Semantic Preserving Embeddings for Multi-Relational Graphs

Pedro Almagro Blanco
Complex Systems Modeling Group
Central University of Ecuador
Quito, Ecuador
Email: palmagro@uce.edu.ec

Fernando Sancho Caparrini
Department of Computer Science
and Artificial Intelligence
University of Seville, Spain
Email: fsancho@us.es

***Abstract*—In this paper a new machine learning approach to the study of Multi-Relational Graphs as semantic data structures is presented. It shows how vector representations that maintain semantic and topological features of the original data can be obtained from neural encoding architectures and considering the topological properties of the graph. Also, semantic features of these new representations are tested by using some machine learning tasks and new directions on efficient link discovery methodologies on large relational datasets are investigated.**

***Keywords*—Multi-relational Graphs; Semantic Networks; Link Discovery; Neural Autoenconder; Feature Extraction**

I. Introduction

Because of the structural complexity of graphs (and specially, of multi-relational graphs), their study within the field of Machine Learning have not been as prolific nor as successful as other data structures with a more simple and defined scheme, such as numerical series, images, or tables from relational databases (that, finally, can be seen as data-frames with heterogeneous structure in their columns). Consequently, the most common approaches to include them in Machine Learning projects have been through an oversimplification of the internal structure of the graph (removing the *multi* feature in them, and considering only one single type of nodes and edges) in order to project its content in an easier way over any of the simpler structures. In this sense, usually applying Machine Learning methodologies on graphs has been carried out by their *linearization*. In fact, these approaches reduce the chances to obtain algorithms able to extract nontrivial semantic information from semantic networks, since the projection of the network leads to a loss of the structural richness that is being studied and frequently characterizes this type of data structures.

For example, a common approximation to learning from graph structures has been by tackling it as a task of learning a function defined on the nodes of the graph, and has received a considerable attention in machine learning circles by representing the functions defined on the graph through a Hilbert space associated with the graph Laplacian ([1], [2], [3], [4], [5]). Only recently some other approaches taking into account the rich structure of graphs, in the context of this work, have been considered.

In recent years, complexity on the type of information that needs to be manipulated and the consequent emergence of new storage and retrieval systems, such as NoSQL solutions [6], moving away from the rigid structure of the classical relational model, make it necessary to find alternatives that maintain as much as possible the rich semantic structure that characterizes multi-relational graphs as the standard tool for representing complex information.

One of the options that can be of interest to get more flexible solutions to this problem, and that has been applied previously to some other machine learning problems, goes through building embeddings of the graph that, by projecting its nodes on an adequate space (usually a vector space), are able to render the semantic features of the graph over easily recognizable objects in this new space. In this way, it is understood that the semantic information stored in the graph has been captured by the embedding and the resulting projected structures.

In the next section, some of the background needed for the development of the proposed solution, mainly focused in embeddings, multi-relational graphs, and neural encoder concepts are shown. In section 3 some previous related works are presented, contextualizing the main contributions of this work. Section 4 is devoted to the presentation of the proposal for building semantic preserving embeddings by using neural encoders, that will be validated with their application in machine learning contexts in section 5. Finally, in section 6 some conclusions, final remarks and future work will be shortly presented.

II. Background

Along this paper, the term *embedding* is used as the process to inject an instance X of some mathematical object into another object Y preserving, in some way, the original structure. This embedding is given by a function, $f : X \to Y$, that sometimes is called a *projection*.

In this case, we are interested in embeddings preserving the topological structures we can recognize in the graph (given by the relations, or edges, between the nodes of the graph), and also preserving the semantic structure (given by types annotation in nodes and edges). Also, we will work with embeddings into finite dimensional vector real spaces, $\mathbb{R}^n$. In this way, we are interested in finding embeddings that can reflect, within the vector space features (distance, linearity, clustering, etc.) some semantic features of the original graph.

In order to model situations with a wide variety of relationships between the elements of the graph, the concept

of *Multi-relational Graph* is introduced, which in a first approach slightly extends the standard graph definition by adding tagging capabilities on edges and nodes:

Definition: A *Multi-relational Graph* is a triple, $G = (V, E, \tau)$, where V is a set of *nodes*, $E \subseteq V \times V$ is a set of *edges* (relations), and $\tau : V \cup E \to \Omega$ is a *tagging function* for the elements of the graph on the set of *tags*, Ω (also named *types set* along this paper).

Figure 1 shows a very simple example of a multi-relational graph with seven nodes and seven edges, and with types associated to the elements of the graph.

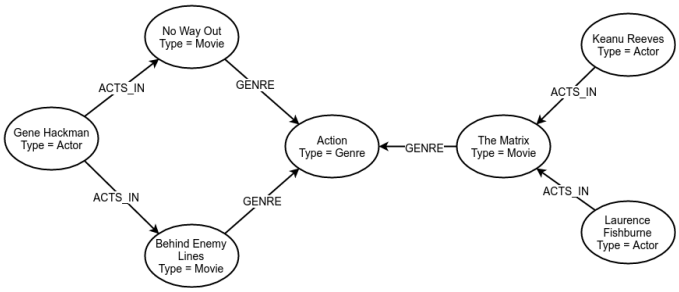


Fig. 1. Example of multi-relational graph with types on nodes and edges.

This concept has been used for some time in several different contexts, although until the last decade formal definitions have not been offered and extended to a more general framework: *Property Graphs*. In [7] a first formalization is given in order to provide a formal definition of *traversals* as a fundamental tool for the query task on this type of structures as support for Graph DataBases. In [8] the equivalence between Property Graphs and the Resource Description Framework specification, RDF [1], is presented.

As some neural network techniques for the construction of the embeddings will be used, next the specific model of *neural encoders* to be used ([10], [11]) in the construction of the embedding proposal is introduced. Any *Artificial Neural Networks* background where it supports on is ommited, and only some sketch about neural enconders functioning are presented.

When adding a hidden layer on a feedforward neural network, all the communication between the input and output layers passes through the hidden layer. Thus, if we are trying to approximate a function using this network, after adjusting the parameters of the network, we can assume that the hidden layer keeps the necessary information from the input to calculate the output data. Therefore, always depending on the function to be approximated, the hidden layer encodes the input data, and the weights (and biases) between these layers define a coding function[2].

