# Multi-Modal Data Exploration with a Learned Relational Embedding

Raul Castro Fernandez, Samuel Madden

## Introduction

In modern enterprises, analysts often engage in data exploration, where, given an unfamiliar database, they seek to understand if the database contains the answer to the question they want to answer. Exploration involves understanding the schema and contents of the database, and then devising a SQL query that provides the answer [1]. During exploration, analysts typically only have access to: i) a SQL interface; ii) the domain knowledge they possess; and iii) some documentation in the form of table descriptions and email exchanges with colleagues. Such resources are often insufficcient to fully understand the database, requiring hours of manual and time-consuming effort to craft the desired SQL query.

Data exploration workloads are quite different from traditional relational workloads, because users need to develop an understanding of table contents and relationships between tables, which requires a variety of ad-hoc analyses. To illustrate the point, consider the following example of a data exploration session:

Suppose an analyst wants to justify new legislation for education in the US. The analyst is given access to a newly published public database of 500+ tables that contains information about US federal spending and wants to determine how much money is spent in education per state. When browsing these tables, it would be useful to have a classification of each of them based on the kind of content they possess and how the tables are related so the analyst can quickly distinguish the relevant tables from irrelevant ones concerning health, infrastructure, etc. In addition, as the analyst is narrowing down the search to a few relations of interest, it would be useful to have access to a meaningful summary of the relations, to confirm whether the tables are interesting or not. If the analyst is deciding whether a set of tables are related to each other, it also would be useful to understand how the rows in each table relate to each other, before actually trying to join them. Finally, at any point, the analyst may want to use keywords to guide their search, to understand which tables, columns and rows contain related information to the keyword.

Unfortunately, supporting the *exploration strategies* in this example is difficult with relational databases, because they do not provide the ability to search, summarize, or identify related tables. Specifically, we would need to define a similarity metric between relations (or use an existing one such as [5]) and then implement the clustering of related tables. We would also need some way to sample rows from tables to create useful summaries, using perhaps one of the techniques from [4]. To understand whether two tables are related to each other, we'd need to spend time and effort figuring out their join keys, before even knowing if they are useful at all. The fundamental problem is that the implementations of the relational model are not designed for exploration workloads, where we need the ability to query data for arbitrary relationships more freely. In this paper, we propose an alternative representation of relational data that is more appropriate to these exploration workloads.

In particular, our idea is to represent relational data in a vector space in which cells, attributes, rows and tables are all vectors, and where the distance between vectors corresponds to how related items are to each other. Here *related* may mean they appear in the same row, the same table, or that they share relationships across tables or with other rows or records–whether they join or not. Suppose we have the federal spending database of the example above in such a vector space. By clustering tables based on their similarity in the vector space, we can identify relations that are similar to each other. We can summarize relations by selecting a diverse set of rows whose vectors are far from each other. Finally, we can directly measure the similarity of the tables by comparing their vectors. Thus, the vector space representation is a good fit for exploration workloads, as many exploration tasks can be expressed as vector operations based on distances in the space.

In this short paper we: i) overview a set of research challenges in building such vector space and ii) summarize applications of vector spaces beyond multi-modal database exploration. The main challenge of building the relational embedding is to come up with a vector space which represents relational data well. We propose a technique to learn the embedding directly from the data, which we call a *relational embedding*. Using this embedding, we build on top of modern language modelling techniques to build a high quality embedding. This process introduces several problems problems that we explain in detail next.

## Relational Embeddings

Building a relational embedding requires us to encode tokens from the relational table into vectors in such a way that the distance between vectors representing related tokens is closer than between the vectors of unrelated tokens. Creating a relational embedding involves several challenges:

**Encoding strategy:** A relation represents two types of relationships. The first type, which we call a *row-relationship*, is between the primary key and each of the attributes of the relation. This relationship indicates that the tokens are related according to the specific attribute name. The second type, a *column-relationship*, is between the values of a given attribute. This relationship indicates that the tokens are of the same type, as indicated by the attribute. A relational embedding thus consists of two dictionaries that map a token to its vector representation: one to represent the tokens based on their row-relationships, and a second one to represent the tokens based on their column-relationships. Each token will thus have two vector representations.

In our approach, vectors are learned directly from the data using a *Skipgram model* [2]. The Skipgram model, originally proposed

for learning a language model, uses the concept of a *context*, which is a window over the text used to determine what pairs of words are positive samples (those within the window) and which pairs are negative samples (pairs formed by one word in the window and one word outside the window). To limit the number of total negative samples—necessary to achieve reasonable training times—a negative sampling strategy is often used that selects only a sample of words out of the window.

When learning a relational embedding we need to decide both the context of the Skipgram model as well as the sampling strategy to use. Our goal is to obtain a better (more predictive of relationships) vector representation than that given by a baseline Skipgram model.

