Domain Specific Knowledge Graph Embedding for Analogical Link Discovery

Nada Mimouni, Jean-Claude Moissinac, Anh Tuan

Domain Specific Knowledge Graph Embedding for Analogical Link Discovery

Nada Mimouni

Cedric-Lab, CNAM Paris
Conservatoire National des Arts et Métiers Paris
France
Email: nada.mimouni@cnam.fr

Jean-Claude Moissinac, Anh Tuan Vu

LTCI, Télécom Paris
Institut Polytechnique de Paris
France
Email: jean-claude.moissinac,anh.vu @telecom-paris.fr

***Abstract*—General purpose knowledge bases such as DBpedia and Wikidata are valuable resources for various AI tasks. They describe real-world facts as entities and relations between them and they are typically incomplete. Knowledge base completion refers to the task of adding new missing links between entities to build new triples. In this work, we propose an approach for discovering implicit triples using observed ones in the incomplete graph leveraging analogy structures deducted from a knowledge graph embedding model. We use a neural language modelling approach where semantic regularities between words are preserved, which we adapt to entities and relations. We consider domain specific views from large input graphs as the basis for the training, which we call context graphs, as a reduced and meaningful context for a set of entities from a given domain. Results show that analogical inferences in the projected vector space is relevant to a link prediction task in domain knowledge bases.**

Keywords–Domain knowledge base; Context graph; Entity embedding; Neural language model; Analogy structure; Facts discovery.

I. INTRODUCTION

Mining graph structures with machine learning proved to be of great aid to extract hidden useful information in various domains. Recently, learning with graph embedding to encode unstructured data has attracted a lot of attention in research. This paper extends the work in [1] with particular focus on knowledge graphs.

General purpose Knowledge Bases (KB) such as YAGO, Wikidata, and DBpedia are valuable background resources for various AI tasks, for example, recommendation [2], web search [3], and question answering [4]. However, using these resources brings to light several problems which are mainly due to their substantial size and high incompleteness [5] as a result of the extremely large amount of real world facts to be encoded. Recently, vector-space embedding models for KB completion have been extensively studied for their efficiency and scalability and proven to achieve state-of-the-art link prediction performance [6], [7], [8], [9]. Numerous KB completion approaches have also been employed that aim at predicting whether or not a relationship, which is not in the knowledge graphs (KG), is likely to be correct. An overview of these models with the results for link prediction and triple classification is given in [10]. KG embedding models learn distributed representations for entities and relations, which are represented as low-dimensional dense vectors, or matrices, in continuous vector spaces. These representations are intended to preserve the information in the KG namely interactions between entities such as similarity, relatedness, and neighborhood for different domains.

In this work, we are particularly interested in adapting the language modelling approach proposed by [11], where relational similarities or linguistic regularities between pairs of words are captured. They are represented as translations in the projected vector space where similar words appear close to each other and allow for arithmetic operations on vectors of relations between word pairs. For instance, the vector translation

$$v(Germany) - v(Berlin) \approx v(France) - v(Paris)$$

shows relational similarity between countries and capital cities. It highlights the clear-cut analogical properties between the embedded words expressed by the analogy "$Berlin$ is to $Germany$ as $Paris$ is to $France$". We propose to apply this property to entities and relations in KGs as represented by diagrams (a) and (b) in Figure 1. The vector translation example is likely to capture the $capital$ relationship that we could represent by a translation vector $v(capital)$ verifying the following compositionnality [11]:

$$v(France) + v(capital) - v(Paris) \approx 0$$

We use the analogical property for KB completion and show that it is particularly relevant for this task. Our intuition is illustrated by diagrams (b) and (c) in Figure 1, where an unobserved triple can be inferred by mirroring its counterpart in the parallelogram. To the best of our knowledge, leveraging the analogy structure of linguistic regularities for KB completion has never been investigated prior to this work.

We consider applying such properties on excerpts from large KGs, we call context graphs, guided by representative entities of a given domain, where interactions between entities are more significant. Context graphs show to be the bearers of meaning for the considered domain and easier to handle because of their reduced size compared to source graphs.

This paper is organized as follows. Section II gives an overview of related work while Section III presents the global approach. Section IV recalls basic notions of the neural linguistic model. Then, Section V describes our approach to build context graphs and learn features for link prediction. Section VI details the evaluation protocol with expanded algorithms and discusses the initial results and finally, in Section VII we conclude.

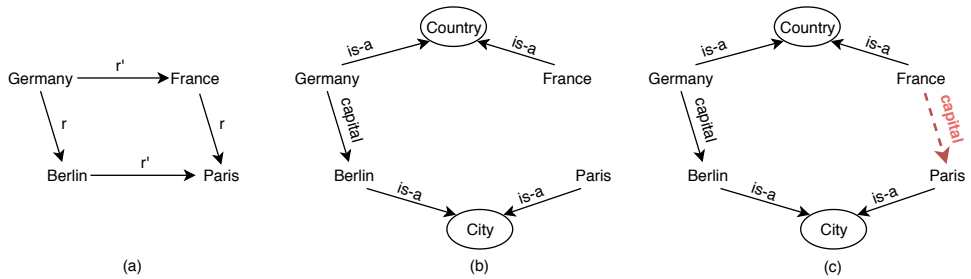

Figure 1. (a) Analogy relation diagram (parallelogram) between countries and capital cities. In KGs (b) and (c), r corresponds to the relation $capital$ and r' is decomposed into two type relations (is-a) to concepts $Country$ and $City$.

II. Related work

In what follows we give an overview of research work related to the embedding of knowledge graphs with a particular focus on approaches based on neural language models for the embedding of RDF graphs from which our approach is derived.

A. Knowledge Graph Embeddings

Several KBs exist such as YAGO [12], DBpedia [13], Freebase [14], or WordNet [15]. These knowledge bases contain billions of real-world facts and are, by nature, highly incomplete [5]. Many NLP tasks are based on these resources. The results of word sense disambiguation [16] and question answering [17], [4], for example, are directly impacted by the quality of those graphs.

Vector-space embedding models have a particularly advantageous performance among other statistical relational learning methods [18] proposed for knowledge graph completion or link prediction task [6]. Their main objective is to generalize the graph representation by calculating highly reduced dimensional vectors of entities and their relations. The first proposed algorithm, TransE [7], was inspired by a linguistic embedding model [11] where words are represented as vectors in an embedding space such that linguistic regularities present in the text reflecting relational similarities are captured and modeled as translations in the projected vector space.

Several variations have been subsequently proposed, for example, translation-based models (TransH [19], TransR [20]), bilinear models (DistMult [21], and ComplEx [22]), tensor factorization models (Rescal [23]), neural tensor networks (Ntn [24], [25]) among others [6], [26].

Formally, consider $\mathcal{E}$ to be the set of entities and $\mathcal{R}$ the set of their relations. A knowledge base $\mathcal{K}$ consists of a set of triples $(s, r, o) \in \mathcal{K}$ such that $s, o \in \mathcal{E}$, $r \in \mathcal{R}$, r is the relation between the subject s and the object o.