If the number of units in the hidden layer used for encoding differs from the number of units in the input layer, then a dimensional change is made while encoding. This encoding process through neural networks is one of the methods that can be used to obtain embeddings that maintain some structural features present in the training sets (eg closeness relationships, or similarity) and that are related with the function the neural network approximates [11]. In order to use such an encoding to produce embeddings preserving semantic (topological) structures on a graph we need to choose an adequate function to approximate (it is common to approximate the identity function). Next section will present some related works that use similar ideas on other contexts to obtain semantic preserving embeddings.

III. Related Works

A. Semantic preserving embeddings of texts: Word2Vec

One application of neural encoders that has delivered impressive results has been made on corpus of texts and their grammatical and semantic contents. In [12], Mikolov et al. presented two neural architectures (*CBOW* and *Skip-gram*) under the generic name of *Word2Vec*, to learn vector representations of words trying to capture many of the grammatical and semantic properties of the words inside the corpus. Additionally, they reduce significantly the computational complexity of the learning process on very big corpus of texts (some other approaches in this direction were made previously, but all of them showing prohibitively high computational costs).

The general process passes through a first preprocessing on the source texts to extract the vocabulary of interest that will be projected (by common tasks on natural language processing, like taking the lemmas of the words, removing stopwords, etc.) and, for the words in this vocabulary, defining their neighborhoods (*contexts*) to be considered. For every occurrence of a word w in a text T of the corpus, the *context of w for that occurrence*, C, is defined as a set of words of T that are adjacent to that occurrence of w. The *size of the context* (or *window size*, i.e. how long the adjacency is considered) is a free parameter of the model, and has to be tuned to obtain the best results. In some way, and this is what Mikolov's work shows, the contexts store the local semantic information needed to preserve the global semantic relations that can be established along the text of the corpus.

Both architectures consist in feedforward neural networks with 3 layers (one only hidden layer), where input and output layers will have as many units as the size of the vocabulary, and each unit will be associated to each word following a *one-hot* structure: if $W = \{w_1, \ldots, w_k\}$ is the vocabulary, then the i-unit of these layers will encode the word w_i (Figure 2). But these architectures differ slightly in their functioning: while CBOW takes the contexts of the words as input and tries to output the word itself, Skip-gram takes the words as input and tries to output their contexts.

When doing a supervised training process to estimate the weights of the network, a set of training pairs in the form (w, C) will be used in the case of Skip-gram, and a set of training pairs of the form (C, w) in the case of CBOW. Note that every word can, and usually will do, appear with several associated contexts, and vice versa.

By learning the relations between words and their contexts, this model captures several types of similarities [13], both

[1]RDF is a data exchange standard model primarily focused on Web, widely used in the context of Semantic Web applications, which allows modeling of flexible multi-type relationships between elements with a set of triplets in the form $(subject, predicate, object)$ [9].

[2]Similarly, the part of the neural network connecting the hidden layer to the output layer can be seen as a decoding function.

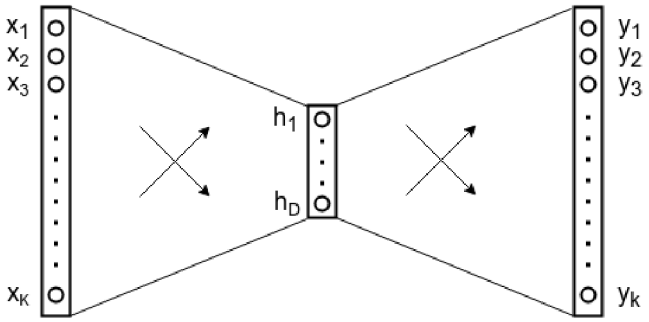


Fig. 2. Feedforward neural network structure used in CBOW and Skip-gram architectures.

functional and structural, and by taking the information stored in the hidden layer it provides an embedding into a vector space that reflects these similarities as (almost) linear vector structures and relations. It can capture similarities between words playing the same role in the sentences they are involved in, or even find more complex semantic relationships between concepts [14].

B. Other Related Works

Although different methodologies manage multi-relational data using different structures (RDF, Multi-Relational Graphs, Property Graphs, Relational Database Systems, etc.), methods which learn from multi-relational data can be grouped under the same field: Multi-Relational Learning.

Multi-relational learning can be divided in three blocks: (1) Statistical Relational Learning (SRL), where methods as Markov Logic Networks [15] can be found, which performs a multi-relational data codification using probabilistic models; (2) Path Ranking Methods ([16], [17]), which explicitly explore the relation space of multi-relational data using random walks; and (3) Embedding Based Methods, which obtain vector representations of multi-relational data through Matrix/Tensor Factorization ([18], [19], [20]), Bayessian Clusterization [21], [22] or Neural Networks ([23], [24], [25], [26]). Embedding Based Methods using neural networks have been demonstrated to be powerful tools due to their high scalability and their generalization possibilities. This section will focus in these last methods as they are closer to the proposal.

Recently, different methods trying to learn vector representation of entities and links from knowledge bases using neural networks have been developed ([24], [25], [27]). They all represent entities as vectors and represent relations as operators that combine the representations of the two entities that the relation connects.

In [25], a multi-relational data embedding that models different relation types by interpreting them as translations operating on the low-dimensional embeddings of the entities, is presented. This model tries to minimize the distances between pairs of nodes connected in the graph, and to maximize the distances between pairs of nodes not connected in the graph. The optimization process is carry out using gradient descent method. [24] follows a similar process but using a siamese neural network to improve the behaviour of the embedding, something that allows to work with graphs with more than 1 million elements. In [28], [29] a comparison of several variants of this kind for embedding multi-relational data (mainly extracted from WordNet and FreeBase datasets) is presented. In those papers the authors show results slightly better than the original implementations, and in the range of those obtained here, but must be taken into account that in this paper, the goal is not related with a fine tuning of the parameters and the accuracy results are reached with no optimization methods on them. A more guided search in the space of hyperparameters of the model will improve these results significantly.

