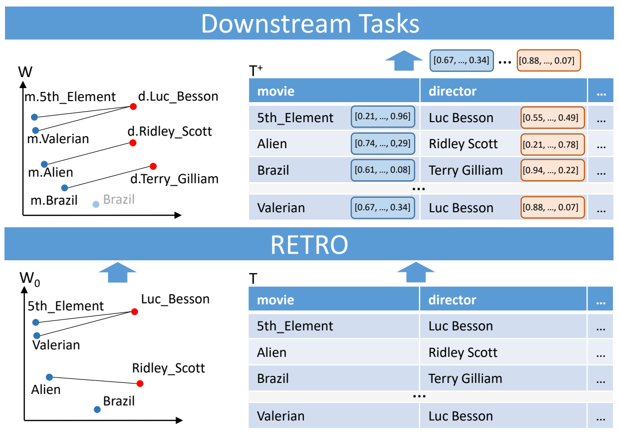
ABSTRACT

Text values are valuable information in relational database systems for analysis and machine learning (ML) tasks. Since ML techniques depend on numerical input representations, word embeddings are increasingly utilized to convert symbolic representations such as text into meaningful numbers. However, those models do not incorporate the context-specific semantics of text values in the database. To significantly improve the representation of text values occurring in DBMS, we propose a novel retrofitting approach called Retro which considers both, the semantics of the word embedding and the relational schema. Based on this, we developed RetroLive, an interactive system, that allows exploring how the retrofitted embeddings improve the performance for various ML and integration tasks. Moreover, the demo includes several interactive visualizations to explore the characteristics of the adapted vectors and their connection to the relational database.

1 INTRODUCTION

Due to their appealing properties, word embedding techniques such as word2vec [8], GloVe [9] or fastText [2] have become conventional wisdom in many research fields such as NLP or information retrieval to represent text values. These word vectors are trained by processing large text corpora, e.g. the GloVe embedding[1] was trained on a corpus with 840 billion tokens. These embeddings are typically used to convert text values in a meaningful numerical representations used as input for ML tasks. However, a naïve application of a word embedding model is not sufficient to represent the meaning of text values in a database which is often more specific than the meaning in the natural language. Thus, we aim for additionally incorporating information given by the disposition of the text values in the database schema into the embedding, e.g. which words appear in the same column or are related. Therefore, we developed a relational retrofitting approach called Retro [5] which is able to automatically derive high-quality numerical representations of textual data residing in databases without any manual effort.

[1]https://nlp.stanford.edu/projects/glove/



movie		director		...
5th_Element	[0.21, ..., 0.96]	Luc Besson	[0.55, ..., 0.49]	...
Alien	[0.74, ..., 0,29]	Ridley Scott	[0.21, ..., 0.78]	...
Brazil	[0.61, ..., 0.08]	Terry Gilliam	[0.94, ..., 0.22]	...
...				
Valerian	[0.67, ..., 0.34]	Luc Besson	[0.88, ..., 0.07]	...

movie	director	...
5th_Element	Luc Besson	...
Alien	Ridley Scott	...
Brazil	Terry Gilliam	...
...		
Valerian	Luc Besson	...

Figure 1: Relational Retrofitting: basis word embedding W^0 and relation T (left), retrofitted word embedding W and augmented relation T^+

Relational Retrofitting. Figure 1 provides a small example sketching the main principles of the relational retrofitting process. Retro expects a database and a word embedding as input, e.g. a movie table T that should be retrofitted into a pre-trained word embedding W_0. W_0 can be generated using well-known text embedding techniques. Those models are able to generate embeddings capturing syntactic and semantic relations between words, such as word analogies, gender-inflections, or geographical relationships. To provide vector representations for textual information in databases one could simply reuse the vectors of pre-trained embeddings, e.g. map each term from T to a term-vector pair in W_0.

However, there often will be a semantic mismatch between word embeddings and textual information in databases:

1) Given the movie table, it is known that all entities within the movie column must be movies, although some of the titles, such as "Brazil" or "Alien", may be interpreted differently by the word embedding model.

2) T provides a specific amount of relation types like the movie-director, whereas in the word embedding representation W_0 large amounts of implicit relations are modeled, e.g. if the director of a movie is also the producer or one of the actors this might be represented in the word embedding although not explicitly visible.

3) Terms in T which occurring infrequent in the general domain can not be modeled accurately by word embedding models.