Each relation r is formulated as a linear map that transforms the subject s, represented as an embedding vector v_r, from its original position near the object o in the vector space. A score function $f(s, r, o)$ is defined by the embedding models to measure the uncertainty of each triple. For example, the TransE f function is as follows:

$$\|v_s + v_r - v_o\|_{l_{1/2}}; v_r \in \mathbb{R}^k$$

where the entities s and o are represented by vectors v_s and $v_o \in \mathbb{R}^k$; k is the vector's dimension. A general margin-based objective function is optimized with different optimization algorithms (e.g., the stochastic gradient descent, abbreviated SGD) to learn the following model parameters: entities vectors and relations vectors or matrices. This function is as follows:

$$\mathcal{L} = \sum_{\substack{(s,r,o)\in\mathcal{K} \\ (s',r,o')\in\mathcal{K}'_{(s,r,o)}}} [\gamma + f(s,r,o) - f(s',r,o')]_+$$

where $[y]_+ = max(0, y)$, γ is the margin hyperparameter; the set of incorrect triples is $\mathcal{K}'_{(s,r,o)}$ generated by altering the correct triple $(s, r, o) \in \mathcal{K}$.

The model Analogy presented in [8] has a particular focus on the study of analogical relations between triples. The approach proposes a formal solution to the problem of multi-relational embedding from an analogical inference point of view by defining analogical properties for entities and relation embeddings as well as optimizing algorithms (objective function) with respect to those properties. The authors argue that analogical inference is particularly advantageous for knowledge base completion and show that their framework improves state-of-the-art performances on benchmark datasets (e.g., WN18 and FB15K).

This assertion supports our decision to exploit the linguistic regularities that reveal analogies between words in order to apply them to entities and their relations in a graph. Since we are working on a particular application domain, we do not apply our method on benchmark datasets. We extract our data from large graphs to target only useful knowledge for this domain. Moreover, our method has the advantage of being based on a simple and efficient model, which directly offers a modeling of the analogy and optimally scales up without the need to define a complex framework for the study of regularities.

B. Neural Language Based RDF Graph Embedding

Another family of techniques for RDF graph embedding uses neural language models and paths in the graph to calculate vectors of entities and relations.

A general technique called Node2vec is proposed in [27]. It aims at creating embeddings for nodes in an (un)directed (a)cyclic (un)weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges with weights $\mathcal{W}$. The embeddings are learned using the Skip-gram model [28] trained on a corpus of sequences of nodes generated using the sampling strategy. The input graph is turned into a set of directed acyclic subgraphs with a maximum out degree of 1 using two hyperparameters for sampling: Return R (probability to go back to the previous node) and Inout Q (probability to explore new parts of the graph).

A closely related approach to our work is described in [29]. The RDF2vec approach uses the neural language model to generate embedding on entities from walks on two general knowledge bases namely DBpedia and Wikidata. Short random walks are created for the whole set of entities in an image of the KB at a given date. Walks using RDF graph kernels are also used on small test datasets due to scalability limitation. The trained models are made available for reuse. The approach we propose here differs in several ways. First, we consider undirected labelled edges in the RDF graph to adapt the neural language model that is compared to directed graph. Second, we use biased walks guided by the application domain to generate sequences that are compared to random walks. Third, rather than using object properties to build the sequences, we consider dataType properties and literals because we assume that they hold useful information for our application domain (e.g., dates, textual descriptions). Finally, and most importantly, we propose to handle scalability issues by contextualizing the input graphs assuming that more relevant information is centralized within a perimeter of α hops around our main entities (α is defined later).

III. KG Completion with Neural Language Model

In the following sections we present the approach we propose for link discovery in KGs. To this aim we use neural language models, namely CBOW and Skip-gram models, that we adapt to the embedding of entities from the KG. Our approach leverages analogical structures extracted from relational similarities between entities generated by the used neural language model. We show in the following how the analogical regularities, applied to entities from KGs, could be used to infer new unobserved triples from the observed ones to complete the original graph. We propose to work on contextualized RDF graphs focused on a specific domain that we extract from large general purpose KGs. We developed a set of algorithms for the construction of context graphs and for the preparation of test data and the evaluation process, which will be detailed as the approach stages progress.

IV. Preliminaries: Neural Language Model

Classic NLP systems and techniques treat words as atomic units represented by indices in a vocabulary without considering similarities between them. Texts are represented as a bag of words using binary feature vectors where each word corresponds to a vector index. Although being simple and robust, such techniques have limited performances in different NLP tasks due to the high dimensional and sparse data vectors generated. With recent advances in machine learning techniques, neural language models have been proposed to overcome these limitations by generating low dimensional distributed representations of words using neural networks. The main goal of word embedding (mapping from words to vectors of real numbers) approaches is to capture as much of the semantic and morphological information as possible from large amounts of unstructured text data. They explicitly model the assumption that words are statistically more dependent as they appear closer in the corpus.

Many different types of models where proposed; however, the earliest suffered from inefficient training of the neural network as they become computationally very expensive on large data sets [30], [31], [32]. An efficient neural model has been proposed in [11], [28], the Word2vec model, and gained wide popularity due to its simplicity. Word2vec is a two-layer neural net model for learning distributed representations of words while minimizing computational complexity. Two different model architectures for parameter learning are used, the Continuous Bag-of-Words model (CBOW) and the Skip-gram model.

A. Continuous Bag-of-Words Model

The intuition behind the CBOW model is the fact of predicting one word while considering a multi-word context within a given window in order to preserve information about the relation of the target word to other words from the corpus. The architecture is shown in Figure 2.

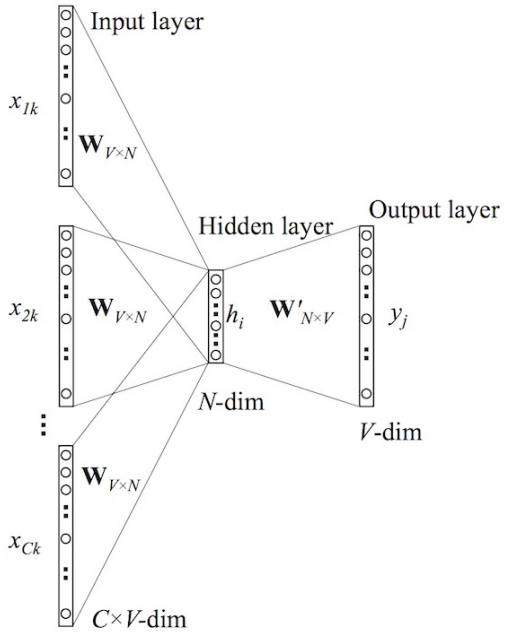

Figure 2. CBOW architecture

The input layer is composed of surrounding words within a given context window c. Their input vectors are calculated from the input weight matrix, averaged and then projected in the hidden layer (or projection layer). The output weight matrix serve to calculate a score for each word in the vocabulary as the probability of being a target word. The architecture of the neural network is formally given by:

- a set of vectors x_i representing training words $w_1, w_2, w_3, ..., w_C$ in the input layer;
- weights matrix W of size $V \times N$ from input to a hidden layer where V is here the entire words vocabulary, and N is the dimension of the hidden layer;
- weights matrix W' of size $N \times V$ as from a hidden to output layer;
- a softmax function as a final activation step.