Despite the relationship of these works with the one presented, the requirement imposed about maximizing unobserved edges goes against one of the pursued goals: since complete information in the datasets that will be used cannot be assumed, edges that are not observed may be due to a lack of information rather than a real absence. Indeed, predicting these unobserved edges is one of the reasons we are interested in an embedding that increases the analysis capability. In addition, most of the methods that try to learn vector representations of entities and links from a knowledge base are oriented to link prediction, conditioning the embedding to this particular task. In this way, the presented methodology pursues to obtain embeddings capturing the similarity between entities' contexts avoiding to be conditioned by a particular task. Consequently, after getting such an embedding, it can be used by several machine learning tasks on the semantic graph simultaneously.

In [26] the authors use neural encoders to represent the nodes of a classical graph (one single type of edge, not multi-relational) using a similar idea to Word2Vec, changing the contexts of the words by paths on the nodes. Later, they use a Skip-gram encoder to achieve the embedding. In this work the size of the networks are under 100 nodes, and the method suggests there should be deeper information available for every node, providing information about connections out of its direct neighborhood. In [30] a similar methodology is presented: A d-dimensional embedding of the entities from a classical graph is obtained by (1) learning $d/2$ dimensions that capture vicinity and (2) the rest of dimensions are learned by trying to obtain a embedding in which nodes that share similar context are placed closer.

In [31] a methodology that extends and joins ideas from [26] and [30] is presented. The authors show an algorithm that generates random walks on the graph guided by two parameters that allow to modify the strategy under the random walk construction in a spectrum ranging from Breath-First Search (BFS) to Depth-First Search (DFS). The random walks obtained by this procedure are used to generate contexts for every node in the graph and to train a neural network similar to that one in [26]. To embed a link they propose different operators that combines the representation of the two entities it connects: Mean, Hadamard Product and Distances. After this work, [26] can be seen as a special case of [31] where parameters guide the random walks through a DFS behaviour. In these works ([26], [30], [31]) the semantic has not been taking in account. Consequently, they must be considered as initial experiments showing the keen interest of embedding graphs into vector spaces using neural network architectures.

Some methodologies that embed multi-relational data into vector spaces have been developed, but in the presented methodology it is shown that using simple neural encoders

(CBOW) allows to train a neural network over big multi-relational graphs to obtain vector representations of their entities that keep the semantic characteristics of the original data under less restricted constraints and, consequently, with a wider application range.

IV. Embedding Multi-Relational Graphs in Vector Spaces

As it has been advanced in previous sections, the presented methodology uses neural encoders to embed the elements of a multi-relational graph into a n-dimensional vector space (where n is the number of units in the hidden layer of the encoder).

If semantic structure of a multi-relational graph is understood as the function which assings types to nodes and links (τ function) and in order to assess to what extent the semantic structure of the graph is preserved, an embedding using only the set of nodes into the vector space will be made. Thus, following the analogy offered by the algorithm Word2Vec, the vocabulary will be the set of nodes in the graph. A context, C, associated with a node $n \in V$ is obtained by randomly selecting (with repetition) a number of nodes connected to n by edges of the graph, regardless of the type of edge that connects them, its direction, or the types of the nodes. The number of nodes selected from its neighborhood will be the *Window Size*.

In the process, a training set of samples consisting of pairs (C, n), where $n \in V$ and C is one of its associated contexts, is generated. These samples will be used to train a CBOW-like neural encoder and, later, the activations from the hidden layer of the neural network are used as a representation of the input node (Figure 3).

In this procedure the *free parameters* of the model, to be tuned when doing real experiments to analyze its feasibility and efficiency, are: D, the number of units in the hidden layer, which determines the dimension of the vector space where the graph is embedding to; N, the size of the training set, is the number of pairs (C, n) used to train the encoder; and the *Window Size*.

The embedding obtained from the trained neural encoder will be denoted by $\pi : V \to \mathbb{R}^D$. Because of the binary nature of the edges in multi-relational graphs, a new embedding of the set of edges E, is obtained from the embedding of the nodes (it will be denoted also with π):

Definition: If $G = (V, E, \tau)$ is a Multi-relational Graph, and $\pi : V \to \mathbb{R}^D$ is an embedding for the set of nodes, V, the natural extension of π to the set of edges, $\pi : E \to \mathbb{R}^D$, will be defined as follows:

$$\text{if } l = (s, t) \in E, \text{ then } \pi(l) = \overrightarrow{\pi(s)\pi(t)} = \pi(t) - \pi(s)$$

Despite nodes (entities) and relations (edges) are projected into the same space, the former are projected as points while the latter are projected as vectors, consequently there will be more chances to find linear structures in the representations of relations, while other clustering structures will be common for nodes.

In order to check the good features of this embedding methodology, in the next section some experimental results that have been obtained using multi-relational graph representations of some well-known datasets are presented.

V. Some Empirical Evaluations

In this section results of some empirical evaluation of the proposal will be shown with two main goals: to verify that the obtained vector representations preserve the semantic features (types on entities and relations) of the original graph datasets, and to evaluate different applications that make use of this embedding proposal for classification and discovering tasks. It is important to note that only some of the possible experiments that this new methodology allows to perform are presented in this work. We are conducting several other experiments to have a more complete measure of the goodness of the proposal.

Also, it is key to emphasize that the first goal of the presented methodology is not to optimize the obtained models for any specific task, but to provide proofs that the resulting embeddings can be useful for several machine learning tasks simultaneously. A finer tuning in the parameters of the model will provide for sure better accuracy levels for the different semantic discovering tasks that the embeddings can be used for.

A first problem to provide a comparison between experiments is to determine an evaluation process to measure the validity of the different embeddings. For that, and taking into account that one of the goals is to preserve the existing semantic properties in the original data, an evaluation of the ability of the new representations to perform automatic classification tasks have been carried out: specifically, how τ function can be recovered from this embedding by using a classical supervised machine learning algorithm.

Note that the embedding algorithm that we present here does not receive information about types of nodes and edges of the graph (that is, it is not informed about τ function) as the contexts associated with different graph nodes are generated by randomly selecting a number of connected nodes, regardless their types (in nodes or edges). Hence, any reconstruction or prediction of this tagging function can be interpreted as a learning of the semantic information in the original graph by the embedding.

As noted, the obtained embedding for a specific dataset is not unique, and not only because the training process depends on the initial and the stop conditions, but because it also depends on the free parameters involved in the construction. Therefore, it is mandatory to analyze the features of the different embeddings that can be obtained by modifying these parameters on the learning process.

