Exploiting Latent Information in Relational Databases via Word Embedding and Application to Degrees of Disclosure

Rajesh Bordawekar
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
bordaw@us.ibm.com

Oded Shmueli
CS Department, Technion
Haifa, Israel 32000
oshmu@cs.technion.ac.il

ABSTRACT

Cognitive Databases is a new approach for enabling Artificial Intelligence (AI) capabilities as standard features within relational database systems. Relations are textified and the text is used to build a Word Embedding (WE) model that captures the latent relationships between database tokens of various data types. For each database token, the model includes a low dimensional vector (say, 200) that encodes the token's relationships with other tokens. The vectors are used in the existing SQL query infrastructure via UDFs. Queries use the model vectors to express semantic similarity/dissimilarity, inductive reasoning, analogies and seamlessly utilize knowledge from external sources such as Wikipedia and PubMed.

WE enables novel capabilities such as the controlled disclosure of database information in a variety of ways. The degree of disclosure may depend on the sensitivity of the information and the recipient's need to know, e.g., test results may be considered sensitive and should be only be openly disclosed to divisions concerned with them. Disclosure may be viewed as a new kind of controlled sharing of information for cooperation and integration purposes.

There are some challenges in integrating WE methods into the database engine, necessitating new techniques. There are also interesting theoretical problems concerning the WE coding power.

ACM Reference format:
Rajesh Bordawekar and Oded Shmueli. 2019. Exploiting Latent Information in Relational Databases via Word Embedding and Application to Degrees of Disclosure. In *Proceedings of 9th Biennial Conference on Innovative Data Systems Research(CIDR'19), Asilomar, California, USA, January 2019 (CIDR 2019),* 6 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Traditionally, relational databases have been used to analyze enterprise datasets that comprise mostly of well-qualified typed entities (e.g., character(n), decimal, float, or timestamp). However, over the years, relational databases have been increasingly used to store and process free-formed unstructured text data (e.g., customer reviews). It is intuitively clear that databases with such unstructured text entities have a significant amount of latent semantic information. However, columns that contain different types of data, e.g., strings, numerical values, images, dates, etc., possess significant latent information in the form of inter- and intra-column relationships. The usual way to utilize this information is using SQL and extensions, such as text extensions, or User Defined Functions (UDFs) to handle exotic data types. However, these extensions are rather limited in their *smarts*. Specifically, SQL queries rely on *value-based* analytics to detect patterns. In addition, the relational data model neglects many inter- or intra-column relationships. Thus, traditional SQL queries lack a *holistic* view of the underlying relations, and thus are unable to extract and exploit semantic relationships that are *collectively* generated by tokens in a database.

A Cognitive Databases [1, 2] is a novel relational database system, which uses word embedding techniques [6, 8, 9, 11] to extract latent knowledge from a database table or a collection of tables. The generated word-embedding model captures inter- and intra-column semantic relationships between database tokens of different types. For each database token (value, field, object), the model includes a vector that encodes contextual semantic relationships. A cognitive database seamlessly integrates the model into the existing SQL query processing infrastructure and uses it to enable a new class of SQL-based analytics queries called Cognitive Intelligence (CI) queries. CI queries use the model vectors to enable complex semantic queries over relational data such as semantic similarity or dissimilarity, inductive reasoning queries such as analogies or semantic clustering, and predictive queries using entities not present in a database but only in corpora on which the model is co-trained.

There are interesting implications of word embedding based modeling for enabling selective information dissemination for relational data. We illustrate how the word-embedding approach can enable a cognitive database to: (1) reveal various degrees of information, (2) invoke semantic CI queries over encrypted data, and (3) provide limited disclosure via semantic encoding of the underlying database schema and data.