The goal is to calculate the probability distribution $p(w_i|w_{i-c}...w_{i+c})$, the vector representation of the word with index i, using the softmax function:

$$p(w_i|w_{i-c}...w_{i+c}) = \frac{exp(\overline{v}^T v'_{w_i})}{\sum_{w=1}^{V} exp(\overline{v}^T v'_w)}$$

where V is the entire words vocabulary, v'_w is the output vector of the word w and $\overline{v}$ is the averaged input vector of context words:

$$\overline{v} = \frac{1}{2c} \sum_{-c \leq j \leq c, j \neq 0} v_{w_{i+j}}$$

The objective of the CBOW model is then to maximize the average log probability:

$$\frac{1}{C} \sum_{i=1}^{C} \log p(w_i|w_{i-c}...w_{i+c})$$

B. Skip-gram Model

The Skip-gram Model is the opposite of the CBOW model where it is matter to predict c context words from one target word in the input. The architecture is shown in Figure 3.

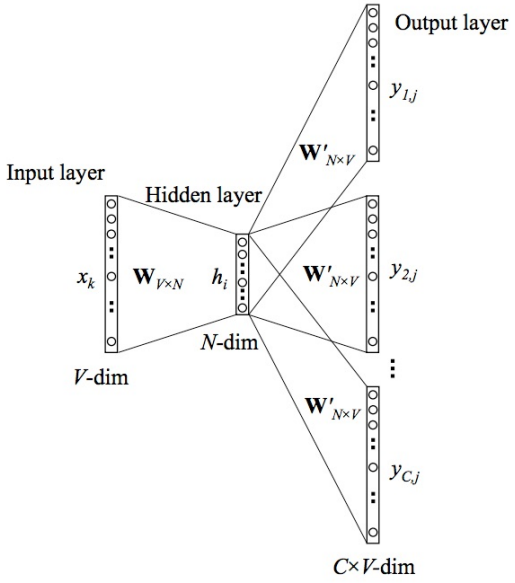

Figure 3. Skip-gram architecture

Formally, given a sequence of training words $w_1, w_2, w_3, ..., w_C$, the objective function to maximize is the following average log probability:

$$\frac{1}{C} \sum_{i=1}^{C} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{i+j}|w_i)$$

where $-c$ and c are the limits of the context window, word w_i is every word from the working corpus. The first step is then to obtain the hidden layer as:

$$h = W^T x := v_{W_I}^T$$

On the output layer, c multinomial distribution is computed sharing the same weights between the output panels from the hidden to output layer weights matrix W'. The activation of the output use the softmax function to calculate the probability $p(w_{i+j}|w_i)$ as follows:

$$p(w_{i+j}|w_i) = \frac{exp(v'^T_{w_{i+c}} v_{w_i})}{\sum_{v=1}^{V} exp(v'^T_{w_v} v_{w_i})}$$

where V is the entire words vocabulary, v_w is the input vector and v'_w is the output vector of the word w. More details about these techniques can be found in [33], [34].

So far, the models presented above, both CBOW and Skip-gram, are in their original forms without any efficiency optimization applied. Computed in a straightforward manner, the computational complexity of those algorithms corresponds to the size of the vocabulary. In fact there exist two vector representations for each word in the vocabulary, v_w and v'_w, the input and the output vector, respectively. During the update process it is required to iterate through every word w_j in the vocabulary, compute their probability prediction, and their prediction error to finally update their output vector v'_j. Doing these computations for all of the words for every training instance makes the learning of output vectors considerably expensive and the whole learning process impractical to scale up for large vocabularies. Two optimization techniques were proposed to solve this problem by limiting the number of output vectors that require an update per training instance, hierarchical softmax and negative sampling [11].

As stated by the authors in [11], negative sampling outperforms hierarchical softmax on the analogical reasoning task, which is our main concern in this paper. They also argue that the linearity of the Skip-gram model makes its vectors more suitable for such linear analogical reasoning. This will guide our parameters calibration for training the word2vec model in the evaluation step.

V. Approach

The approach we propose in this work adapts the neural language model word2vec for the embedding of knowledge graphs. It benefits from the properties offered by this model, which explicitly assume that closer words in text documents are statistically more dependent. For RDF graphs, words are replaced by entities and relations between entities. Sequences of entities and relations should be generated from the RDF data in order to apply the word2vec model as on sentences of words. After graph conversion, we can train the neural language model to represent entities and relations between them as vectors of numerical values in a latent feature space. These vectors can directly be used to perform analogical inferences for link prediction. In our approach, we work on excerpts from large knowledge graphs, we call context graphs, which represent condensed information from a given domain. Many reasons motivate our choice; we discuss them in the following subsection and show how such context graphs are built and explored.

A. Building Context Graphs

For the remainder of this article, we will study a knowledge graph extract; we name it Context Graph (CG). We first start by explaining why CGs and, more generally, extracts from knowledge graphs, are interesting to study and how they can be constructed.

1) Motivation For Context Graphs: In the semantic web community, the concern has arisen to work with extracts from large graphs, rather than with large graphs themselves. Various motivations are encountered for this. To begin with, a query on a graph, in particular a `CONSTRUCT` query, creates a 'view' on this graph that constitutes an extract on which we generally wish to do targeted operations: insertion into another graph, display, processing, etc. We may want to put aside the obtained extract, for example, because the corresponding request is long to process and therefore costly in performance for the triple store of the original graph.

An extract targeted on a class or a category of objects can be useful, for example, for an analysis and an improvement of the quality of these types of objects in the large graph: used properties, missing properties, consistency with a schema, etc.

We can assume that queries on an extract can execute faster than the same queries on the full graph (and give the same results for a category of queries targeted by the extract). This can be a way to bypass timeout problems on graphs such as DBpedia. We have no precise benchmarks on this subject. To create such benchmarks; it is necessary to install a copy of DBpedia on the same computer, then build the extract, and test a list of queries, which is out of scope in this work.

The same is true for other processings on graphs. For example, in [29], building a model from DBpedia may take up to several days with a powerful machine [35] while building a model for this work takes 11 minutes on a laptop with four GB of memory and a quad-core 2.8 GHz processor (see more details at the end of the evaluation section). We argue that with smaller and more focused graphs as our Context Graphs, it is easier and more reliable to apply inference methods.

Extracts can also be shared more easily to contribute to students work. A dated and versioned excerpt can also be a significant dataset to support research work by having access to the data that have been used for this work. This is particularly useful if you are working on constantly evolving data graphs such as Wikidata.

2) Context Graphs Building Process: We define a Context Graph as a sub-graph of a general $\mathcal{KG}$ (e.g., DBpedia) representative of a domain D.