In this situation, experiments with two big semantic graphs obtained from well-known public datasets (**WordNet®**[32] and **The Movie Database**[3] (TMDb), that were reduced and converted into multi-relational graphs) have been carried out.

WordNet is a large lexical database of English words, where the entities (Figure 4(a)) are grouped into sets of synonyms and related by several criteria (4(c)). Only entities connected with relations of types shown in 4(b) have been considered, obtaining a graph with $\approx 97K$ nodes and $\approx 240K$ edges.

[3] https://www.themoviedb.org

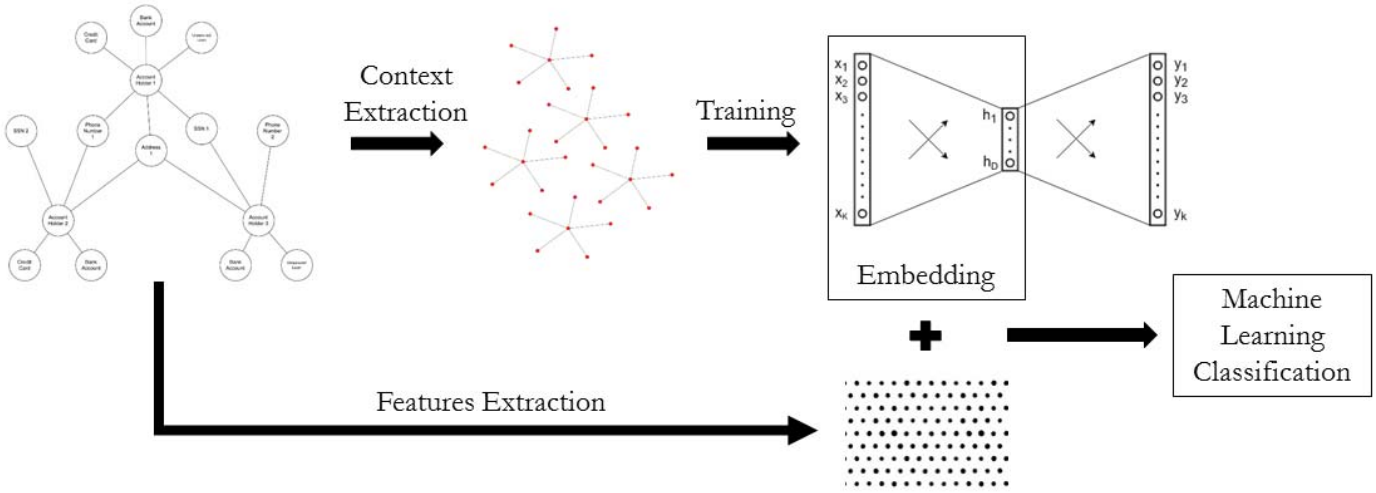
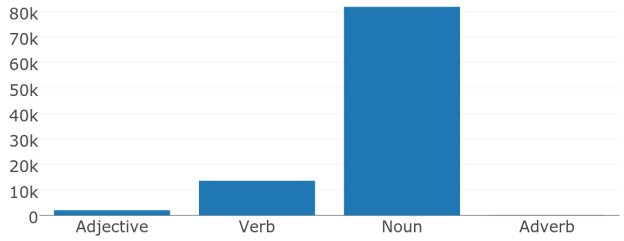
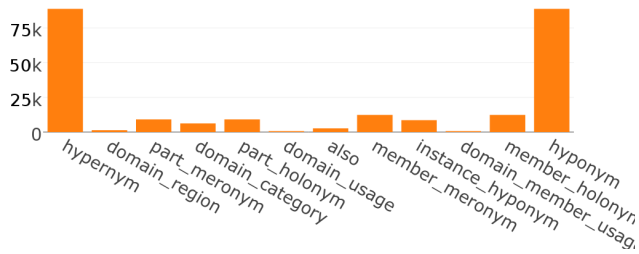


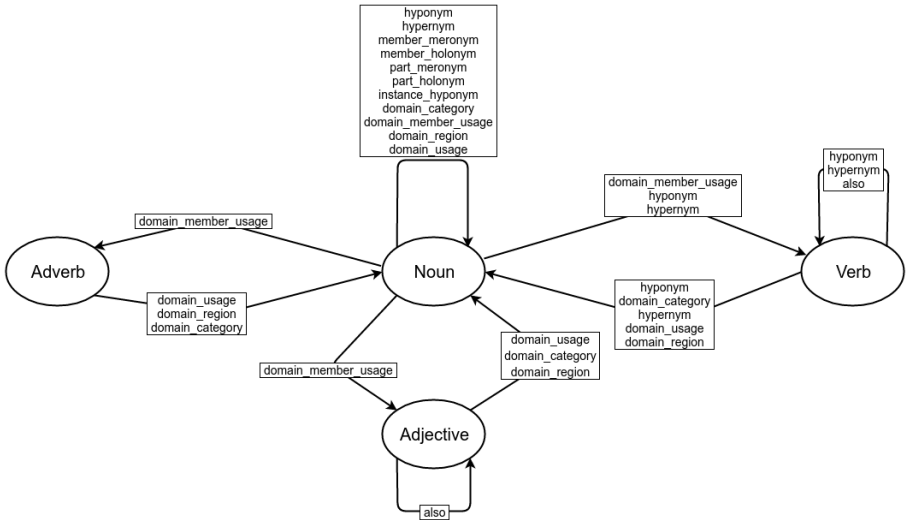
Fig. 3. Proposal methodology for semantic preserving embedding of multi-relational graphs as machine learning tasks feeder.