2 COGNITIVE DATABASE DESIGN

In the database context, vectors may be produced by either learning on text transformed and extracted from the database itself and/or using external text sources, such as Wikipedia. Training a word-embedding model from a relational database requires two stages. The first stage, *textification*, takes a relational table with different SQL types as input and returns an unstructured but meaningful text corpus consisting of a set of sentences. This transformation allows us to generate a multi-modal embedding model with uniform semantic representation of different SQL types. In addition to text tokens, our current implementation supports numeric values and images (we assume that the database being queried contains a `VARCHAR` column storing links to the images). We use different

CIDR 2019, Asilomar, California, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

strategies for converting a non-text relational data to text: e.g., values in a numeric column are first clustered using a standard clustering approach (e.g., K-Means), and then replaced by a text token that represents the corresponding cluster. For images, one approach classifies images into classes using a pre-trained model and then represents each image by a string token that represents its class. Alternatively, one can first extract text features from an image using off-the-shelf image services, such as IBM Watson Visual Recognition Service [5], and then use the extracted features to train the embedding model.

Usually, to focus learning, the relevant portion of a table to be learned is defined by using a relational view. We use an unsupervised training approach based on the Word2Vec (W2V) [7] implementation to build the word embedding model from the generated text corpus (other training approaches perform similarly). The text corpus is organized as a set of *English-like* sentences, separated by stop words (e.g., newline). Each sentence correspond to a row in the relational view and is used as a neighborhood context during the training of the word embedding model. Hence, *the inferred semantic meaning of the relational entities reflect the collective relationships defined by the associated relational view (generated by relational operations such SELECT, PROJECT, and JOIN.)*

Our training implementation builds on the standard W2V implementation, but it varies from the that approach in a number of important aspects: (1) A sentence generated from a relational row is generally not in any natural language such as English. Therefore, the underlying assumption from word2Vec that the influence of any word on a nearby word decreases as the word distance increases, is not applicable. In our implementation, every token has the same influence on the nearby tokens in the context. (2) Another consequence is that unlike an English sentence, the last word is equally related to the first word as to its other neighbors. To enable such relationships for the last word, the first word can be viewed as its immediate neighbor). (3) For relational data, we provide special consideration to primary keys, which are unique (and therefore usually have a limited number of appearances which hinders learning). First, the standard W2V discards less frequent words from learning. In our implementation, every token, irrespective of its frequency, is assigned a vector. Second, irrespective of the distance, a primary key is considered a neighbor of every other word in a sentence and included in the neighborhood window for each word. Also, the neighborhood extends via foreign key occurrences (of a key value) to the row in which that value is key. (4) Finally, our implementation is designed to enable incremental training, i.e., the training system takes as input a pre-trained model and a new set of generated sentences, and returns an updated model. This capability is critical as a database may be updated regularly and one can not rebuild the model from scratch every time. External information may be incorporated in two basic modes: (a) by providing text that augments the database-derived text for training, and (b) by providing external pre-trained models derived from the external information. The use of pre-trained models is an example of transfer learning, where a model trained on an external knowledge base can be used either for querying purposes or as a basis of forming a new model. This of course necessitates management of models as well as models' identification when used within user-defined functions (UDFs).

3 COGNITIVE INTELLIGENCE QUERIES

A cognitive relational database is an extension of the underlying relational database, and thus supports all existing standard relational database features. In addition, a cognitive relational database supports a new class of business intelligence (BI) queries called Cognitive Intelligence (CI) queries. The CI queries extract information from a relational database based, in part, on the contextual semantic relationships among database entities, encoded as meaning vectors. At runtime, the SQL query execution engine uses various UDFs that access the trained vectors from the system table, as needed, and answers CI queries. Similarly to other relational queries, CI queries take relations as input and return a relation as output. CI queries augment the capabilities of the traditional relational BI queries and use all standard existing SQL operators.

Our current implementation is built on the Apache Spark 2.2.0 infrastructure. The implementation supports, via UDFs, four types of CI SQL queries: similarity queries, inductive reasoning, prediction, and cognitive OLAP. These queries can be executed over databases with multiple data types: we currently support text, numeric, and image data. The similarity queries compare two *relational variables* based on similarity or dissimilarity between the input variables. Each relational variable can be either a set or sequence of tokens. In case of sequences, computation of the similarity value in some UDFs takes the ordering of tokens into account where the closest the token to the beginning of the sequence, the higher the weight. The similarity value is then used to classify and group related data. The inductive reasoning queries exploit latent semantic information in the database to reason from part to whole, or from particular to general [12, 13]. We support different types of inductive reasoning queries: analogies, semantic clustering, analogy sequences, clustered analogies, and odd-man-out. Given a token from an external data corpus (which is not present in a database), the predictive CI query can identify database tokens that are similar, or dissimilar, to the external token by using the externally trained model. Finally, cognitive OLAP allows SQL aggregation functions such as MAX(), MIN() or AVG() over a set that is identified by contextual similarity computations.