1) The first step to build a CG is to identify a list of seeds defining the domain. A seed is an entity from $\mathcal{KG}$ corresponding to a concept that is considered relevant for D. For example, if the concept is '*Musée du Louvre*', the corresponding entity in DBpedia is <http://dbpedia.org/resource/Louvre>. In some domains this list is obvious as for museums, hotels, or restaurants. In general, the common practice is to rely on a reference dataset (such as IMDB for cinema).
2) The second step extracts from $\mathcal{KG}$, the neighborhood for each seed within a given depth filtering useless entities or predicates (not informative for D) and returns the final CG as the union of elementary contexts. We use the CG in the following algorithm as the basis for the embedding model. For example, DBpedia-fr uses the predicate <http://dbpedia.org/ontology/wikiPageRevisionID>, which is maintenance information about the source of the entity. The node revision ID is inadequate for our work and thus we consider it as out of our target domain.

Algorithm 1: *CONTEXT BUILDER*

```
1  Function ContextBuilder(KG, seedsEntities,
     radius, filteredEntities)
       Input  : A knowledge graph KG
       A neighborhood depth to reach radius
       A set of entities which are used as seeds
       seedsEntities
       A set of entities which are excluded from the seeds
       filteredEntities
       Output : Context Graph context
2      level ← 0
3      context ← ∅
4      while level < radius do
5          newSeeds ← ∅
6          foreach s ∈ seedsEntities do
7              C_s ← FindNeighbors(KG, s)
8              context ← context ∪ C_s
9              newSeeds ← newSeeds ∪
                EntityFilter(C_s, filteredEntities)
10         end
11         level ← level +1
12         seedsEntities ← newSeeds
13     end
14     context ← context ∪ AddClasses(KG,
         Entities(context))
15     return context
```

We create the algorithm `CONTEXT BUILDER` (Algorithm 1) to build a context graph $context$ from a knowledge graph $\mathcal{KG}$ for a given domain D. For a set of seeds ($seedsEntities$), `findNeighbors`(s) extracts a neighboring context C_s from a knowledge graph $\mathcal{KG}$ for each seed s. The final context, $context$, is updated adding C_s. A list of new seeds, $newSeeds$, is updated with the new collected entities after filtering the terminal nodes with the `EntityFilter` method. The exploration depth $level$ is incremented by 1 at each step up to the desired $radius$ limit. At the end of process, the resulting context $context$ is expanded with the classes of entities extracted from $\mathcal{KG}$ by the methods `AddClasses` and `Entities`.

The purpose of the `EntityFilter` method is to eliminate a certain number of entities that seem useless for a given application. The entities to be filtered are given in the list *filteredEntities*. In our application, examples of elements in this list are entities belonging to the T-Box due to their very general nature (e.g., in DBpedia: dbo:Building, dbo:Place or owl:Thing) and structuring nodes (e.g., DBpedia-fr pages layout, <http://fr.dbpedia.org/resource/Modèle:P.>). Those nodes introduce a great deal of noise without bringing relevant information for the targeted domain. For example, in our experiments on DBpedia, we found $3,486$ entities built on <http://fr.dbpedia.org/resource/Modele:????> giving rise to $2,692,515$ links and $101,235$ links to <http://www.w3.org/2004/02/skos/core#Concept>.

The `Entities` method finds all the entities in the created context graph. The `AddClasses` method adds the classes of

all the entities after finding them in the source graph. At this stage, we also add the ontology (the T-Box) that organizes these classes, if it exists. Thus, we have a knowledge of the nature of the manipulated entities, which allows us to draw conclusions on the basis of these classes as we will see below. It should be noted that the concept of a class can differ from one knowledge graph to another; this is, for example, the case for Wikidata that uses its own concepts, while DBpedia uses the concepts defined by RDFS [36].

B. Context Graph Sub-structures

We transform the entities and relations in the CG as paths that are considered as sequences of words in natural language. To extract context graph sub-structures, we use the breadth-first algorithm to get all the graph walks or random walks for a limited number N.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an RDF graph where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of directed edges. For each vertex $v_x \in \mathcal{V}$, we generate all or N graph walks P_{v_x} of depth d rooted in the vertex v_x by exploring indifferently direct outgoing and incoming edges of v_x. We iteratively explore direct edges, both incoming and outgoing, of neighbors v_{x_i} of vertex v_x until depth d is reached. The paths after the first iteration follow this pattern $v_x \rightarrow e_i \rightarrow v_{x_i}$ where $e_i \in \mathcal{E}$. The first token of each path $p \in P_{v_x}$ is the vertex v_x followed by a sequence of tokens that might be labels of edges or vertices. The final set of sequences for $\mathcal{G}$ is the union of the sequences of all the vertices $\bigcup_{v_x \in \mathcal{V}} P_{v_x}$.

Hereafter are some examples of entity sequences that have been extracted from DBpedia using graph walks of depth 2, 4 and 8, respectively:

dbr:Walker,_Texas_Ranger → dc:subject → dbr:Category:Martial_arts_television_series

dbr:Walker,_Texas_Ranger → dbo:creator → dbr:Paul_Haggis

dbr:Walker,_Texas_Ranger → dbp:country → United States

fdbr:Maison_de_Balzac → dbo:wikiPageWikiLink → fdbr:Honoré_de_Balzac → fdbp:auteur → Patricia Baudouin

fdbr:Maison_de_Balzac → dbo:wikiPageWikiLink → fdbr:La_Comédie_humaine → dbo:wikiPageWikiLink → fdbr:Venise → dbo:wikiPageWikiLink → fdbr:Paul_Bourget → dbo:wikiPageWikiLink → fdbr:Catégorie: Grand_officier_de_la_Légion_d'honneur

The following correspondences are used for properties:

fdbr	http://fr.dbpedia.org/resource/
dbo	http://dbpedia.org/ontology/
dbp	http://dbpedia.org/property/
dc	http://purl.org/dc/terms/

C. Feature Learning

Next, we train the word2vec neural language model, which estimates the likelihood of a sequence of entities and relations appearing in the graph and represents them as vectors of latent numerical features. To do this, we use the continuous bag of words (CBOW) and Skip-gram models as described in Section IV. CBOW predicts target words w_i from context words within a context window c while Skip-gram does the inverse and attempts to predict the context words from the target word. The probabilities $p(w_i|w_{i-c}...w_{i+c})$ and $p(w_{i+j}|w_i)$ are calculated using the softmax function.

Once the training is finished, all entities and relations are projected into a latent feature space where semantically similar entities are positioned close to each other. Moreover, we can perform basic mathematical operations on the vectors in order to extract different relations between entities.