(a) Type Nodes Distribution



(b) Type Edges Distribution



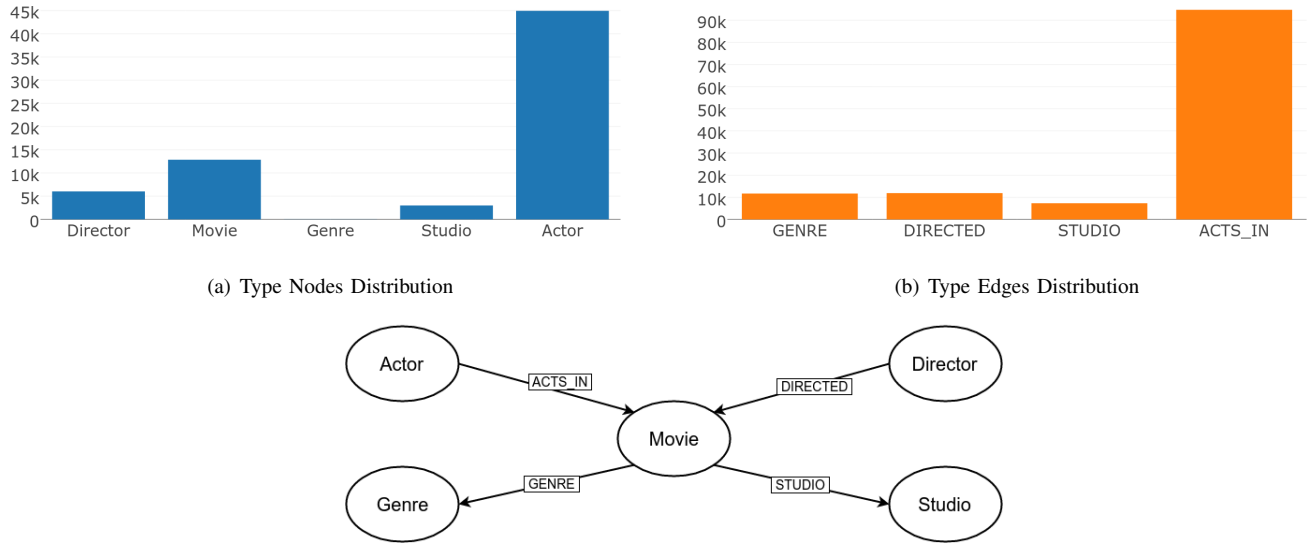
(c) Graph Scheme

Fig. 4. Distributions and Graph Scheme for WordNet Dataset.

TMDb is a database containing information about actors, films, and TV contents. Only entities (Figure 5(a)) connected by relationships from 5(b) have been considered, obtaining a graph with $\approx 66K$ nodes and $\approx 125K$ edges. Let us note that types $Actor$ and $Director$ are not disjoint, in the obtained graph there are 846 nodes with both types, something that will affect the classification performance.

For the CBOW architecture implementation a customized version of *Gensim*[4] library (v.0.12.4) for Word2Vec on Python

[4]https://radimrehurek.com/gensim



(a) Type Nodes Distribution

(b) Type Edges Distribution

(c) Graph Scheme

Fig. 5. Distributions and Graph Scheme for TMDb Dataset.

2.7 has been used, as well as Neo4j[5] (v.2.2.5) as Graph Database engine for storing and retrieving the graph datasets. The standard *K-Nearest Neighbors* (KNN) model [33] with $k = 3$ for all the tests has been selected, if this model can find good enough semantic patterns in the vector representations we can infer that the embedding process has been successful. Every experiment has been repeated 10 times in order to decrease the standard deviation of the errors.

For both classification tasks, *Type Nodes Prediction* (TNP) and *Type Edges Prediction* (TEP), some searches in the parameter space have been carried out in order to have some insight about how they affect the robustness and accuracy of the embeddings (Figure 6). WordNet shows a better performance ($\geq 95\%$) in TNP, while TMDb looks better ($\geq 85\%$) in TEP. After previous considerations, it can be confirmed that in both datasets and in both tasks the embedding process preserve the semantic information of the original datasets.

It is interesting the effect of the *Window Size* parameter in TNP. While in WordNet better accuracy is reached as the value of this parameter is increased, reflecting that more information about concurrent connections is more informative, in TMDb it looks like the best option is to consider only 3 related neighbors for every context and the behaviour is different when this parameter is odd/even. In any case, the optimum value for this parameter is greater than 1 in all the cases, what means that knowing about the concurrence of several connections help inferring the semantic. In TNP task, also this dataset shows that the model does not generalize well and some overfitting appears when increasing the size of the training set, maybe because of the size of the original graph.

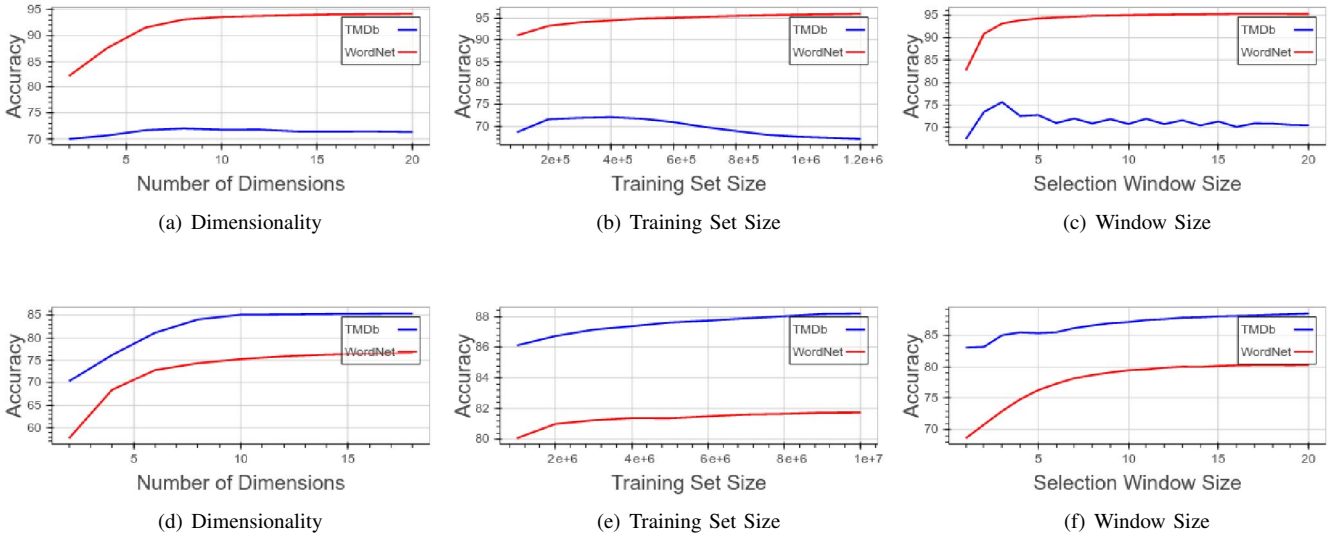
For a more detailed analysis, the confusion matrices for the best accuracy experiments in both datasets are provided in Figure 7, showing that the overlapping between *Actor* and *Director* decreases the performance of TNP in TMDb, and also in the TEP because of the confusion between *acts_in* and *directed* type edges. The problems with *Genre* arises from the fact that this entity type has very few associated nodes and the model could not learn to differentiate it from other entity types. Also, as Figure 8 shows, the edges associated to Genre entities are grouped in non-unique clusters, but depending on the specific genre it has, showing that they follow different semantic rules than the other entity types, that appear as clustered groups in the same representation.