To demonstrate the capabilities of Cognitive Databases, consider a semantic clustering CI query on a relational *multi-modal* database (Figure 1): the original database lists national parks with string tokens representing image file names, e.g., `n00015388_18458.jpeg`. We first create a training table using text features extracted from the images by using the Watson VRS system. The training table is then used to build a multi-modal word embedding model that captures relationships between text and image features. This model is then used to answer CI queries that use both text and image variables. For example, the goal of query shown in Figure 1 is to identify all images that are similar to every image in the set of user chosen images. Such images share one or more features with the input set of images. For this query, we select images of a lion, a vulture, and a shark as the input set and use the `combinedAvgSim()` UDF to identify images that are similar to all these three images. Although the input images display animals from

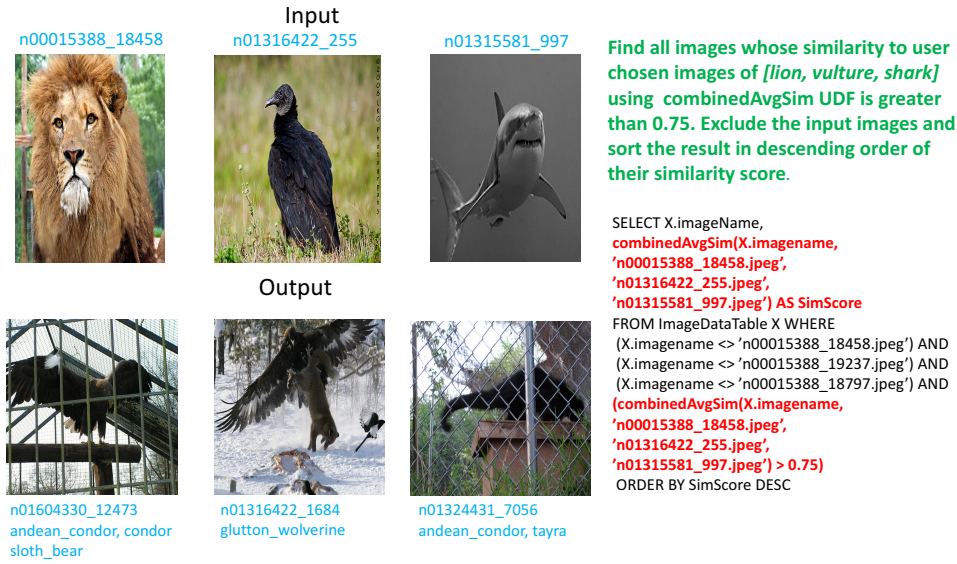

Figure 1: Inductive reasoning CI query for semantic clustering of images

three different classes, they share one common feature: all three animals are carnivorous. The UDF computes the average vector of the three input images and then selects those images whose vectors are similar to the computed average vector with similarity score higher than 0.75. Figure 1 shows the top three image results: andean condor, glutton wolverine, and tyra. Although these animals are from different classes, they all are carnivores, a feature that is shared with the animals from the input set.

To further illustrate CI capabilities, consider a query to find all images of animals whose classD similarity score (in ImageDataTable) to the concept of *Hypercarnivore* of Wikipedia, which does not appear in the database, exceeds 0.50. Exclude animal images that are already tagged as carnivore, herbivore, omnivore or scavenger. The query is presented in Figure 2. The query uses a UDF called `proximityAvgExtKB()`. See [1] for further details.

4 CONTROLLED DISCLOSURE VIA WORD EMBEDDING

We now outline interesting implications of using word embedding models for querying relational databases. We will use the process illustrated in Figures 3 and 4 as a running example.

