Finding Interpretable Concept Spaces in Node Embeddings using Knowledge Bases

Maximilian Idahl, Megha Khosla, and Avishek Anand

L3S Research Center, Hannover
{idahl,khosla,anand}@l3s.de

Abstract. In this paper we propose and study the novel problem of explaining node embeddings by finding embedded human interpretable subspaces in already trained unsupervised node representation embeddings. We use an external knowledge base that is organized as a taxonomy of human-understandable concepts over entities as a guide to identify subspaces in node embeddings learned from an entity graph derived from Wikipedia. We propose a method that given a concept finds a linear transformation to a subspace where the structure of the concept is retained. Our initial experiments show that we obtain low error in finding fine-grained concepts.

Keywords: Interpretability · Node Embeddings · Conceptual Spaces.

1 Introduction

Representations of nodes in a graph or node embeddings have proven useful in many applications such as question answering [1], dialog systems [14], recommender [21] systems and knowledge-base completion [15]. The core idea behind *node representation learning* (NRL) [22, 4, 10] approaches is to distill the high-dimensional discrete representation of nodes into a dense vector embedding using dimensionality reduction methods, which optionally not only incorporate the graph structure, but also features attached to nodes. These representations can be seen as features extracted from only the topology or from both the topology and the available node attributes. The dense representations thereby learnt form a latent feature space where the basis or dimensions are non-interpretable.

Consequently, in spite of their success, there is a lack of an understanding of what the latent dimensions encode in terms of existing human knowledge. This is problematic for downstream tasks requiring interpretability, since using such embeddings results in the input already being non-interpretable. For aiding interpretability and utility of these embeddings in downstream application scenarios we initiate an inquiry into presence of *interpretable* or human understandable subspaces in the learnt feature representation space of these graph embeddings. We ask the fundamental question: *What do node embeddings encode in terms of human world knowledge?* Recent works in interpretability for learning on structured data either focus on generating interpretable embeddings or explaining the predictions made by a classifier to which embeddings form the

input [26]. But none of these methods provide insights into the embedding itself, a problem which we propose and study in this work.

We take an alternate view on interpretability of node embeddings in that we want to find sub-spaces in the embedding space corresponding to human-understandable concepts. Our main contribution is in finding interpretable sub-spaces in the latent feature representation space and thus characterizing the behavior of node representations when projected into these interpretable spaces. This has two distinct advantages – first we do not compromise on the effectiveness of these embeddings as we post-hoc analyze the presence of interpretable spaces in the already learned representation space. Secondly, we ground the interpretable space to existing world knowledge in the form of knowledge bases.

To this extent, in this work, we use external knowledge bases (KB) to learn conceptual spaces for corresponding characteristics that can be attributed to a given node. In particular, we assume that we have an input graph of labelled or named nodes. As a use case we focus on a hyperlink graph of named entities. We observe that KBs like YAGO [7] encode human understandable concepts organized in a taxonomy which can be used as the source of world knowledge assuming that the nodes/entities in the input graph are also present in the taxonomy. In principle one can use any input graph and KB as long as the input graph node names are grounded in the KB. Having extracted the possible concepts from the taxonomy, we then propose methods to explain a node embedding in terms of the applicability of various concepts. For example, a node named *Albert Einstein* could be explained by concepts like *Theoretical physicists*, *Scientists* etc.

We propose two simple algorithms, SAS and CSD, to explain node embeddings in terms of concepts and provide promising first results for pre-trained embeddings corresponding to two unsupervised random walk based node embedding methods, namely, DeepWalk [22] and LINE [24]. We show that our second approach CSD that projects a node embedding to a common learnt concept space distinguishes the applicable and non applicable concepts better than our first approach which operates in the original embedding space.

2   Related Work

Supervised learning approaches are either *interpretable by design* [3, 13, 25] or explanations can be generated in a post-hoc manner after the model is trained [23, 12, 19]. Post-hoc methods for interpretability either operate introspectively (full access to the model parameters) [12, 19] or are model agnostic [23]. We operate in the model introspective interpretable regime where we assume full access to the model parameters. For other notions of interpretability and a more comprehensive description of the approaches we point the readers to [5].