In the next section, some comments about this low accuracy fact are provided. From our point of view, it does not reveal a weakness on the methodology but some ambiguity in the semantic structure of data sources. In this sense, Wordnet is a very curated dataset with well established semantic categories, while TMDb probably needs some extra work on the ontology behind its semantic structure in order to avoid some inconsistences and improve the quality and usefulness of the embeddings.

To stress the semantic capabilities of the proposed embedding, some more experiments have been performed. We present here some results related with *Entity Retrieval* [34], [35], [36], a set of challenging tasks that are considered to be among the hardest in *Information Retrieval* topic. Specifically, we will try to recover an entity in a relation knowing the source node of the relation, and the type of relation we are looking for. For example, in TMDb, starting with a *movie*, and looking for a relation of type *directed_by* we try to recover the *director* of the *movie*.

To achieve this task, a subset of links, $E' \in E$, has been selected in the original graph $G = (V, E, \tau)$ which will be deleted to obtain a new graph, $G' = (V, E \setminus E', \tau)$. Then, we learn the embedding from G' and try to obtain the target node

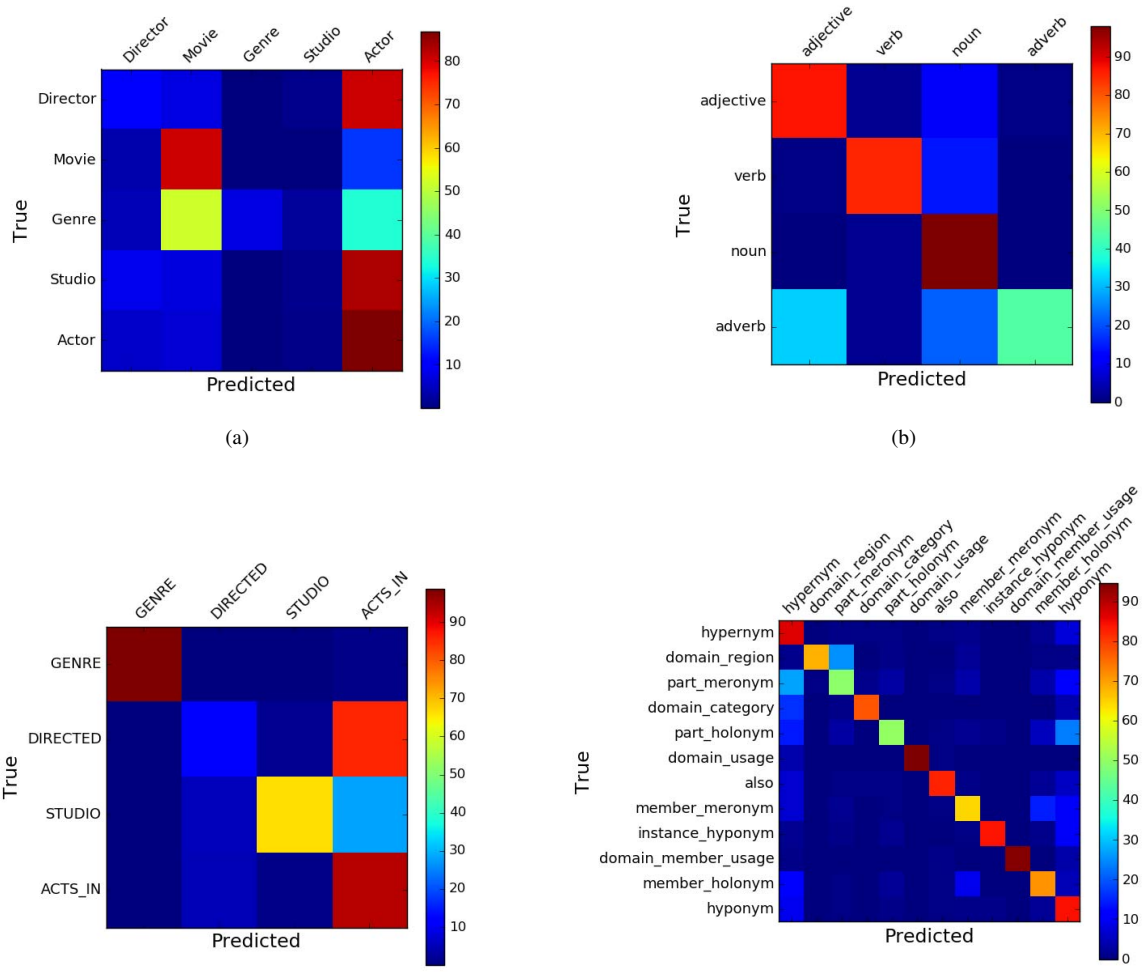
[5]http://neo4j.com



(a) Dimensionality (b) Training Set Size (c) Window Size

(d) Dimensionality (e) Training Set Size (f) Window Size

Fig. 6. Accuracy of TNP (top) and TEP (bottom) as function of the free parameters of embedding for WordNet (red) and TMDb (blue).



(a) (b)

(c) (d)

Fig. 7. Confusion Matrices for the datasets and tasks. From left to right: (a) TNP on TMDb, (b) TNP on WordNet, (c) TEP on TMDb, (d) TEP on WordNet.