D. Analogical Inference

In this last step, we extract analogical properties from the feature space to estimate the existence of new relationships between entities. We use the following arithmetic operation on the feature vectors (entities of Figure 1): $v(Berlin) - v(Germany) + v(France) = v(x)$ which we consider is solved correctly if $v(x)$ is most similar to $v(Paris)$. On the left-hand side of the equation, entities contribute positively or negatively to the similarity according to the corresponding sign. For example, $Germany$ and $France$ having the same type $Country$ contribute with different signs, $Berlin$, of a different $City$ type, contribute with the opposite sign of the corresponding Country. The right-hand side of the equation contains the missing corner of the diagram, which remains to be predicted. We then use cosine similarity measures between the resulting vector $v(x)$ and vectors of all other entities of the same type in the embedding space (discarding the original ones of the equation) in order to rank the results.

In this article, the assessments focus on relationships between museums and artists. We will therefore be interested in equations on the model $v(artist1) - v(museum1) + v(museum2) = v(x)$. In addition, we verify that x has the good type, here Artist.

E. Semantic filtering

Analogical inference returns as result a set of entities who's vectors are similar to the reference vectors.

It is possible to acquire entities which do not have the required type (City in the example). To handle this, we apply a semantic filtering to keep only entities of the expected type. Doing so, we use a combination of embeddings and semantics to achieve filtered results to explore for evaluation.

VI. Experimental Evaluation

In what follows we present the experimental evaluation of the proposed approach with a case study in the tourism field, particularly the museums of Paris. Data is extracted from general purpose KBs on which the approach is applied and evaluated against ground-truth of artists and museums analogies.

A. Case Study

We test our approach on a sub-graph of DBpedia representing a target domain; here we chose museums of *Paris Musées*, a federation of museums in Paris. We propose to address the scalability issue by contextualizing the input graphs, assuming that more relevant information is centralized within a perimeter of α hops around the main entities of this domain. We used $\alpha = 2$ as suggested by [37]. We build our $\mathcal{KG}$ as the union of individual contextual graphs of all entities representing the input data from the cultural institution *Paris Musées* (12 sites). We identify each site by its URI on DBpedia-fr after an entity resolution task (in the following, we denote the URI http://fr.dbpedia.org/resource/entity shortly as dbr:entity).

Obtaining URIs is a classic Entity Linking problem. As we are only dealing with a small number of elements, we used a

rudimentary method: starting from the name of each museum, a search in DBpedia-Fr of entities with the same label - by testing different variants of a character case - we get a set of propositions that are validated by a person. It is also a person who carries out a search in DBpedia-Fr for entities that could not be found by this process. Of course, for a greater number of entities describing a domain, it would be necessary to use an advanced method of Entity Linking and a context for each entity to avoid any ambiguities [38].

The final graph contains $448,309$ entities, $2,285$ relations, and $5,122,879$ triples. To generate sequences of entities and relations we use random graph walks with $N = 1000$ for depth $d = \{4, 8\}$. We also consider for each entity all walks of depth $d = 2$ (direct neighbours).

The context graph for *Paris Musées* was constructed with a depth of 2. The core of the context therefore has a depth of 1. Studying the impact of choosing this depth is beyond the scope of this article. A *blacklist* has been created essentially comprised of all the elements of the *T-Box*, considered as terminal nodes. For example, if a node brought us to `owl:Thing` and we followed the links from there, clearly, we would bring back $1,527,645$ entities that are not necessarily related to our domain.

Table I gives a description of a context graph extracted from DBpedia-fr for the depth $N = 2$. We test different settings for the constitution of such a graph. The values in this table are therefore only indicative.

TABLE I. Description of a context graph $\mathcal{CG}$ extracted from DBPedia-fr with $N = 2$

	Context $\mathcal{CG}$	DBPedia-fr	%
Distinct entities	448309	10515624	4.2
Distinct predicates	2285	20322	11.24
Links	5122879	185404534	2.76
Links by entity (mean)	11.42	17.63	

It is normal that there are fewer links per node in $\mathcal{CG}$ than in DBPedia-fr, since, by construction, we have eliminated some links that are not very informative in our application framework as explained above.

We therefore have a number L of links 36 times lower and a number S of vertices 23 times lower in the $\mathcal{CG}$ that in $\mathcal{KG}$. On an algorithm in $O(L + S)$, such as the Breadth-first search, we can anticipate a gain factor of around 30, which can strongly contribute to the applicability of a large number of methods. The gain can become considerable on algorithms such as those of the shortest path search between two nodes, when it is also a matter of giving weights to edges, the complexity of which is in $O(S^2)$.

We then train the Skip-Gram word2vec model on the corpus of sequences with the following parameters: window size $= 5$, number of iterations $= 10$, negative samples $= 25$ (for the purpose of optimisation), and dimension of the entities' vectors $= 200$. We use gensim implementation of word2vec [39]. The parameters values are inspired by those used for the training of the RDF2vec model [29] and, for the remaining ones, the default values, as proposed in [11], are used. Moreover, we trained our model with the CBOW method and with a larger vector dimension (500). We notice in general better performance with the Skip-Gram method, but cannot make any assertion about the vector dimension.

Our method cannot be evaluated against others using benchmark datasets such as FB15K, WN18 [7], [8]. It requires defining a context and extracting a subgraph from it; none of the other methods use such a context in the available evaluations and our proposal is strongly linked to the definition of such a context. So far, we have no knowledge of any experiments that rely on DBpedia-Fr. Some rely on DBpedia, but French museums are poorly represented in DBpedia.

B. Evaluation Protocol

Existing benchmarks for testing analogy tasks in the literature are designed for words from text corpora. To the best of our knowledge, using a language model driven analogy for link prediction in knowledge graphs has not been investigated yet.

To evaluate our approach, we build a ground-truth for an analogy between entities in the KG. Each entry corresponds to a parallelogram as described in Figure 1, with one unobserved triple in the $\mathcal{KG}$. For each entity, corresponding to a museum site in our application, we collect a list of well-known artists for this site as follows: find in DBpedia-fr the list of artists (dbo:Artist) or otherwise, individuals (dbo: Person) who are associated with the site. For some sites, we manually create the list, for example, by searching for well-known artists for a museum on the website [40].

The data used for the evaluation is made up of two tables indexed by the museum identifier. The first table associates a museum with an artist considered as key for this museum; for example, Zadkine for the Zadkine museum. The second table associates a list of people - essentially artists - important for this museum. These lists were established by consulting the list of artists from the Paris Musées Collections site [40] and consolidating this information with the corresponding Wikipedia pages. For each artist and museum in the data, we use the URI in DBpedia-Fr.

In the following paragraphs of this subsection, we first present the method used to obtain the results. Then, we propose two ways to evaluate them.

We create the algorithm `FIND SIMILARS` (Algorithm 2) which builds a two-dimensional table. The first dimension is the museum for which we search artists. The second dimension is the museum for which we know a principal artist. As the result, each cell of the table contains a list of proposed artists.

The first test is formalized as follows: given a pair ($museum_a$, $artist_a$ = main artist for $museum_a$), for each $museum_b$, $a \neq b$, we search for a list of artists $artist_b$, which has a similar role as the role of $artist_a$ for $museum_a$ (here, we consider the artist who has the most works exhibited or who is the most present in the museum, that we call the main artist). Then, we assess the pertinence of each $artist_b$ for $museum_b$.