Methods focussing on building interpretable representations include MEmbER [8] which learns entity embeddings using max-margin constraints to encode the desideratum that (salient) properties of entities should have a simple geometric representation in the entity embedding. Jameel and Schockaert [9] propose

a method which learns a vector-space embedding of entities from Wikipedia and constrains this embedding such that entities of the same semantic type are located in some lower-dimensional subspace. Minervini et al. [18] leverage equivalence and inversion axioms during the learning of knowledge graph embeddings, by imposing a set of model dependent soft constraints on the predicate embeddings. Post-hoc methods include GNN-Explainer [26] which provides interpretations for GNN predictions on link prediction, node classification and graph classification tasks. The interpretations are tied to specific tasks. We, on the other hand, propose to understand the node representations directly in terms of user provided conceptual categories.

Unlike the above works we focus on explaining the node vector representation itself which might have been obtained using an arbitrary embedding method.

3 Preliminaries

In this section we give a brief overview of YAGO and node embedding methods used in this work.

3.1 Knowledge Graphs

As a source of *concepts* or human understandable world knowledge we use the YAGO [7] knowledge base (KB), which was automatically constructed from Wikipedia. Typically, each article in Wikipedia becomes an entity in the knowledge base (e.g., since Albert Einstein has an article in Wikipedia, Albert Einstein is an entity in YAGO). Each entity is organized into a taxonomy of classes. In addition, every entity is an instance of one or multiple classes and every class (except the root class) is a subclass of one or multiple classes. therefore yielding a hierarchy of classes the *YAGO taxonomy*.

Each class name is of the form `<wordnet_XXX_YYY>` or `<wikicat_XXX_YYY>` , where XXX is the name of the concept (e.g., singer), and YYY is the WordNet 3.0 synset id of the concept (e.g., 110599806). For example, the class of singers is `<wordnet_singer_110599806>`. Additionally, each class is connected to its more general class by the `rdfs:subclassOf` relationship.

Not all Wikipedia categories correspond to classes in YAGO. The lowest layer of the taxonomy is the layer of instances. Instances comprise individual entities such as rivers, people, or movies. For example, the lowest layer contains `<Elvis_Presley>`. Each instance is connected to one or multiple classes of the higher layers by the relationship rdf:type. For example, for entity Albert_Einstein we have:

```
<Albert_Einstein>  rdf:type  <wikicat_Nuclear_physicist>.
```

One can therefore walk from the instance up to its class by `rdf:type`, and then further up by `rdfs:subclassOf`. In Section 4 we will provide details about how the concepts derived from the taxonomy are used as explanations for node embeddings.

3.2   Node Embeddings

Node representations or node embeddings can be understood as the set of features extracted from the graph topology and (if given) node attributes. The present set of techniques for node representation learning generally fall into one of these categories : (1) random walk based [22, 4, 24, 10], (2) matrix factorization based [2, 20] or (3) deep learning or Graph Neural Network (GNN) based [11, 6]. In this section we describe briefly the two random walk based approaches which we employ in this work. In future we will investigate our methods using a general set of unsupervised and semi-supervised embedding approaches.

The basic idea behind random walk based embedding techniques is to transform the graph into a collection of node sequences, in which, the occurrence frequency of a node-context pair measures the structural distance between them. *DeepWalk* [22] was the first method to exploit random walk techniques to build sentence like structures from graphs to train a *SkipGram* model [17]. It employs truncated random walks to create vertex sequences, which are later used in a word2vec fashion to learn vertex embeddings given its context. For a graph G, it samples uniformly a random vertex v as the root of the random walk W_v. A walk samples uniformly from the neighbors of the last vertex visited until the maximum length t is reached. For each $v_i \in W_v$ and for each $u_k \in W[j-c : j+c]$ (c is the window size), (v_j, u_k) forms a vertex-context training pair (similar to word -context pair in word embeddings). The objective is then to maximize the probability of observing u_k given the representation of v_j. LINE [24] optimizes first order proximity (i.e. embeds nodes sharing a link closer) and second order proximities (embeds nodes closer if they have similar neighborhoods) using an SGNS (Skip-gram with negative sampling) objective function [16]. Similar to DeepWalk, it can be understood as sampling random walks of length 1 and uses vertices sharing an edge as training pairs.