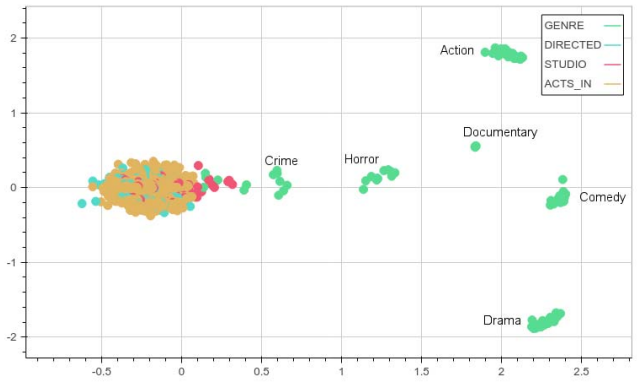


Fig. 8. 2D projection of embedded edges from TMDb Dataset.

TABLE I. TOP-10 ENTITIES RETRIEVED PROJECTING *hypernym* FROM DIFFERENT NODES IN WORDNET.

	spasm	*justification*	*neoconservatism*
1	ejection	reading	pruritus
2	rescue	explanation	conservatism
3	putting to death	analysis	sight
4	sexual activity	proposition	hawkishness
5	behavior modification	religious doctrine	coma
6	disturbance	accusation	scientific method
7	mastectomy	assay	autocracy
8	sales event	confession	judiciousness
9	instruction	research	reverie
10	debasement	discouragement	racism

of every link in E' using only the point representing its source node and the *representative vector* of its type.

Definition: Given a multi-relational graph $G = (V, E, \tau)$, and an embedding of the graph on a vector space, π, the *representative vector* associated to a link type $\alpha \in \tau(E)$, $\pi(\alpha)$, is the mean of all projections of links of the given type α. If we denote $E_\alpha = \tau^{-1}(\{\alpha\}) = \{l \in E : \ \tau(l) = \alpha\}$, then:

$$\pi(\alpha) = \frac{1}{\#(E_\alpha)} \sum_{l \in E_\alpha} \pi(l)$$

Given a link $l = (s, t) \in E'$ and its type $\tau(l)$, we will try to obtain t from the actual information $\pi(G')$, $\tau(l)$ and s. Following the expected linearization of the embedding on the set of relations/links, and using this representative vector as a real embedding for the deleted link, the predicted projection of the target node for s can be recovered by: $\pi(t_l) = \pi(s) + \pi(\alpha)$. Of course, it is unlikely this new $\pi(t_l)$ to be one of the projections of real nodes from G, for that, once $\pi(t_l)$ (the point representing the entity retrieved) is obtained, a ranking for the nodes in V can be built, ranking the entities according to the closeness of its projections to $\pi(t_l)$.

Table I shows the first 10 entities in the ranking obtained projecting $hypernym$ type link for several nodes in WordNet dataset. In this table the result is filtered to show only close entities tagged with type $noun$.

To evaluate the accuracy of the entity retrieval method presented, we will use *Mean Reciprocal Rank* metric [37], a common metric in *Information Retrieval* field used in many researches [38], [28]. This factor provides a statistic measure for evaluating any process producing a list of possible responses to a sample of queries, ordered by probability of correctness.

Figure 9 shows Entity Retrieval performance for WordNet dataset as a function of the size of the training set, N. In this case, the nodes have not been filtered according to their types, something that would improve the obtained results.

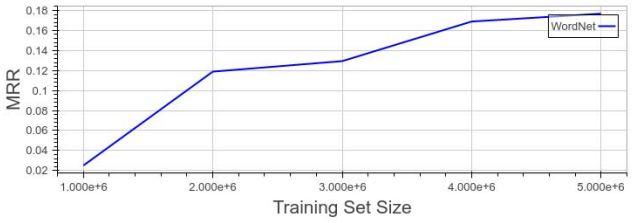


Fig. 9. Entity Retrieval performance as a function of Training Set Size in WordNet without filtering nodes according their types.

As observed, using the embedding proposal for Entity Retrieval task in WordNet dataset allows to obtain correct results over 0.18 with $N \geq 5M$. This result tell us about the quality of the representative vectors for WordNet graph: (1) if the representative vector of a link type allows to get the target of a given relation for a source node, then it should be little deviation in the vectors which represent links of the given type in the dataset and, (2) the set of source nodes and the set of target nodes for a given link type should form two clusters sufficiently disjoint in order to allow the representative vector to connect them. Again, this task is heavily conditioned by the semantic quality of the source dataset.

VI. CONCLUSIONS AND FUTURE WORK

In comparison with other machine learning tasks, there are only a few works that have used neural networks encoders for embedding multi-relational graphs, or similar graph structures, into vector spaces. The presented methodology obtains, using simple neural encoding architectures, vector representations that maintain the semantic and topological features of the original graph data. Moreover, with the obtained results, semantic links that remain hidden in the original data (due to incompleteness of the data stored in the graph or data inconsistency) could be discovered.

Although in this work embeddings for multi-relational graphs have been proposed, the presented methodology has been developed for the more generic framework of Property Graphs, where more information can be added in the elements by means of properties. In this way, local information about properties in nodes and edges can be used to increase the ability to preserve features in the embeddings and, consequently, machine learning processes provide better performance when applied to them. Some tasks related to this more generic structure that can be improved from this new perspective and that we are further developing in our current research, are briefly presented hereafter.

Geometrical features of the structures formed by nodes/edges in the new vector space can help assign types or missing properties to the elements of the graph dataset (using distance, linearity or clustering, for example, but not

only limited to this basis ones) or can even help to identify new relationships between elements not explicitly presented in the original graph. This action can be of fundamental interest for very common processes that are performed on big relational datasets, where incompleteness of data is an usual obstacle for information retrieval tasks.

Also, as has been noted after the different results obtained from the evaluation tests, the performance and accuracy of several machine learning tasks on vector representations of graph dataset (graph databases, overall) can provide information about the semantic structure of the dataset itself, not only about the algorithms in use. For example, the confusion of some nodes/edges in classification tasks can tell us that the schema behind the dataset needs to be adjusted in order to reflect the semantic features of the data correctly. In this sense, a detailed report about how the different types, properties and clusters are overlapped and confused in the embeddings provides a method to normalize the data schema itself, something which is lacking in current Graph Database proposals, where there is no normalization methods as it can be found in classical relational databases.

Since usual operations in vector spaces are widely used in current computing processors (CPUs and GPUs), this new representation can help the development of more efficient algorithms to analyze, repair and retrieve information from big sets of multi-relational data, in general, and more specifically, from big graph databases.

