Interpreting Node Embedding with Text-labeled Graphs

Giuseppe Serra[†§], Zhao Xu[†], Mathias Niepert[†], Carolin Lawrence[†], Peter Tiňo[§] and Xin Yao[§‡]
[†]NEC Laboratories Europe, 69115 Heidelberg, Germany
[§]CERCIA, School of Computer Science, University of Birmingham, UK
[‡]Southern University of Science and Technology, Shenzhen, China
{giuseppe.serra, zhao.xu, mathias.niepert, carolin.lawrence}@neclab.eu, p.tino@cs.bham.ac.uk, xiny@sustech.edu.cn

***Abstract*—Graph neural networks have recently received increasing attention. These methods often map nodes into latent spaces and learn vector representations of the nodes for a variety of downstream tasks. To gain trust and to promote collaboration between AIs and humans, it would be better if those representations were interpretable for humans. However, most explainable AIs focus on a supervised learning setting and aim to answer the following question: *"Why does the model predict y for an input x?"*. For an unsupervised learning setting as node embedding, interpretation can be more complicated since the embedding vectors are usually not understandable for humans. On the other hand, nodes and edges in a graph are often associated with texts in many real-world applications. A question naturally arises: could we integrate the human-understandable textural data into graph learning to facilitate interpretable node embedding? In this paper we present interpretable graph neural networks (iGNN), a model to learn textual explanations for node representations modeling the extra information contained in the associated textual data. To validate the performance of the proposed method, we investigate the learned interpretability of the embedding vectors and use functional interpretability to measure it. Experimental results on multiple text-labeled graphs show the effectiveness of the iGNN model on learning textual explanations of node embedding while performing well in downstream tasks.**

***Index Terms*—Node embedding, interpretability, text mining**

I. Introduction

Learning graph data with neural networks [1]–[4] has recently attracted considerable interest. Graph neural networks (GNNs) have been applied to a variety of applications with great success, e.g., knowledge base completion [5], [6], disease classification [7] and protein interaction prediction [8], machine translation [9], [10] and relation extraction [11], [12], visual question answering [13], [14] and object detection [15]. In this framework, node embedding plays an important role. Many GNN approaches formulate graph learning with node embedding, and the downstream task, such as link prediction, node classification and (sub-)graph classification, is modeled with node embedding vectors. However, the resulting embedding vectors are usually not understandable: a 100d or 200d vector makes little sense from the human perspective. Thus, for humans to gain trust in AI, along with high performance on graph analysis tasks, interpretability of node representation is expected. Node embedding interpretability potentially includes two folds: 1. what the learned vectors mean (i.e. outcome explanation), 2. why a machine learning method generates a specific node embedding vector given a graph (i.e. model explanation). In this paper, we target to explore the first type of interpretability: the meaning of the embedding itself. Our goal is to use human-understandable information (i.e. texts) to interpret the embedding vectors. The textual explanation allows humans to grasp the gist of the embedding vectors, the (dis)similarity between two nodes, as well as the results of the follow-up learning tasks, e.g. recommendation. For example, we could explain the embedding vector of a user, from a user-restaurant network, as {*circumstances-sensitive, visiting with family*}.

Our goal in this paper is to present an approach to learn textual explanations of node embedding vectors. Starting from a text-labeled graph, we integrate the textual information into the model (e.g. medical narratives in a doctor-patient graph, reviews in a user-product graph) to learn interpretable node embeddings. The textual explanation of a node i is formulated as a node-specific word distribution conditioned on its embedding vector $\mathbf{x}_i$. Since the creation of the word-vectors is directly linked with the node embeddings (which are supposed to work well in downstream tasks), we use an objective function that combines the accuracy of a downstream task (e.g. rating prediction, sentiment analysis) with the likelihood of the textual corpus. We introduce an additional node clustering to model the patterns among the embedding vectors, the corresponding textual explanations and the texts associated. The additional cluster embedding allows our model to learn the discrete structures of the graph data. Since discrete cluster samples from categorical distributions are non-differentiable, sparsegen-lin is employed to draw *soft*-categorical samples from the nodes. We test the performance of the proposed method with multiple text-labeled graph data sets. The experiment results demonstrate the effectiveness of our model on learning textual explanations starting from node embeddings.

The rest of the paper is organized as follows. We start off with a brief review of related works. Afterwards we describe the proposed model to learn human understandable explanation of node embedding. Before concluding, we present our experimental results on multiple data sets. In the experimental section, we both perform quantitative and qualitative evaluations of the resulting output.

II. Related Works

There are two lines of research related to the work.

Graph Neural Networks Among the GNN approaches for network embedding, the pioneer Deepwalk [4] is undoubtedly one of the most popular. It samples a set of random walks from a graph as sentences, and learns node embedding with the SkipGram method. Deepwalk is then extended by [16] to address large-scale networks, and by [2] to model flexible network neighborhood of nodes. Another line of GNN research is TransE [1] and its variants [6], [17]. They generally model edges of a graph as functions of embedding vectors. Recently, graph convolutional neural network and its extensions [3], [18], [19], become popular due to high flexibility and good generalization. The family of methods directly learn a single embedding function, instead of a set of embedding vectors. However, none of these methods takes into account the textual information. To include textual data into the embedding, to name a few, [20] learns embedding of both textual and network structure, and concatenates them to obtain node embedding. [21] uses a word alignment mechanism to absorb impacts from proximate texts more effectively. In both cases, the textual information is just used to improve the resulting node representation. However, they do not attempt to directly generate explanations of the node embeddings. Recent advances in GNNs can be found on some excellent surveys, e.g. [22]–[25].