4   Research Questions and our Approach

We propose a general approach for post-hoc interpretability of node representation learned by an unsupervised or semi-supervised method. We bring in a completely new perspective of interpretability of extracted features of nodes by using external knowledge to determine the concepts that a given representation encodes. More precisely, we use Wikipedia entity graph, $G = (V, E)$, as the input graph, where the nodes are Wikipedia pages and the edges correspond to the hyperlinks between them. We employ DeepWalk and LINE to generate embeddings for all $v \in V$. We ignore the edge direction to learn the embeddings. We also recall that the present topic of this work is to define and validate interpretability on node embeddings and the choice of embeddings methods is therefore arbitrary. Let Φ_v represent the embedding vector corresponding to v. We ponder over the following question:

RQ 1 *What concepts do these embeddings encode?*

As the embeddings are usually generated only considering the structure of the graph or/and node attributes, an embedding vector Φ_v encodes the concepts which it shares with its neighborhood (neighborhood here depends on the employed embedding method). Consider, for example, an entity *Barack Obama*, which could be understood as sharing characteristics with other *Presidents* and *Nobel Prize winners*. *Presidents* and *Nobel Prize winners* here are the human understandable world knowledge or concepts. Rather than characterizing nodes in terms of their neighbors, we in this work use these implicit human understandable concepts to characterize an embedding vector. In particular, for a given embedding vector Φ_v and a concept c, we assign a score $\mathcal{S}(\Phi_v, c) \in \mathcal{R}$ which quantifies the characteristic c of the embedding Φ_v. Roughly speaking, the score measures the amount of the characteristic that an embedding vector possesses.

The challenge here is that often only the graph structure or sometimes the node attributes are also available but there are no explicit concepts provided. We therefore ask the following question:

RQ 2 *How can explicit concepts be constructed given an input graph with named vertices?*

In order to generate possible concepts related to an entity, we propose the use of external knowledge base like YAGO (see also Section 3.1), which provides a hierarchy of concepts related to any given node, say v in the graph. These concepts form the characteristics of v. The user can then query the encoding of possible concepts in the trained node embedding. For example, a user may ask how much the embedding vector corresponding to *Barack Obama* encodes *American Presidents* and *Scientists*. One might assume that the Obama's embedding vector should not have anything to do with the concept *Scientists*, which might not be true as the underlying graph might put Obama in close proximity with other Nobel Prize winners who are also Scientists. Having defined or collected concepts from external knowledge bases, the next natural question is:

RQ 3 *For a given embedding vector, Φ_v and a concept c, how can we score the applicability of c to Φ_v ?*

To quantify the applicability of concept corresponding to an embedding or to explain an embedding in terms of the applicable and not applicable concepts, we propose two algorithms: Simple Aggregation Strategy (SAS) and Concept Space Discovery (CSD).

4.1   Simple Aggregation Strategy

The first approach uses a simple aggregation strategy to build concept representations from the representations of the nodes (from the training set) to which the concept is applicable (test nodes are held out). In particular, we first compute a vector representing the given concept by taking the element-wise mean of all the embedding vectors corresponding to nodes to which the concept applies, excluding the query nodes. This vector defines the *concept center*. To score a