Consider a single relational database relation R with five columns: A, B, C, D and E. Further, assume its first column, A, contains the primary key, a string that is unique for each relation tuple (record, row). In disclosing R to a recipient we identify the following *Disclosure Steps*:

(1) Deciding which columns should be completely eliminated, say due to a very high degree of sensitivity. In our example, we decide to eliminate column E.
(2) Deciding the content of which columns should be encrypted prior to producing word vectors. In our example, we decide that column D should be encrypted. This keeps equality between equal entries in different tuples (rows) for *this* column, but severs identifying these values in other columns (inter-column severance) as well as hides the true nature of the content within an encrypted column. Denote the modified relation R as R'.
(3) Vector construction based on texitifying R'. This step associates a vector with each token in relation R', see Figure 3. Each row of the table Vectors depicts the 200 entries of the vector associated with the database token in the Token column.
(4) Deciding which columns of R' are to be disclosed to the recipient(s). In our example, we decide to disclose all columns, A, B, C, and (the encrypted version of) D. Denote by R'' the relation obtained by restricting R' to the columns to be disclosed (i.e., in our example, R' =R'').
(5) Deciding which R'' columns, that are to be disclosed to recipient(s), should be encrypted prior to disclosure. Assume that, in our example, we decide to encrypt column B prior to disclosure. Recall that the Associated Vectors were produced prior to encrypting column B. The vector associated with any encrypted value in column B is the one that was associated with the pre-encrypted value. For example, in Figure 4, The vector of *e200301* is the one associated with $C_{72}H_{95}C_lN14O_{14}$. We shall refer to the end-result relation as Rf, see Figure 4.
(6) Disclosing Rf and the collection of pairs (w, v) in Associated Vectors where w is a token occurring in the disclosed columns of Rf and v is the associated vector.

4.1 Degrees of Disclosure

The end result of the outlined disclosure process is that the recipient is presented with a relation (Rf in our example) and with each token, its associated vector, see Figure 4. The important point to note is