For instance word2vec has a limited vocabulary according to a frequency threshold. Many terms appearing in a database will therefore have no counter-part within the embedding.
Therefore, we developed Retro [5] which incorporates the database semantics and augments all terms in database relation by a dense vector representation. In the context of our movie example, Retro would generate a new embedding for "Terry Gilliam" which should be close to other directors and their respective vectors. Furthermore, Retro would create vectors for all other textual values in the movie table that encode the semantics given by the pre-trained word embeddings and the database. On one hand, this ensures that vectors appearing in the same column, such as movies or directors, are close to each other. On the other hand, this ensures that movie-director pairs can be resolved. These vectors are ready-to-use for a wide range of ML, retrieval and data cleaning tasks such as classification, regression, null value imputation, entity resolution, and many more.
Demonstration. Our web-based demo called RetroLive showcases our novel retrofitting approach and guides the user through the whole retrofitting process. The demo provides different database schemas and target word embeddings the users can choose from and allows to explore their data statistics and characteristics. Additionally, RetroLive allows to set and fine-tune the various hyperparameters of the retrofitting learning problem. The impact of the hyperparameter values, the input data characteristic and target embeddings can be studied in detail using either 2-dimensional projections or by using word similarity and analogy benchmarks. To demonstrate the usefulness of the embedding generated by the relational retrofitting approach, RetroLive comes with various pre-defined extrinsic evaluation tasks for classification and regression. Furthermore, the users are able to define their own machine learning tasks. Finally, the demo includes several baseline approaches such as the node embedding technique DeepWalk [10] and the original retrofitting approach [4] from Faruqui et al.

2 RETRO – BACKGROUND

Retro aims at combining word embeddings with relational text data to generate good vector representations for text values residing within databases. Therefore, our relational retrofitting approach learns a matrix of vector representations $W = (\boldsymbol{v_1}, \ldots \boldsymbol{v_n})$ with $\boldsymbol{v_i} \in \mathbb{R}^D$ for every unique text value $T = (\boldsymbol{t_1}, \ldots \boldsymbol{t_n})$ in a database. To find initial word vectors for every text value, we tokenize the them based on the vocabulary of the basis word embedding model and build centroid vectors which is a convenient way to obtain a representation of text values consisting of multiple tokens [1, 11]. These vectors are stored in a matrix $W^0 = (\boldsymbol{v}'_1, \ldots \boldsymbol{v}'_n)$ forming the basis for the retrofitting process. Besides, *columnar and relational connections* are extracted from the database (see Section 2.1). This encompasses semantic relations between text values, which are derived from the relational schema. Those connections are used to create a representation capturing the context of the text value in the database and thus help to preserve their semantics more accurately compared to a plain word embedding representation. In order to apply node embedding techniques like DeepWalk [10], all information is condensed into a common graph representation shortly presented in Section 2.2. The core procedure of the relational retrofitting is the adaption of the basis vectors W^0. This is performed by solving an optimization problem detailed further in Section 2.3.

2.1 Extracting Relational Information

One can derive different structural relations from the alignment of text values in the relational schema.
Columnar Connections: Text values with the same attribute, i.e. appearing in the same column, usually form hyponyms of a common hypernym (similar to subclass superclass relations). Thus, they share a lot of common properties which typically leads to similarity. We capture this information and assign each text value t_i to its column $C(i)$.
Relational Connections: Relations exhibit from the co-occurrence of text values in the same row as well as from foreign key relations. Those relations are important to characterize the semantics of text value in the database. We define a set of relation types R which is specific for a certain pair of text value columns. Those columns are related because they are either part of the same table or there exists a foreign key relationship between their tables. For every relation type $r \in R$ there is a set E_r containing the tuples of related text value ids. Relation types are directed. Accordingly, there is an inverted counterpart $\bar{r}$ for each relation r with $E_{\bar{r}} = \{(j,i)|(i,j) \in E_r\}$.

2.2 Graph Generation

A graph representation is a pre-requisite for training node embeddings such as DeepWalk [10] which serve as a strong baseline for Retro. To compile a graph $G = (V, E)$ the text values extracted by the tokenization process together with columnar and relational connections (Section 2.1) are combined. The node-set $V = V_C \cup V_T$ consists of text value nodes V_T for every distinct text value in a database column and blank nodes for every column V_C. The edge set $E = \bigcup_{r \in R} E_r \cup E_C$ consists of a set of edges E_r for every relational type and edges E_C connecting text values of one column to a common category node.