query node, we compute the L2 distance between its embedding vector and the concept center.
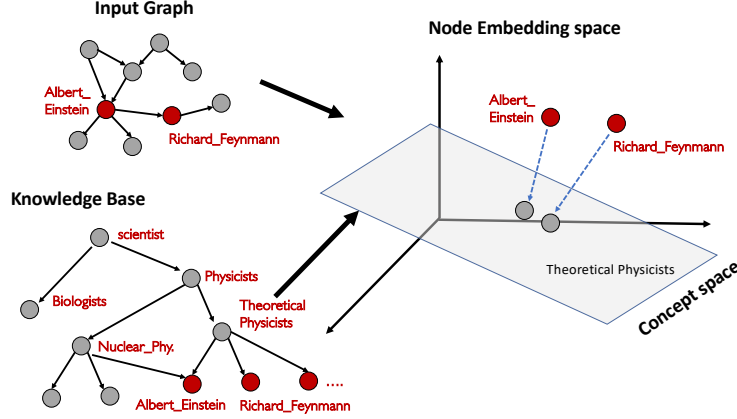

Fig. 1: Extracting Concept Spaces

4.2   Concept Space Discovery (CSD)

The second algorithm is more involved and explicit, in the sense that for each concept c it learns a linear transformation, which is used to project the node vectors into a more restricted space for c, that we call *concept space*. The original embedding vectors are projected into this new space to extract their effective representations which best encode the given concept (refer Figure 1). We learn the parameters for this transformation on triplets of entities, using triplet loss. Let a be the entity node (also called anchor node) which is a direct descendant of concept c, p be some sibling of a in the taxonomy and n be the negative example , i.e., an entity which is not a sibling of a in the taxonomy. For any node v, let Φ_v represent the corresponding embedding vector. The triplet loss $\mathcal{L}(a, p, n)$ is then defined as follows.

$$\mathcal{L}(a, p, n) = max\{d(\Phi_a, \Phi_p) - d(\Phi_a, \Phi_n) + m, 0\} \qquad (1)$$

where $d(\Phi_x, \Phi_y) = ||x - y||_2$ and m is a margin specific to the negative entity in a triplet. We set this margin to be the distance from the target concept to the lowest common ancestor concept shared by the positive and the negative entity, i.e. negative entities that are conceptually close to the positive entity have lower margins and ones that are conceptually far away have higher margins. We refer to negative entities with low margins as soft negatives and to negative entities with high margins as hard negatives. An illustrative example for computing margin is provided in Figure 2.

Score Computation. The scoring of how much a concept applies to a query entity is analogous to the first approach, but of course operates in the *concept*
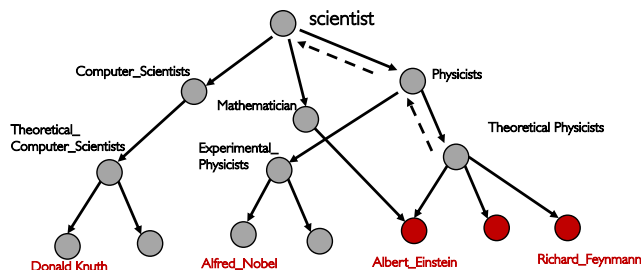
Fig. 2: Margin for Triplet loss is determined by the similarity in the taxonomy graph. The margin between Albert Einstein and Donald Knuth is 2, where as the margin between Albert Einstein and Alfred Nobel is 1.

space. That is, for a given concept c and the positive entities (the training set) corresponding to the concept, we first compute their projections into common concept space and then compute the mean of the resulting projected vectors to represent the concept. Again for a given query node, we first compute its projection into the concept space and the final score is then given by the L2 distance between the concept vector and the query projection. Lower the score, better is the concept encoded by the query node. Note that both loss function and scoring make use of the same distance metric, the L2 distance.