Finally, this work have mainly explored how linear vector structures can be used for retrieving information on multi-relational graphs, as entity retrieval experiments show, but it is more likely that looking for more complex structures on the projected space will improve dramatically the accuracy and usefulness of the embedding. Indeed, using a second layer of machine learning models after the neural encoding can improve results for several necessary tasks on information retrieval for semantic graphs. The results in this paper about using a neural encoder to generate semantic preserving embeddings of multi-relational graphs show that this is a research line that worths to be considered.

ACKNOWLEDGMENT

This work has been partially supported by TIC-6064 Excellence Project of the Junta de Andalucía and TIN2013-41086-P from Spanish Ministry of Economy and Competitiveness (co-financed with FEDER funds) and by Research and Graduate Studies Head Department of Central University of Ecuador.

REFERENCES

[1] M. Herbster, M. Pontil, and L. Wainer, "Online learning over graphs," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 305–312.

[2] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *In COLT*. Springer, 2004, pp. 624–638.

[3] R. I. Kondor and J. D. Lafferty, "Diffusion kernels on graphs and other discrete input spaces," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 315–322. [Online]. Available: http://dl.acm.org/citation.cfm?id=645531.655996

[4] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *IN ICML*, 2003, pp. 912–919.

[5] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning theory and kernel machines*. Springer, 2003, pp. 144–158.

[6] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*. O'Reilly Media, Inc., 2013.

[7] M. A. Rodriguez and P. Neubauer, "The graph traversal pattern," *CoRR*, vol. abs/1004.1001, 2010. [Online]. Available: http://arxiv.org/abs/1004.1001

[8] O. Hartig, "Reconciliation of rdf* and property graphs," *CoRR*, vol. abs/1409.3288, 2014. [Online]. Available: http://arxiv.org/abs/1409.3288

[9] D. Wood, M. Lanthaler, and R. Cyganiak. (2014, Feb.) RDF 1.1 concepts and abstract syntax. [Online]. Available: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[10] Y. Bengio, "Learning deep architectures for ai," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1561/2200000006

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[13] T. Mikolov, W. tau Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 746–751. [Online]. Available: http://www.aclweb.org/anthology/N13-1090

[14] T. Mikolov, J. Kopecky, L. Burget, O. Glembek, and J. Cernocky, "Neural network based language models for highly inflective languages," in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, ser. ICASSP '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 4725–4728. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2009.4960686

[15] M. Richardson and P. Domingos, "Markov logic networks," *Mach. Learn.*, vol. 62, no. 1-2, pp. 107–136, Feb. 2006. [Online]. Available: http://dx.doi.org/10.1007/s10994-006-5833-1

[16] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: ACM, 2014, pp. 601–610. [Online]. Available: http://doi.acm.org/10.1145/2623330.2623623

[17] N. Lao, T. Mitchell, and W. W. Cohen, "Random walk inference and learning in a large scale knowledge base," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 529–539. [Online]. Available: http://dl.acm.org/citation.cfm?id=2145432.2145494

[18] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 650–658. [Online]. Available: http://doi.acm.org/10.1145/1401890.1401969

[19] M. Nickel, V. Tresp, and H. peter Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, 2011, pp. 809–816. [Online]. Available: http://www.icml-2011.org/papers/438_icmlpaper.pdf

[20] M. Nickel, V. Tresp, and H.-P. Kriegel, "Factorizing yago: Scalable machine learning for linked data," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 271–280. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187874

[21] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda, "Learning systems of concepts with an infinite relational model," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, ser. AAAI'06. AAAI Press, 2006, pp. 381–388. [Online]. Available: http://dl.acm.org/citation.cfm?id=1597538.1597600

[22] I. Sutskever, J. B. Tenenbaum, and R. R. Salakhutdinov, "Modelling relational data using bayesian clustered tensor factorization," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1821–1828. [Online]. Available: http://papers.nips.cc/paper/3863-modelling-relational-data-using-bayesian-clustered-tensor-factorization.pdf

[23] A. Paccanaro and G. E. Hinton, "Learning distributed representations of concepts using linear relational embedding," *IEEE Trans. on Knowl. and Data Eng.*, vol. 13, no. 2, pp. 232–244, Mar. 2001. [Online]. Available: http://dx.doi.org/10.1109/69.917563

[24] X. Glorot, A. Bordes, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *CoRR*, vol. abs/1301.3485, 2013. [Online]. Available: http://arxiv.org/abs/1301.3485

[25] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," 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., 2013, pp. 2787–2795. [Online]. Available: http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *CoRR*, vol. abs/1403.6652, 2014. [Online]. Available: http://arxiv.org/abs/1403.6652

[27] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," 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., 2013, pp. 926–934. [Online]. Available: http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.pdf

[28] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Learning multi-relational semantics using neural-embedding models," *CoRR*, vol. abs/1411.4072, 2014. [Online]. Available: http://arxiv.org/abs/1411.4072

[29] Bishan Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *CoRR*, vol. abs/1412.6575, 2014. [Online]. Available: http://arxiv.org/abs/1412.6575

[30] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15. New York, NY, USA: ACM, 2015, pp. 1067–1077. [Online]. Available: http://doi.acm.org/10.1145/2736277.2741093

[31] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," 2016, cite arxiv:1607.00653Comment: In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. [Online]. Available: http://arxiv.org/abs/1607.00653

[32] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[33] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theor.*, vol. 13, no. 1, pp. 21–27, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1967.1053964

[34] H. Fang, R. R. Sinha, W. Wu, A. Doan, and C. Zhai, "Entity Retrieval over Structured Data."

[35] M. Sayyadian, A. Shakery, A. Doan, and C. Zhai, "Toward entity retrieval over structured and text data," 2004.

[36] N. Craswell, G. Demartini, J. Gaugaz, and T. Iofciu, *L3S at INEX 2008: Retrieving Entities Using Structured Information*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 253–263. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03761-0_26

[37] E. M. Voorhees, "The trec-8 question answering track report," in *In Proceedings of TREC-8*, 1999, pp. 77–82.

[38] K. Chang, W. Yih, B. Yang, and C. Meek, "Typed tensor decomposition of knowledge bases for relation extraction," 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*, 2014, pp. 1568–1579. [Online]. Available: http://aclweb.org/anthology/D/D14/D14-1165.pdf