The evaluation test aims at discovering $artist_a$ for $museum_a$ considering a known triple <$museum_b$, $artist_b$> while varying b and measuring the mean of the returned results. We use conventional metrics: Mean Reciprocal Rank (MRR) and the number of correct responses at a fixed rate (Hits@).

Algorithm 2: *FIND SIMILARS*

Function `FFindSimilarArtists(`$MainArtists$, $Museums$,`)`
  Input :
  List of main artist uris, one by museum $MainArtists$
  List of museum uris $Museums$
  Max of returned similar entities Max
  Output :
  Table of proposed similar artists for M_j given (M_i, A_i) $SimilarArtists$ [][]
  foreach $M_i \in Museums$ **do**
    $A_i \leftarrow MainArtists\ [M_i\ ]$
    foreach $M_j \in Museums$ **do**
      $Similars\ [M_i\ ][M_j\ ] \leftarrow$ `FindSimilars(`A_i, M_i, M_j, Max`)`
      $SimilarArtists\ [M_i\ ][M_j\ ] \leftarrow$ `TypeFilter(`$Similars\ [M_i\ ][M_j\ ]$, $tArtist$`)`
    end
  end
  return $SimilarArtists$

The evaluation protocol is as follows: for each URI M_i, URI of a museum, let URI A_i be the URI of the first artist identified for M_i, consider all $M_j \mid j \neq i$, and find the top most similar entities of the predicted vectors with positives = $[A_i,M_j]$ and negative =$[M_i]$. In the list of results, we filter by type $Artist$, and then examine the intersection with artists A_l associated with M_j.

In other words, we fix the target M_j and look at the obtained results by varying the reference pair (M_i, A_i). We consider the relevance of the returned result on M_j as the average relevance over all pairs (M_i, A_i). Algorithm `EVALUATION A` (Algorithm 3) shows the steps of the first evaluation test.

In Algorithm 3, the `EvaluateAgainstRef` function uses the list of proposed artists and the reference list for museum M_i. This function checks whether the artists expected by the reference list have been found in the 3^{rd} -or 5^{th} or 10^{th}- first returned results, and compares the rank of found results with the reference ranking (with MRR metric). The `Mean` function averages the obtained results for the different M_i tested.

It is worth noticing that we frequently find loosely qualified links between museums and artists; such links are very common in DBpedia and use the property `wikiPageWikiLink` representing an untyped link. Subsequent work is required to qualify them.

C. Results

Table II shows the results of MRR and Hits@$\{3, 5, 10\}$ (%) for $d = \{4, 8\}$ and $N = 1000$. The final row of Table II with columns $d = 8$ shows the impact of considering longer paths on the performances of the approach. In fact, longer paths capture richer contexts for entities and results in better vectors estimation by the neural language model.

We compared our approach with the one presented in [29], which creates a model, modelDB, for all entities in DBpedia. For each entity in our ground-truth built on DBpedia-fr, we look for its equivalent in DBpedia and verify that it is contained in the vocabulary of modelDB built with $d = 4$. Only 7 out of 12 museum entities are in modelDB, as well as their first associated artist among others. The analogy tests return globally poor results. ModelDB were unable to retrieve relevant entities in the top 100 returned answers, as was the case for our model trained on the CG, without any improvement even if extended to top 5000. This result is to be expected when we look at the following table, which shows that our CG has better coverage of the ground-truth domain entities, mainly artists, compared to DBpedia.

Algorithm 3: *EVALUATION A*

Function `MuseumsFocusedEvaluation(`$SimilarArtists$, , $Museums$`)`
  Input :
  Table, result of Algorithm 2 $SimilarArtists$
  Reference lists of artists, one list by museum
  List of museum $Museums$
  Output :
  List, mean of MRR values by museum $MMrr$
  List, mean of Hit@3 values by museum $MHit3$
  List, mean of Hit@5 values by museum $MHit5$
  List, mean of Hit@10 values by museum $MHit10$
  foreach $M_j \in Museums$ **do**
    foreach $M_i \in Museums$ **do**
      $Proposal \leftarrow SimilarArtists\ [M_i\ ][M_j\ ]$
      $Mrr\ [M_i\ ]$, $Hit3\ [M_i\ ]$, $Hit5\ [M_i\ ]$, $Hit10\ [M_i\ ] \leftarrow$ `EvaluateAgainstRef(`$Proposal$, $ArtistsRefList\ [M_i\ ]$`)`
    end
    $MMrr\ [M_j\ ]$, $MHit3\ [M_j\ ]$, $MHit5\ [M_j\ ]$, $MHit10\ [M_j\ ] \leftarrow$ `Mean(`Mrr, $Hit3$, $Hit5$, $Hit10$`)`
  end
  return $MMrr$, $MHit3$, $MHit5$, $MHit10$

TABLE III. GROUND-TRUTH ENTITIES IN DBPEDIA AND DDBPEDIA-FR.

	dbo:Person	dbo:Artist	No type	dbo:Museum
DBpedia	272	190	44	7
DBpedia-fr	272	327	6	12

The first row of Table III shows that not all dbo:Artist are linked to dbo:Person (ex: dbr:Sonia_Delaunay). With 12 museums and 334 artists in the reference list, 97.90% can be identified as an artist in our context graph vs. 56.88% in DBPedia, which partly explains the poor results with modelDB. As we filter the returned results by type Artist (or more generally by Person), several relevant answers are filtered.

We also compared our approach with a random selection of entities of type dbo:Artist in the vocabulary of the model. The results, given in columns $d = 4R$ of Table II, show a great benefit of leveraging the regularities in the vocabulary space to extract relationships between entities.

While analysing values on Table II, we noticed wide discrepancies between the results of different museums. For

TABLE II. MRR AND HITS@$\{3, 5, 10\}$ (%) OF A SUBSET OF REPRESENTATIVE EXAMPLES OF *Paris Musées* DATA FOR $d = \{4, 8\}$ AND $N = 1000$ WITH ANALOGY AND RANDOM FOR $d = 4$ (D=4R).