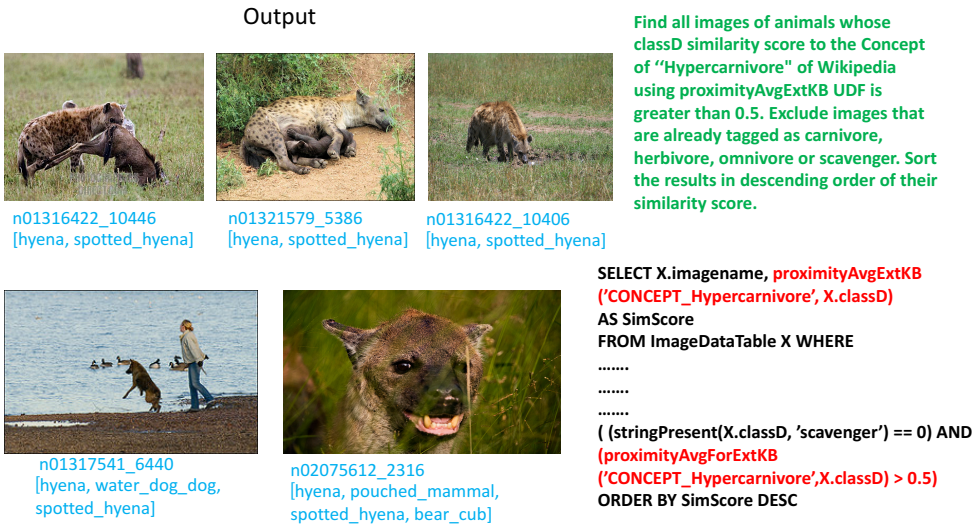

Figure 2: A CI query using an external concept

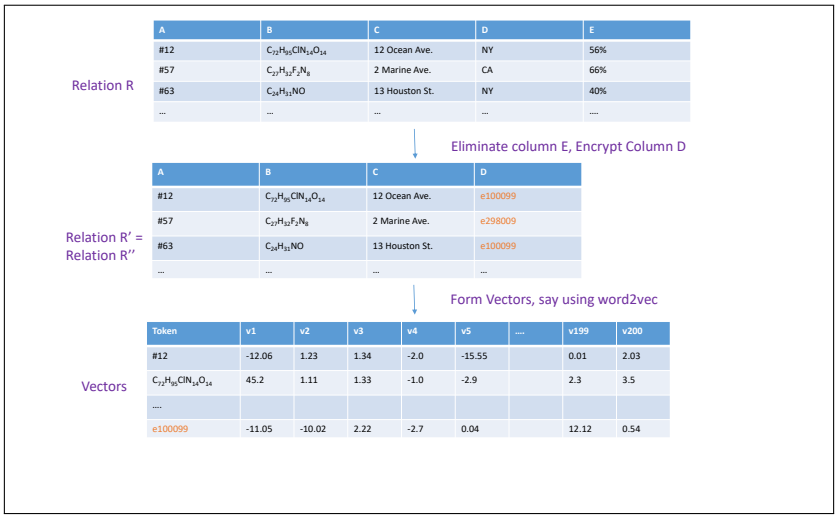

Figure 3: Producing word vectors from a modified relation, Disclosure Steps 1-4

that the recipient obtains significant additional information beyond the mere content of Rf. The vectors, in fact, encode knowledge not present in Rf, i.e., knowledge accessible through the vectors, say using CI queries. For example, column D is encrypted just prior to disclosure. However, the vectors that are associated with the encrypted tokens in column D were produced *prior* to performing encryption. Therefore, these vectors embed knowledge regarding these pre-encrypted tokens and their co-occurrences with other tokens. This knowledge is no longer available in Rf in isolation (i.e., without vectors). This way, a fine line is drawn in that although the precise identity of these encrypted columns is not disclosed, knowledge about their nature and associations is disclosed indirectly through their vectors. On the other hand, *decoding* vectors and associating them to original relational tokens appears to be a daunting task (the precise hardness is an open problem). Therefore, indirect information disclosure via vectors provides a level of information hiding that may be appropriate to many real-life situations.

Let us consider another example over the same four columns relation R to illustrate the interplay between exposed and hidden information. Suppose this time our table is describing employees, column A is the employee badge number and column C records employees' addresses. This time, in the final stage of forming Rf, we encrypt column C instead of column B (that describes expertise). Let Rf' denote the final disclosed relation in this case. We observe that while forming vectors, column C was not encrypted (i.e., clear-text).

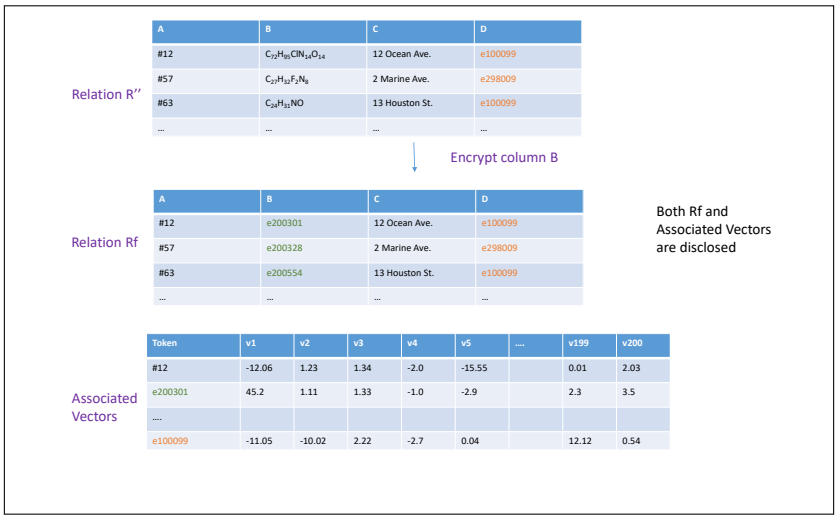

Figure 4: Producing word vectors from a modified relation, Disclosure Steps 5-6

If two employee addresses are identical, this is easy to detect in the supplied information (although the addresses themselves cannot be easily discerned) as these addresses are identically encrypted. If two addresses are *close* (say, same town and street, different number), this information will likely be exposed in that the vectors of these two addresses are likely to be close (i.e., high cosine similarity). In this way, information may be hidden but partially exposed to a certain degree. Note that the street address information will also affect closeness of the employee's, say Joe Smith's, (badge) column A value associated vector to other employees (badge) column A vectors values (as well as closeness to vectors for tokens in other relation columns). Therefore, if one is interested in employees close (according to some facets) to *Joe Smith*, this street address information, that is encrypted in `Rf'`, will likely affect querying results.