**Space and Performance Overhead:** To learn good quality vectors we must use a high capacity model: vectors must be high-dimensional to learn good quality embeddings. Each token from the relational data is mapped to two high-dimensional vectors (one for row-relationship and one for column-relationship), resulting in a significant storage and performance overhead. For example, the relational embdding of a small, 100MB database, takes about 2GB in space. A key challenge is to reduce the storage requirements of this embedding. Among the strategies use are: i) reducing the precision of the vector components once they've been learned; ii) reducing the dimensionality of the embedding using non-linear techniques [3] that maintain as much as possible the pairwise distances of the vectors in the original embedding. This second point is the most promising: a non-linear reduction to 3 dimensions reduces the relational embedding to around 40MB in the example mentioned above (a 50X reduction in storage) which permits querying the embedding with higher performance as well. However, it also introduces a data quality problem, which we explore in the next paragraph.

**Cleaning Vector Data:** When dealing with high-dimensional data, one must pay attention to the problems associated to the curse of dimensionality. Two specific problems are a challenge that makes part of our research agenda: the existence of *hubs*, vectors that appear close to most other vectors, and therefore do not add useful information, and the phenomenon of *distance concentration*, which makes distances between high-dimensional vectors tend to the same magnitude. In addition to these two, because the relational embedding is learned, it is noisy. Thus, a research challenge is how to define dirty data in the context of vector data—for example, hubs—and propose techniques to reduce them, of which those of topological and shape analysis seem promising. This is necessary because, otherwise, the effect of dirty data makes dimensionality reduction techniques very inefficient.

**Performance Primitives/Structures:** Many of the above requirements demand the computation of pairwise distances between a large number of vectors, e.g., to find hubs, to perform non-linear dimensionality reduction, etc. This operation can be very time consuming. Speeding up the process calls for the design and implementation of index structures geared towards vector data.

**API guarantees:** Both the multi-modal exploration application outlined in this paper as well as few others mentioned below demand querying vector data with variants of top-k queries. For the applications to be robust to user demands, we must produce rankings that are informative to users. Specific design points are the distance metric to use for measuring distance, and the size of the ranking for which the results are likely of high precision. In addition, we must implement API calls efficiently for maximum performance.

## Applications of Relational Embeddings

In addition to the multi-modal data exploration that motivates this short paper, we have identified and investigated other applications of relational embeddings which we outline below:

**Fabric of data:** The major advantage of the relational embedding is that it works as an intermediate representation in which we can represent independently created databases as well as unstructured data, from PDFs, to emails, to conversations of Slack channels. Having a common representation for all these kinds of data without any human effort has several potential advantages. First, it facilitates discovering related data among heterogeneous sources without spending any upfront effort. Second, it can help with understanding how the data can be curated, i.e., filling missing values or complement rows with information from a PDF. Last, it can help with integrating data from different databases by understanding how concepts in different database related to each other. This application, which we call the *fabric of data* aims to improve users' productivity and enable new applications.

**Generative model of relational data:** We can use the relational embedding as the training data of a generative model. Creating a generative model can be useful, for example, to grow the database with new new synthetic data that can then be used for performance benchmarking. The advantages of the model is that it allows to grow data that follows the distribution of each row- and column-relationship, instead of demanding a data distribution beforehand. In addition to the previous application, another promising application of the generative model is filling missing values, detecting outliers and other data quality issues.

**Secure data exchange:** It is plausible to share different translations of the embedding with different entities based on the security clearances of those individuals. This can be done in such a way that only the person with the right translation can query the embedding and obtain results that are sensical. Exploring how to leverage differential privacy in this context is another way in which the relational embedding may facilitate secure data exchange.

**Talk Outline:** In the talk, I would motivate the applications of the relational embedding and fabric of data outlined above. I would explain in detail the challenges of working with data in a vector space. Finally, I would present the preliminary results we have obtained and, if time permits, I'd demo the current prototype we have.

## Conclusion

Today's analysts engage in complex data exploration sessions to understand databases and answer the questions necessary to perform their job. Relational embedding is an abstraction that represents data as a vector space, and that has the potential of facilitating a multi-modal exploration strategy of relational data. We have identified an interesting research agenda around the construction and usage of relational embeddings, as well as other applications that are enabled by the new abstraction.

## 1. REFERENCES


[1] S. Idreos, O. Papaemmanouil, et al. Overview of Data Exploration Techniques. In *SIGMOD*, 2015.

[2] T. Mikolov, I. Sutskever, et al. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 2013.

[3] L. van der Maaten and G. E. Hinton. Visualizing Data using t-SNE. 2008.

[4] Y. Wang, A. Meliou, et al. RC-index: Diversifying Answers to Range Queries. *VLDB*, 2018.

[5] X. Yang, C. M. Procopiuc, et al. Summarizing Relational Databases. *VLDB*, 2009.