	MRR			Hits@3			Hits@5			Hits@10		
Entity	d=4R	d=4	d=8	d=4R	d=4	d=8	d=4R	d=4	d=8	d=4R	d=4	d=8
dbr:Musée_Bourdelle	0.05	0.39	0.43	0.09	0.50	0.42	0.18	0.50	0.42	0.18	0.66	0.50
dbr:Musée_Carnavalet	0.01	0.43	0.59	0.00	0.58	0.67	0.09	0.66	0.75	0.09	0.83	0.75
dbr:Musée_Zadkine	0.00	0.43	0.44	0.00	0.41	0.42	0.00	0.50	0.50	0.00	0.50	0.50
dbr:Musée_Cernuschi	0.01	0.42	0.50	0.00	0.50	0.58	0.00	0.58	0.67	0.09	0.75	0.67
dbr:Petit_Palais	0.04	0.38	0.63	0.09	0.50	0.75	0.09	0.66	0.75	0.09	0.66	0.75
dbr:Maison_de_Balzac	0.03	0.23	0.44	0.09	0.25	0.58	0.09	0.41	0.58	0.09	0.41	0.58
dbr:Musée_Cognacq-Jay	0.09	0.33	0.49	0.09	0.33	0.58	0.09	0.33	0.58	0.09	0.33	0.58
dbr:Musée_d'art_moderne	0.04	0.36	0.71	0.09	0.41	0.75	0.09	0.50	0.83	0.09	0.58	0.83
dbr:Musée_Romantique	0.03	0.34	0.48	0.09	0.41	0.50	0.09	0.41	0.58	0.09	0.50	0.58
dbr:Palais_Galliera	0.00	0.36	0.48	0.00	0.50	0.50	0.00	0.50	0.58	0.00	0.50	0.58
dbr:Maison_de_Victor_Hugo	0.01	0.38	0.55	0.00	0.50	0.58	0.00	0.58	0.58	0.18	0.58	0.67
dbr:Musée_de_Grenoble	0.00	0.34	0.33	0.00	0.41	0.33	0.00	0.50	0.33	0.00	0.50	0.33
All entities in *Paris Musées*	0.02	**0.37**	**0.52**	0.04	**0.44**	**0.58**	0.06	**0.51**	**0.62**	0.09	**0.57**	**0.64**

example, Hits@10 values for dbr:Musée_d'art_moderne and dbr:Musée_de_Grenoble are respectively: 0.83 and 0.33. This impacts the global performance of all museums (see last row of Table II). The result means for the second value that the system was not able to retrieve the corresponding artist for dbr:Musée_de_Grenoble in the top returned results. We argue this is mostly related to the representativeness of this artist's entity in the KG and how it is linked to the museum's entity; less interlinked entities (directly or indirectly through neighbors) have a lower chance of being related with the analogy structure in the embedding space.

To explain this, we run the following evaluation test:

- goal: evaluate how well we find museum which has $artist_a$ as main artist,
- input: each known pair ($museum_b$, $artist_b$) where $artist_b$ is the main artist for $museum_b$,
- method: we look for museums that play a similar role for $artist_a$ to which played by $artist_b$ for $museum_b$ and verify if the returned museums have a relationship with the $artist_a$ in the ground-truth

For example, consider the following pair $(museum_b, artist_b) = (Maison\_de\_Victor\_Hugo, Victor\_Hugo)$ and the relation $main\_artist\_for$ between them, and consider $artist_a = Honore\_de\_Balzac$. We want then to find museums related to $Honore\_de\_Balzac$ with the $main\_artist\_for$ relation (as by analogy of the relation between the pair $(Maison\_de\_Victor\_Hugo, Victor\_Hugo)$. In other words, find museums for which $Honore\_de\_Balzac$ is the main artist.

The evaluation protocol is as follows: for each $Auri_i$, URI of an artist, consider all known triples <$Muri_j$, $Auri_j$ > | $j \neq i$, and find the top most similar entities of the predicted vectors ranked by similarity. In the list of results, we filter by type $Museum$, and we then examine the intersection with museums $Muri_l$ associated with $Auri_i$.

In other words, we fix the pair (M_i, A_i) and assess all possible targets M_j in the returned results. We consider here that the pertinence for the artist A_i is the average pertinence for all possible targets. Algorithm `EVALUATION B` (Algorithm 4) shows the steps of this second evaluation.

In Algorithm 4, the `EvaluateAgainstRef` function uses the list of proposed artists and the reference list for the museum M_j. This function checks whether the artists expected by the reference list have been found in the 3^{rd} -or 5^{th} or 10^{th}- first returned results and compares the rank of found results with the reference ranking (with MRR metric). The `Mean` function averages the obtained results for the different M_j tested.

Algorithm 4: $EVALUATION\ B$

```
1  Function ArtistsFocusedEvaluation(SimilarArtists,
   ArtistsRefList, Museums)
       Input :
       Table, result of Algorithm 2 SimilarArtists
       Reference lists of artists, one list by museum
       List of museum Museums
       Output :
       List, mean of MRR values by museum MMrr
       List, mean of Hit@3 values by museum MHit3
       List, mean of Hit@5 values by museum MHit5
       List, mean of Hit@10 values by museum MHit10
2      foreach M_i ∈ Museums do
3          ← [M_i ] foreach M_j ∈ Museums do
4              Proposal ← SimilarArtists [M_i ][M_j ]
5              Mrr [M_j ], Hit3 [M_j ], Hit5 [M_j ],
                Hit10 [M_j ] ←
                EvaluateAgainstRef(Proposal,
                ArtistsRefList [M_j ])
6          end
7          MMrr [], MHit3 [], MHit5 [], MHit10 []
            ← Mean(Mrr, Hit3, Hit5, Hit10)
8      end
9      return MMrr, MHit3, MHit5, MHit10
```

Table IV shows the results of MRR and Hits@$\{3, 5, 10\}$ (%) for $d = 4$ and $N = 1000$. For example, the line dbr:Antoine_Bourdelle is built for the given pair (Musée Bourdelle, Antoine Bourdelle), then, for each other museum, we search the proposed artists. We evaluate the results with MRR, Hit@3, Hit@5, Hit@10, and put in the table the mean of the results for each museum.

The wide differences between artists' results in the last column of Table IV (Hits@10) (e.g.,dbr:Victor_Hugo and dbr:Geer_Van_Velde) reveals the impact of the triple inter-

linkage in the graph on the analogy prediction test. Thus, good prediction performance of new triples could be achieved with a good representativeness of known triples by the context graph. For evidently strong interlinkages such as for (Musée Victor Hugo, Victor Hugo), (Musée Bourdelle , Antoine Bourdelle), we have satisfactory results. For a weaker link, such as (Musée Carnavalet, Israël Silvestre), the results are unsatisfactory. We need to investigate what can be foreseen as a good interlinkage.

TABLE IV. MRR AND HITS@$\{3, 5, 10\}$ (%) OF REPRESENTATIVE EXAMPLES OF ARTISTS EXHIBITED IN MUSEUMS OF *Paris Musées* FOR $d = 4$ AND $N = 1000$

Entity	MRR	Hits@3	Hits@5	Hits@10
dbr:Antoine_Bourdelle	0.61	0.72	0.81	0.81
dbr:Israël_Silvestre	0.08	0.09	0.13	0.13
dbr:Gustave_Courbet	0.38	0.45	0.45	0.72
dbr:Ossip_Zadkine	0.67	0.81	0.90	0.91
dbr:Xu_Beihong	0.74	1.0	1.0	1.0
dbr:Honoré_de_Balzac	0.53	0.72	0.81	0.81
dbr:François_Boucher	0.65	0.72	0.81	0.81
dbr:Geer_Van_Velde	0.09	0.09	0.09	0.09
dbr:Ary_Scheffer	0.62	0.72	0.81	0.91
dbr:Jacques_Heim	0.18	0.18	0.45	0.72
dbr:Victor_Hugo	0.53	0.63	0.72	0.91

D. Performances

In this final section, we provide input on the execution performances. Building a model for this work takes 11 minutes on a laptop with 4 GB memory and a quad-core 2.8 GHz processor. We exclude the building of the context graph, which takes place only once. The following time values are only indicative and may vary slightly depending on the parameters chosen to build the model. We use the library rdflib [41] to program the entire process in Python 3. Loading the context graph takes 270 seconds; this only happens once for a set of tests with different parameters to build the model. Generating the 500 walks for each museum takes 30 seconds. Building the model with the walks takes 28 seconds. So, it is possible to generate a model on the fly during interactive sessions.