The *Disclosure Steps* outlined above introduce a sequence of measures of information hiding: eliminating columns, encrypting prior to vector construction, eliminating a column prior to disclosure, and encrypting a disclosed columns prior to disclosure. However, there are measures that reduce information hiding. One such measure that increases information exposure is the use of external data sources, e.g., Wikipedia. During training, we can mix the text obtained by texitifying the relation with text derived from external source(s). This way, the vectors of database tokens may encode closeness to terms (tokens) that do not even appear in the database, thereby exposing additional information. For example, suppose that relation R deals with medical drugs. The word toxic may not appear in R. However, column B contains chemical formulas. Certain compounds may be identified by an external source as toxic. Training on both R and the external text source may reveal closeness of a token of a column, say B (or A) vector, to the vector of `toxic` even though `toxic` does not appear in R at all.

4.2 Querying using Encrypted Tokens

As shown in Figures 3 and 4, a token used in forming the word embedding model can be the encrypted version of the original (cleat-text) data. This raises the question of what limitations are imposed by encryption on querying. As CI queries use string tokens to access the associated vectors, they can operate on either clear-text or encrypted versions of the tokens. Therefore, even if the source database is entirely encrypted, the generated word embedding model will be able to capture relationships between the tokens as if they were in the clear-text (unencrypted) format. However, there are some important limitations on querying:

(1) Usually when the SQL queries are composed, the encrypted token names (e.g., *e10099*) are not known. Therefore, some manual editing may be required prior to query execution.
(2) Encrypted columns cannot be compared to, or equated with, non-encrypted values in the query.

Lastly, we mention the phenomenon of *phantom connections*. Given a disclosed relation and associated vectors table, one can textify the relation and produce new vectors for the tokens thus presented. Then, one can compare closeness ranking of the supplied vectors versus the closeness between newly generated vectors. These closeness differences hint at sources of closeness that were *eliminated* in the disclosed relation, for example a column that was used originally in learning vectors and has been eliminated prior to disclosure. This underlines the fact that information hiding in this scheme is *soft* and is designed to make obtaining additional information from the disclosed information harder, but not impossible.

4.3 Disclosing Information by Disclosing an equivalent Synthetic-Text

Learning a model (vectors) on text obtained from various internal and external sources is a key idea in expanding the expressive power of SQL to use terms not explicitly mentioned in the source data. This sub-section deals with providing an information source

with neither explicitly providing the underlying table or database in any form, nor explicitly providing vectors per tokens. The idea is to provide a synthetic text generator that essentially produces a continuous stream of text upon demand (e.g., "provide the next 1000 words") with the concurrence statistics of the disclosed information source. The tokens (words) that are generated are the ones from the underlying information source. As in the case of controlled disclosure, discussed above, one may drop certain columns and encrypt others prior to preparing the data structure that enables synthetic text generation. This enables further control over the disclosed content.

Logically, the main data structure that enables synthetic text generation includes records of the form:

$$(token_1,\ token_2, ..., token_k,\ token_{out},\ prob)$$

The meaning is that if the last k tokens to be generated are $token_1$, ...,$token_k$ then with probability $prob$ the token to be generated is $token_{out}$. Conceptually, such records enable synthetic text generation after starting the text with an authentic sequence out of the original text. Efficient implementation techniques of this idea are presented in [3]. We note that the larger k is, the more precise the captured statistics is. This provides another control level on (a) the preciseness of disclosed information, and (b) the computational cost of generating the synthetic-generation enabling data structure.