2.3 Optimization Problem

Retro considers relational and columnar connections (see Section 2.1) to retrofit an initial embedding. Accordingly, we define a loss function Ψ adapting embeddings to be similar to their basis word embedding representation W^0, the embeddings appearing in the same column, and related embeddings.

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

The columnar loss is defined by Ψ_C and treats every embedding $\boldsymbol{v_i}$ to be similar to the constant centroid $\boldsymbol{c_i}$ of the basis embeddings of text values in the same column $C(i)$.

$$\Psi_C(\boldsymbol{v_i}, W) = ||\boldsymbol{v_i} - \boldsymbol{c_i}||^2 \quad \boldsymbol{c_i} = \frac{\sum_{j \in C(i)} \boldsymbol{v}'_j}{|C(i)|} \quad (2)$$

The relational loss Ψ_R treats embeddings v_i and v_j to be similar if there exists a relation between them and dissimilar otherwise. E_r is the set of tuples where a relation r exists. $\widetilde{E_r}$ is the set of all tuples $(i,j) \notin E_r$ where i and j are part of relation r. Thus, each of both indices has to occur at least in one tuple of E_r.

$$\Psi_R(\boldsymbol{v_i}, W) = \sum_{r \in R} \left[\sum_{\substack{j:(i,j) \\ \in E_r}} \gamma_i^r ||\boldsymbol{v_i} - \boldsymbol{v_j}||^2 - \sum_{\substack{k:(i,k) \\ \in \widetilde{E_r}}} \delta_i^r ||\boldsymbol{v_i} - \boldsymbol{v_k}||^2 \right] \quad (3)$$

α, β, γ and δ are hyperparameters. Ψ has to be a convex function. In [5] we proved the convexity of Ψ for hyperparameter configurations fulfilling the following inequation:

$$\forall r \in R, i \in \{1, \dots, n\} \quad (\alpha_i \geq 0, \quad \beta_i \geq 0, \quad \gamma_i^r \geq 0) \tag{4}$$

$$\forall \boldsymbol{v_i} \in W \quad (4\alpha_i - \sum_{r \in R} \sum_{j:(i,j) \in \widetilde{E_r}} \delta_i^r \geq 0)$$

Given the property of convexity, an iterative algorithm can be used to minimize Ψ. Using sparse matrix calculations this can be done in parallel with linear time complexity according to the number of text values in W. Details are outlined in [5].

3 SYSTEM OVERVIEW

RetroLive is a fully functional system built on top of PostgreSQL. The front-end is a web application managing the provided input databases and embeddings (Section 3.1), controlling the relational retrofitting process (Section 2) and performs several ML tasks (Section 3.2). The system overview is depicted in Figure 2.
Based on an input *database schema* (1) and a target word embedding wodel (2) a so-called *basis word embedding* (3) is created which encodes each text value in the database as a dense vector. These vectors together with the extracted relational schema information (4) comprise the input for the relation retrofitting which returns a retrofitted embedding (5). In addition, a *graph representation* (7) of the text value relations (see Section 2.2) is generated and used to create node embeddings (8) which serve as a strong baseline for our approach. Moreover, the *graph representation* is used to apply the original retrofitting approach of Faruqui et al. (6). The basis word embeddings, node embeddings, simple and relational retrofitted embeddings are used to train and test ML models (9).

3.1 Datasets

The deployment used for the demonstration contains several preloaded database schemas and target word embeddings. Additionally, it provides several baseline embeddings.
Relational datasets: We prepared three datasets and imported them into PostgreSQL: The Movie Database (TMDB)[2], Google Play Store Apps[3] and Open Food Facts [4] which are all very popular datasets on Kaggle with a significant portion of textual data.
Target Word Embeddings: We used the popular 300 dimensional Google News embeddings[5] and an English fastText[6] model as target word embeddings for our retrofitting process.
Baseline Embeddings: We applied the original retrofitting approach [4] using the proposed standard parameter configuration of $\alpha_i = 1$ and the reciprocal of the outdegree of i for β_i within 20 iterations. DeepWalk (DW) is trained with its standard parameters and a 300 dimensions representation size.