5   Experiments

5.1   Data Acquistion

We conduct our experiments on the Wikipedia entity graph, where the nodes are Wikipedia pages and the edges correspond to the hyperlinks between them. In addition, we use the type hierarchy of YAGO as the KB and consider all leaves under a concept node as belonging to the concept, as described in Section 3.1.
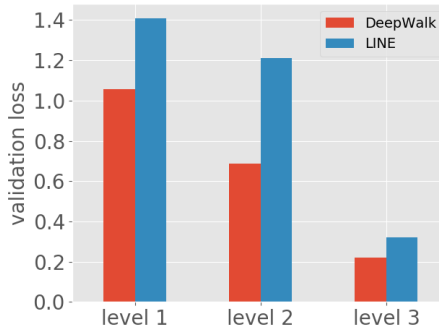
5.2   Methodology

Given a query entity q and a start concept c_{start} we learn concept spaces for c_{start} and its sibling concepts in the taxonomy. Note that we limit the number of concepts due to computation (Some concepts have a large number of siblings). For each selected concept, we a learn a concept representation as described in Section 4. Below we give more details about the training employed in our second approach CSD.

For CSD where we use triplet loss function to learn the concept space we choose positive and negative examples as follows. For each concept c, the set of positive entities (examples) consists of all entities contained in c. Next, we rank all ancestor concepts of c by the margin, which is the distance of the concept to c. Following Figure 2, if c is *Theoretical Physicists*, then entities which belong to the concept *Physicists* are negative entities with a margin of 1, entities belonging

to the concept *Scientists* are negative entities with a margin of 2, and so on. Note that an entity is always assigned the lowest possible margin. In this example, all physicists get assigned a margin of 1 and only all scientists that are not physicists get assigned a margin of 2. We also exclude the query entity q from the sampling process. We split the sets of positive and negative entities into a training and a validation set, taking 20% of the entities for the validation set.

In order to generate a triplet, we select a positive entity uniformly from the set of positive samples. An anchor entity is selected in the same way, with respect to the anchor not being the same entity as the chosen positive one. Next, we select a margin m uniformly from the available margins in the set of negative entities. Then, we select a negative entity uniformly from the negative samples corresponding to margin m. To train one concept space, we sample a total of ten thousand triplets. We then train the linear transformation using Stochastic Gradient Descent with Momentum for 100 epochs, with a mini batch size of 16 and a leaning rate of 0.001. We stop the training early if the validation loss does not improve over 5 epochs. After training, we score the query entity as described in Section 4 corresponding to our two approaches.

Fig. 3: Mean validation losses for training concept space projections for concepts of different hierarchy levels. Level 1 includes concepts high up in the hierarchy, namely *person*, *organization* and *country*. The second level includes *scientist*, *educational institution* and *countries in Europe*. Level 3 then covers the more fine-grained concepts *theoretical physicist*, *university or college in Germany* and *states of Germany*.

6   Results

In Figure 3 we show the errors corresponding to each concept level for different node embedding approaches (DeepWalk and LINE). Concepts at a higher level, as expected, exhibit higher error but the error reduces to a small value for more specific concepts. It is interesting to observe that it is easier to find interpretable concept spaces in DeepWalk as opposed to LINE. In this regard DeepWalk can be in some sense regarded as more interpretable than LINE.

Figure 4 and Figure 5 show the scores of different concepts for the query entities *Albert Einstein* and *Donald Trump*, respectively. We recall that lower the score $\mathcal{S}(\Phi_v, c)$, more is the applicability of c towards the embedding vector Φ_v or the entity v. Concepts under which the query entity is listed in YAGO are shown in green, concepts under which it is not listed in red.