VII. CONCLUSION

In this paper we presented an approach for link discovery in knowledge bases based on neural language embedding models. We worked on contextualized RDF graphs focused on a specific domain that we extract from large general purpose knowledge graphs. Our approach leverages analogical structures extracted from relational similarities between entities generated by the neural language model. We show how these analogical regularities, applied to entities from knowledge graphs, could be used to infer new unobserved triples from the observed ones. We presented a set of algorithms, on which our approach is based, for the construction of context graphs and the preparation of test data and the evaluation process. The results of applying our approach on a domain-specific ground-truth are promising. We will continue to expand upon the research and compare it with state-of-the-art approaches for knowledge base completion on the standard baselines.

ACKNOWLEDGMENT

This work is conducted as part of the Data & Musée project selected in the $23rd$ call for projects of the Fonds Unique Interministériel (FUI) and certified by Cap Digital and Imaginove. http://datamusee.fr/le-projet/.

REFERENCES

[1] N. Mimouni, J.-C. Moissinac, and A. T. Vu, "Knowledge Base Completion With Analogical Inference on Context Graphs," in Proceedings of SEMAPRO 2019 : The Thirteenth International Conference on Advances in Semantic Processing, Porto, Portugal, Sep. 2019, pp. 44–47.

[2] M. Al-Ghossein, T. Abdessalem, and A. Barré, "Open data in the hotel industry: leveraging forthcoming events for hotel recommendation," J. of IT & Tourism, vol. 20, no. 1-4, 2018, pp. 191–216.

[3] S. Szumlanski and F. Gomez, "Automatically acquiring a semantic network of related concepts," in Proceedings of the 19th ACM CIKM, 2010, pp. 19–28.

[4] J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li, and X. Li, "Neural generative question answering," in Proceedings of IJCAI. AAAI Press, 2016, pp. 2972–2978.

[5] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek, "Distant supervision for relation extraction with an incomplete knowledge base," 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, Jun. 2013, pp. 777–782.

[6] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, ser. AAAI'16. AAAI Press, 2016, pp. 1955—-1961.

[7] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in NIPS, 2013.

[8] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in Proceedings of ICML, 2017, pp. 2168–2178.

[9] A. García-Durán, A. Bordes, N. Usunier, and Y. Grandvalet, "Combining two and three-way embedding models for link prediction in knowledge bases," J. Artif. Intell. Res., vol. 55, 2016, pp. 715–742.

[10] D. Q. Nguyen, "An overview of embedding models of entities and relationships for knowledge base completion," CoRR, vol. abs/1703.08098, 2017.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in NIPS, 2013.

[12] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A large ontology from wikipedia and wordnet," Journal of Web Semantics, vol. 6, no. 3, 2008, pp. 203–217, world Wide Web Conference 2007Semantic Web Track.

[13] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia," Semantic Web, vol. 6, no. 2, 2015, pp. 167–195.

[14] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 1247—1250.

[15] C. Fellbaum, Ed., WordNet: An Electronic Lexical Database, ser. Language, Speech, and Communication. Cambridge, MA: MIT Press, 1998.

[16] R. Navigli and P. Velardi, "Structural semantic interconnections: a knowledge-based approach to word sense disambiguation," IEEE transactions on pattern analysis and machine intelligence, vol. 27, no. 7, 2005, pp. 1075–1086.

[17] A. Fader, L. Zettlemoyer, and O. Etzioni, "Open question answering over curated and extracted knowledge bases," in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '14. Association for Computing Machinery, 2014, pp. 1156—-1165.

[18] L. Getoor and B. Taskar, Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press, 2007.

[19] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in Twenty-Eighth AAAI conference on artificial intelligence, 2014.

[20] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in Twenty-ninth AAAI conference on artificial intelligence, 2015.

[21] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," arXiv preprint arXiv:1412.6575, 2014.

[22] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction." International Conference on Machine Learning (ICML), 2016.

[23] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data." in International Conference on Machine Learning (ICML), vol. 11, 2011, pp. 809–816.

[24] 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, 2013, pp. 926–934.

[25] D. Chen, R. Socher, C. D. Manning, and A. Y. Ng, "Learning new facts from knowledge bases with neural tensor networks and semantic word vectors," arXiv preprint arXiv:1301.3618, 2013.

[26] H. Liu and Y. Yang, "Cross-graph learning of multi-relational associations," arXiv preprint arXiv:1605.01832, 2016.

[27] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in ACM SIGKDD, 2016.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2013.

[29] P. Ristoski and H. Paulheim, "Rdf2vec: Rdf graph embeddings for data mining," in Proceedings of ISWC, 2016, pp. 498 —- 514.

[30] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," J. Mach. Learn. Res., vol. 3, no. null, Mar. 2003, pp. 1137—1155.

[31] Y. Bengio and Y. Lecun, Scaling learning algorithms towards AI. MIT Press, 2007.

[32] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model." in INTERSPEECH, T. Kobayashi, K. Hirose, and S. Nakamura, Eds. ISCA, 2010, pp. 1045–1048.

[33] X. Rong, "word2vec parameter learning explained," CoRR, vol. abs/1411.2738, 2014. [Online]. Available: http://arxiv.org/abs/1411.2738

[34] D. Meyer, "How exactly does word2vec work?" http://www.1-4-5.net/~dmm/ml/how_does_word2vec_work.pdf, accessed: 2020.05.30.

[35] T. Soru, S. Ruberto, D. Moussallem, A. Valdestilhas, A. Bigerl, E. Marx, and D. Esteves, "Expeditious generation of knowledge graph embeddings," 2018.

[36] "Rdfs schema," https://www.w3.org/TR/rdf-schema/, accessed: 2020.05.30.

[37] I. Hulpus, C. Hayes, M. Karnstedt, and D. Greene, "Unsupervised graph-based topic labelling using dbpedia," in Proceedings of WSDM, 2013, pp. 465–474.

[38] W. Shen, J. Wang, and J. Han, "Entity linking with a knowledge base: Issues, techniques, and solutions," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 2, Feb 2015, pp. 443–460.

[39] "Gensim implementation of word2vec," https://radimrehurek.com/gensim/models/word2vec.html, accessed: 2020.05.30.

[40] "Paris musées collections website." http://parismuseescollections.paris.fr/fr/recherche, accessed: 2020.05.30.

[41] "Rdflib python library," https://github.com/RDFLib/rdflib, accessed: 2020.05.30.