3.2 ML Tasks

To demonstrate the usefulness of the relational retrofitting, the user can perform different *extrinsic* evaluation tasks, such as binary classification, missing value imputation and link prediction, on the embeddings generated with the relational retrofitting approach as well as the various baseline embeddings.
RetroLive comes with a set of pre-defined ML models:
Binary Classification: Text values can be assigned to two distinct classes by a classifier, e.g. we defined the task to distinguish

[2]https://www.kaggle.com/rounakbanik/the-movies-dataset
[3]https://www.kaggle.com/lava18/google-play-store-apps/
[4]https://www.kaggle.com/openfoodfacts/world-food-facts
[5]https://code.google.com/archive/p/word2vec/
[6]https://github.com/facebookresearch/fastText/blob/master/docs/pretrained-vectors.md

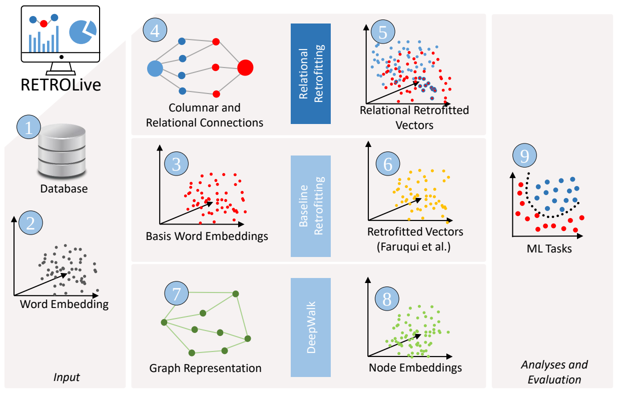


Figure 2: System Overview

US-American and non-US-American directors in TMDB.
Regression: Regression tasks assign input data points to a related numerical value, e.g. the budget or revenue in US dollar for a movie title.
Null Value Imputation: Text values are assigned to one category out of a finite set of category values, e.g. the "original language" of a movie or the app category for apps in the Google Play Store Apps dataset.
Link Prediction: The link prediction problem is typically defined on graphs where the goal is to predict links that are missing or likely to be created in the future. In our case, we consider the link prediction task for a specific relation to predict missing foreign key relations.
Our models consist of different multi-layer dense neural networks that can be applied on the generated node embeddings, pre-trained word embeddings, and our retrofitted embeddings. Details on the specific neural network architectures for all ML models are given in [5]. All experiments are repeated multiple times to obtain the distribution of the accuracy values.

4 DEMONSTRATOR WALKTHROUGH

Users access RetroLive through an interactive web interface shown in Figure 3 where they can configure and explore the whole relational retrofitting process. There are six main views:
Config and Retrofit: Those views allow to choose the database (three are pre-defined) and the target word embedding and configure the retrofitting process by setting the hyperparameters (A). If the retrofitting is done on the same data used for the ML Task (e.g. to predict the genres of movies expressed in the database by foreign key relations), users can blacklist specific relations for the retrofitting in the config view. Further, the relational retrofitting process can be triggered and monitored (B).
Results: In this view the user can inspect the extracted relational information (see Section 2.2) from the input schema in form of graph (C). Additionally, the embedding statistics for each text column are presented (D).
Analysis: In the analysis view, the users can inspect the characteristic of the retrofitted embeddings and compare them with the plain input word embedding. An interactive histogram (E) shows the distribution of the cosine similarity between the plain word vectors and the retrofitted vectors, i.e. to which degree certain vectors have been changed during the retrofitting process. The user can click on the individual bins to see additional information, e.g. in how many relations certain terms have been