(a) SAS DeepWalk

(b) SAS LINE
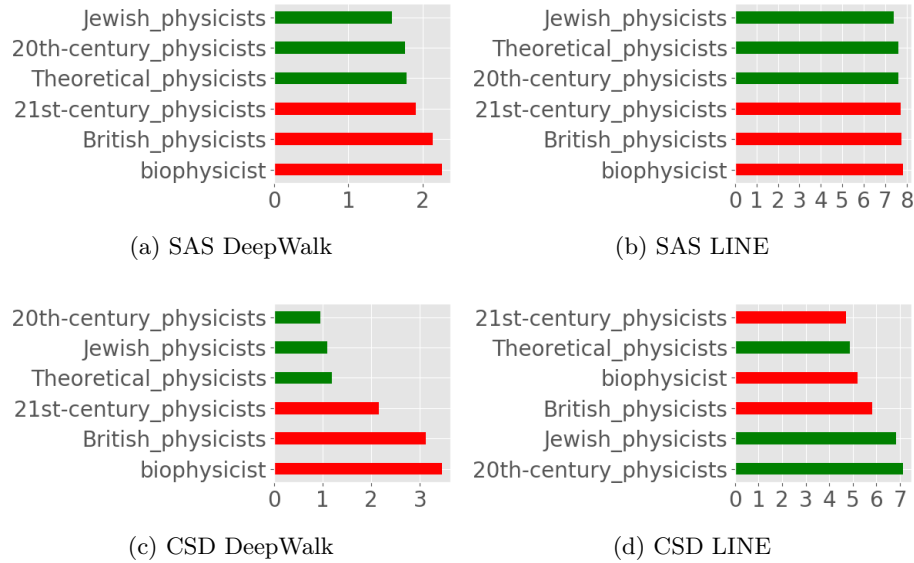
(c) CSD DeepWalk

(d) CSD LINE

Fig. 4: Concept Ranking for Albert Einstein

We note that for the query entity *Albert Einstein*, scoring concepts in both of the original embedding spaces (Figure 4 a,b) yields a correct ranking of the concepts. Yet, there is not much difference between the scores of concepts which apply to the query entity and the scores of non-applicable concepts. This is more prominent the case for the embeddings generated by LINE, where differences in the scores are barely noticeable.

We observe a similar behaviour with our second query entity *Donald Trump*. An interesting observation here is that the best ranked concept in Figure 5 b, *Leaders of organizations* which is not listed as applicable concept in the taxonomy, in fact applies to the query entity *Donald Trump*. This is another finding, in the sense that the embeddings encode knowledge not present in YAGO. Using concept spaces to score the query entity increases the differences between scores. This seems to work well for both query entities when using the embeddings generated by DeepWalk. The concept spaces deliver scores where it is much clearer whether a concept applies to the query entity or not, as there is a large gap between applicable ones and non-applicable ones.

7   Conclusions and Future Work

In this work we proposed a method to find interpretable concept spaces for graph embeddings. We hypothesize that latent feature spaces that embed named vertices are not interpretable themselves but contain subspaces that do contain human understandable concepts. We propose an algorithm that tries to find

(a) SAS DeepWalk

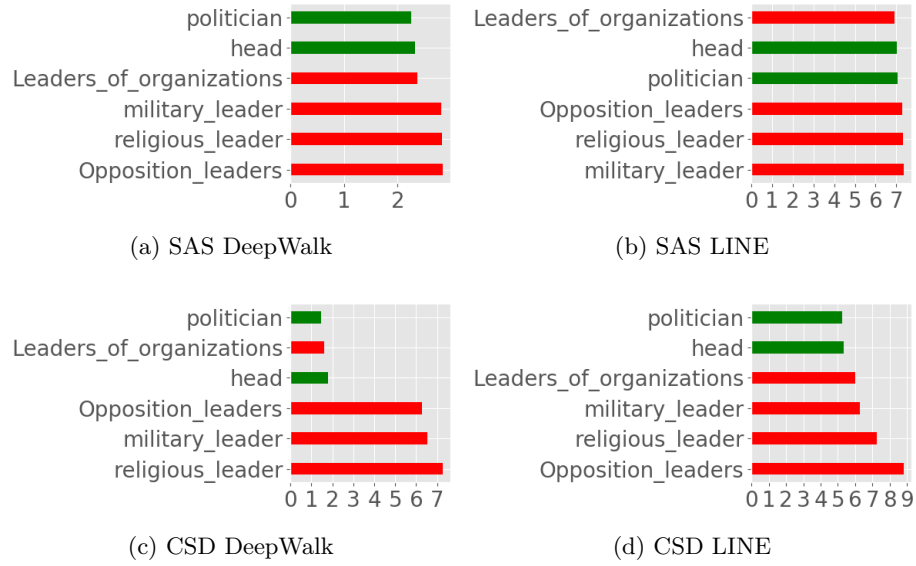(b) SAS LINE

(c) CSD DeepWalk

(d) CSD LINE

Fig. 5: Concept Ranking for Donald Trump

subspaces in the feature representation space by exploiting similarity of entities in the KB using triplet loss. We anecdotally show the effectiveness of our approach on a small subset of concepts chosen from the KB.