5 SYSTEM CHALLENGES

Cognitive Databases present new system requirements, these include:

(1) Efficiently training models at a large scale. This involves efficient textification and model learning with a variety of machine learning techniques.
(2) When new tokens are introduced, for reasonably large databases, their vectors will have little influence on existing vectors. However, for query processing, new tokens need be associated with vectors. This raises the need for incrementally and quickly learning new vectors.
(3) Managing a vast collection of models, both internal and external.
(4) Efficiently performing UDFs. This is challenging as many UDFs process a large collection of vectors. New algorithm as well as hardware acceleration may be necessary. See work in this area in [4].
(5) Designing and integrating AI-oriented UDFs into the query compilation and optimization process.
(6) Automatically executing versions of the same query with different parameters (e.g., executing a query with two different cosine distance bounds, choosing the 'better' one).

6 CONCLUSIONS AND FUTURE WORK

In this paper, we briefly reviewed the concept of Cognitive Databases, a novel relational database system, which uses unsupervised word-embedding models to capture and exploit latent information in relational data. We view Cognitive Databases as a precursor to a new generation of relational databases that seamlessly integrate AI capabilities into the database data manipulation capabilities, in a uniform, dynamic and generic fashion. This should be contrasted with the practice in which targeted learning is performed over database-stored data in a separate system aiming at achieving a specific task. We discussed how the embedding approach can be used for controlling data access by enabling various degrees of information disclosure over relational tables. While our suggested methods are preliminary, we hope they will spur further exploration and analysis into this developing area.

A potentially important property of word embedding is its enabling the encoding of source databases. Consider a scenario where a word embedding model with encrypted tokens is being used for supporting CI queries. We conjecture that given just a word embedding model with encrypted tokens and vectors of real-valued numbers, it is not practically possible to deduce the schema and data of the source data used to create the model (the source data can be a database table or unstructured text corpus, or both). The precise encoding power of word embedding techniques is an interesting open problem. We hope this paper will encourage researchers to investigate and explore the theory underlying word-embedding (and other neural networks [10] based encoding schemes) for use in relational databases.

REFERENCES

[1] Rajesh Bordawekar, Bortik Bandyopadhyay, and Oded Shmueli. 2017. Cognitive Database: A Step towards Endowing Relational Databases with Artificial Intelligence Capabilities. *CoRR* abs/1712.07199 (2017). arXiv:1712.07199 http://arxiv.org/abs/1712.07199

[2] Rajesh Bordawekar and Oded Shmueli. 2017. Using Word Embedding to Enable Semantic Queries in Relational Databases. In *Proceedings of the 1st Workshop on Data Management for End-to-End Machine Learning (DEEM'17)*. ACM, New York, NY, USA, Article 5, 4 pages. DOI: https://doi.org/10.1145/3076246.3076251

[3] Rajesh Bordawekar and Oded Shmueli. 2018. System and method for natural language processing using synthetic text. *United States Patent* 10,025,773 (2018).

[4] Michael Günther. 2018. FREDDY: Fast Word Embeddings in Database Systems. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018,* Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 1817–1819. DOI: https://doi.org/10.1145/3183713.3183717

[5] IBM Watson. 2016. Watson Visual Recognition Service. www.ibm.com/watson/services/visual-recognition/. (2016).

[6] Omer Levy and Yoav Goldberg. 2014. Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014*. 171–180. http://aclweb.org/anthology/W/W14/W14-1618.pdf

[7] Tomas Mikolov. 2013. word2vec: Tool for computing continuous distributed representations of words. (2013). github.com/tmikolov/word2vec.

[8] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting Similarities among Languages for Machine Translation. *CoRR* abs/1309.4168 (2013). http://arxiv.org/abs/1309.4168

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *27th Annual Conference on Neural Information Processing Systems 2013*. 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality

[10] Nicolas Papernot, Martin Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. *CoRR* abs/1610.05755 (2017). arXiv:1610.05755 http://arxiv.org/abs/1610.05755

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf

[12] David E Rumelhart and Adele A Abrahamson. 1973. A model for analogical reasoning. *Cognitive Psychology* 5, 1 (1973), 1 – 28. DOI: https://doi.org/10.1016/0010-0285(73)90023-6

[13] Robert J Sternberg and Michael K Gardner. 1979. *Unities in Inductive Reasoning*. Technical Report Technical rept. no. 18, 1 Jul-30 Sep 79. Yale University. http://www.dtic.mil/docs/citations/ADA079701