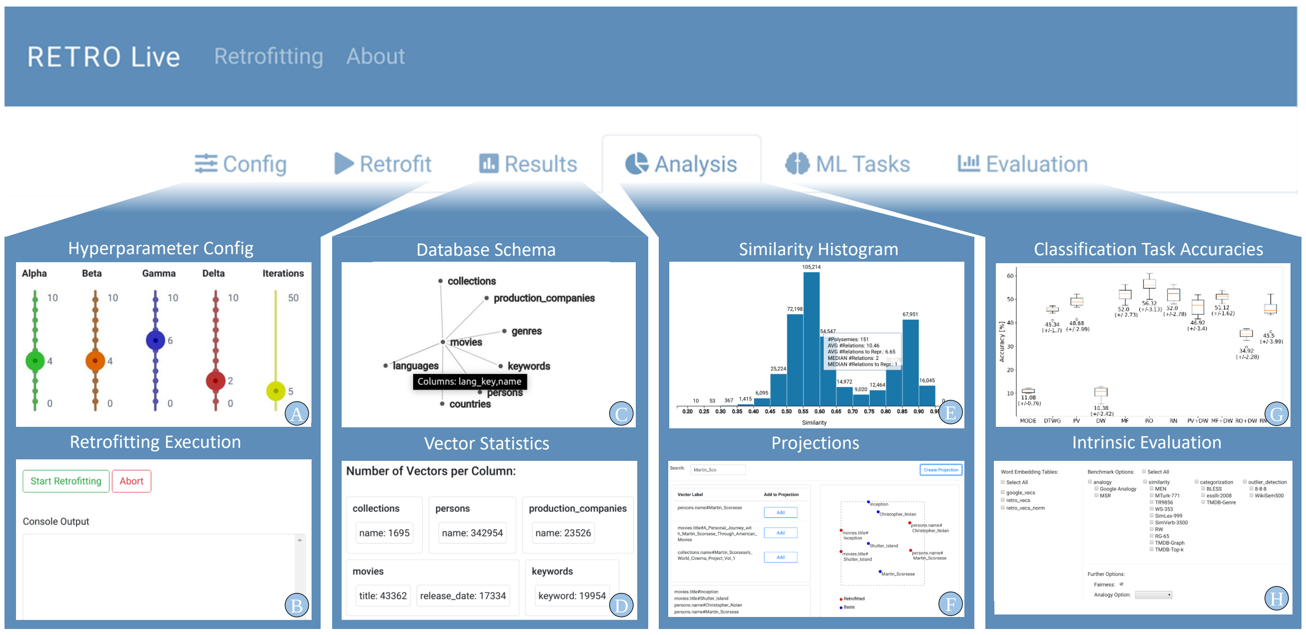


Figure 3: Demo User Interface

involved or in how many different columns a term appeared or to get a complete list of embeddings in the selected bin. Moreover, a 2-dimensional projection (PCA) shows the user-selected plain text and retrofitted vectors (F). In context of the TMDB database example, it can be seen that the vectors for movies and directors are arbitrarily distributed in the word embedding (red). However, after applying relational retrofitting the movie and director vectors (blue) are clustered and the difference vectors between movie and director pairs are of same length and orientation.

ML Tasks: To show the benefits of relation retrofitting the users can run different ML tasks (Section 3.2). The users select the embedding model (retrofitted, node, plain, etc.) they want to use for the given task. Training and testing data is retrieved from the database using pre-defined SQL queries which can be also modified by the user. Diagrams visualize the results (G).

Evaluation: Our demonstrator includes 14 intrinsic evaluation tasks to test word similarity, e.g. SimLex999 [6] MEN [3] or analogies, e.g. Google Analogy [7] (H). Here, the users can investigate whether original retrofitting and relational retrofitting affect the intrinsic task performance compared to plain word embeddings.

5 CONCLUSION

In this paper, we presented RetroLive, a novel system which allows generating retrofitted embeddings for an arbitrary database in a fully automated fashion. Participants can experience first-hand the power of relational retrofitting by tuning different hyperparameters, playing with different input datasets and investigating the effect of using a rich set of visualizations. Our demonstration highlights the potential of relational retrofitted vectors which achieve very good results for machine learning tasks like regression, binary, and multi-class classification. RetroLive also implements state-of-the-art baselines such as DeepWalk and the approach of Faruqui et al. which are both out-performed. Our system is publicly available under the permissive MIT license[7].

[7] https://github.com/guenthermi/postgres-retrofit

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 Intel®AI Research.

REFERENCES

[1] Abdulaziz Alghunaim, Mitra Mohtarami, Scott Cyphers, and Jim Glass. 2015. A Vector Space Approach for Aspect Based Sentiment Analysis. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. 116–122.
[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.
[3] Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *J. Artif. Int. Res.* 49, 1 (Jan. 2014), 1–47.
[4] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1606–1615.
[5] Michael Günther, Maik Thiele, and Wolfgang Lehner. 2019. RETRO: Relation Retrofitting For In-Database Machine Learning on Textual Data. arXiv:1911.12674
[6] Felix Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. 2015. Embedding Word Similarity with Neural Machine Translation. In *ICLR 2015*.
[7] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781
[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119.
[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.
[10] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.
[11] Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2015. Representation Learning for Aspect Category Detection in Online Reviews. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.