As future work there are plenty of avenues to investigate in detail. First, we would want to improve our evaluation procedure to quantitatively establish the effectiveness of our concept space discovery approach. This would require us to not only experiment with a large set of concepts but increase our coverage to multiple unsupervised and semi-supervised node representation learning methods. Secondly, we would want to find out that if there are non-linear sub spaces that encode coarse-granularity concepts like scientists, politicians etc. Currently, we see room for improvement in finding subspaces for coarser granularity topics due to choice of linear subspaces.

Bibliography

[1] Bordes, A., Chopra, S., Weston, J.: Question answering with subgraph embeddings. arXiv preprint arXiv:1406.3676 (2014)

[2] Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: CIKM. pp. 891–900. ACM (2015)

[3] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., Elhadad, N.: Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1721–1730. ACM (2015)

[4] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD. pp. 855–864. ACM (2016)

[5] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 93 (2018)

[6] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS. pp. 1024–1034 (2017)

[7] Hoffart, J., Suchanek, F.M., Berberich, K., Lewis-Kelham, E., De Melo, G., Weikum, G.: Yago2: exploring and querying world knowledge in time, space, context, and many languages. In: Proceedings of the 20th international conference companion on World wide web. pp. 229–232. ACM (2011)

[8] Jameel, S., Bouraoui, Z., Schockaert, S.: Member: Max-margin based embeddings for entity retrieval. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 783–792. ACM (2017)

[9] Jameel, S., Schockaert, S.: Entity embeddings with conceptual subspaces as a basis for plausible reasoning. In: Proceedings of the Twenty-second European Conference on Artificial Intelligence. pp. 1353–1361. IOS Press (2016)

[10] Khosla, M., Leonhardt, J., Nejdl, W., Anand, A.: Node representation learning for directed graphs. In: ECML-PKDD (2019)

[11] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)

[12] Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. arXiv preprint arXiv:1703.04730 (2017)

[13] Letham, B., Rudin, C., McCormick, T.H., Madigan, D., et al.: Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics **9**(3), 1350–1371 (2015)

[14] Ma, Y., Crook, P.A., Sarikaya, R., Fosler-Lussier, E.: Knowledge graph inference for spoken dialog systems. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5346–5350. IEEE (2015)

[15] Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., Stucken-schmidt, H.: Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In: ISWC. pp. 3–20. Springer (2018)

[16] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013), `http://arxiv.org/abs/1301.3781`

[17] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS. pp. 3111–3119 (2013)

[18] Minervini, P., Costabello, L., Muñoz, E., Nováček, V., Vandenbussche, P.Y.: Regularizing knowledge graph embeddings via equivalence and inversion axioms. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 668–683. Springer (2017)

[19] Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition **65**, 211–222 (2017)

[20] Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: SIGKDD. pp. 1105–1114. ACM (2016)

[21] Palumbo, E., Rizzo, G., Troncy, R., Baralis, E., Osella, M., Ferro, E.: An empirical comparison of knowledge graph embeddings for item recommendation. In: DL4KGS@ ESWC. pp. 14–20 (2018)

[22] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD. pp. 701–710. ACM (2014)

[23] Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144. ACM (2016)

[24] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW. pp. 1067–1077 (2015)

[25] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: International Conference on Machine Learning. pp. 2048–2057 (2015)

[26] Ying, R., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnn explainer: A tool for post-hoc explanation of graph neural networks. arXiv preprint arXiv:1903.03894 (2019)