Explainable AI The adoption of complicated machine learning methods (e.g. deep neural networks) in real-world applications have recently led to an increasing interest in interpretable AI [26]. The extensive use of machine learning methods in industrial, medical and socio-economical applications requires a better understanding of these models. The improper use of machine learning could lead to high-impact risks since humans could not fully understand the underlying system or the resulting output. Recent advances in interpretable AI propose many post-hoc interpretability approaches and, depending on the setting and the application domain, they can be categorized as follows:

- *Global/local explanations*: LIME [27] and SHAP [28] are the most popular approaches in this category. They are surrogate models able to explain the outputs of any classifier in an interpretable way. For example, in a medical environment, they can visualize what are the symptoms in the patient's history that led to the prediction *'The patient has the flu'*.
- *Logic-based approaches*: the logic-based methods create falling rules and interpretable decision sets. In [29], inspired by healthcare applications, the authors propose a Bayesian framework for learning falling rule lists that consist of an ordered list of if-then rules to determine which sample should be classified by each rule. [30] propose to learn decision sets (i.e. independent sets of if-then rules) through an objective function that simultaneously optimizes accuracy and interpretability of the rule.
- *Attribution methods*: in this broader category, the methods try to explain the output by highlighting characteristics of the output itself (e.g. textual explanations with highlighted relevant words) or by highlighting characteristics of the input that strongly influence the result. In the latter case, the most popular approaches attempt to analyze the gradient to investigate how the input features condition the output. [31], [32] generate *saliency maps* in convolutional neural networks (CNNs). [33] estimates the influence of training examples in neural matrix factorization models.

For a comprehensive discussion on methods for explainable AI, we refer the reader to recent popular surveys [34], [35].

In this paper, we focus our attention on explaining outputs in a different unsupervised learning case, namely node embedding. There are few previous works attempting to improve the interpretability of node embeddings. The existing works mainly aim to explain the embedding dimensions as clusters in an implicit manner, e.g. employing Canonical Polyadic decomposition [36], and assigning a meaning to each vector dimension [37]. Unlike these approaches, our method focuses on improving the interpretability of node embeddings explicitly to get human understandable explanation by exploiting the extra textual information associated with the graphs. Our method maps the latent space of node embeddings into textual space through word-based vectors. Thus, the additional available textual information works as a human-understandable source to generate explanations of node embeddings.

III. Towards Interpretation of Node Embedding

In this section, we introduce the proposed method iGNN to learn interpretable node representations. In summary, our model attempts to combine two objectives: a) learn node embeddings that perform well in downstream tasks b) generate textual explanations of the learned vector representations. To illustrate the method, we use a typical review network as a running example. Assume there is a bipartite graph $\mathcal{G}$ with N number of users and M number of products. Between a user i and a product j, there is an edge $e_{i,j}$. Each edge is associated with a set of words (namely, a review) $s_{i,j} = \{w_{i,j,1}, \ldots, w_{i,j,S}\}$ and a rating $r_{i,j}$. The size of the vocabulary is V. The number of reviews is R.

A. The Generative Process

Technically, we embed the iGNN in a neural generative modeling framework, which integrates good properties of probabilistic generative models and neural networks. In particular, we sample the edges and the associated texts as follows:

- For each user i, there is an embedding vector $\mathbf{x}_i \in \mathcal{R}^D$ associated, which is sampled from a multivariate Gaussian with zero mean and a diagonal covariance matrix $\mathbf{I}$:
$$\mathbf{x}_i \sim \mathcal{N}_D(\mathbf{0}, \mathbf{I})$$
- For all users, we introduce K clusters, and each user cluster k is associated with an embedding vector $\mathbf{c}_k \in \mathcal{R}^D$, which is again drawn from a Gaussian:
$$\mathbf{c}_k \sim \mathcal{N}_D(\mathbf{0}, \mathbf{I})$$

Here we assume all embedding vectors have the same dimension to avoid complicated notation.

- The user cluster weights θ_i are computed based on the embedding vectors:

$$\theta_{i,k} = \text{sparsegen-lin}\left(f(\mathbf{x}_i, \mathbf{c}_k; \phi); \lambda_u\right)$$

which specifies the probability of the user i in the cluster k. The function f quantifies the relevance or similarity between the user $\mathbf{x}_i$ and the cluster $\mathbf{c}_k$. We define the function using a neural network with parameters ϕ, e.g., a MLP with $\mathbf{x}_i$ and $\mathbf{c}_k$ as inputs and sparsegen-lin as output layer. The hyperparameter λ_u represents a regularization term shared among all users. Sparsegen-lin is a controllable extension of sparsemax [38], defined in [39] as:

$$\text{sparsegen-lin}(\mathbf{z}; \lambda) = \text{sparsemax}\left(\frac{\mathbf{z}}{1-\lambda}\right)$$

where the coefficient $\lambda < 1$ controls the regularization strength. In particular for $\lambda \to 1^-$, the probability distribution has the minimum support (i.e. *hardmax*) whereas for $\lambda \to -\infty$, the resulting distribution is non-sparse (i.e. *uniform*).

- For each product j we can proceed in an equivalent manner introducing L clusters and generating $\mathbf{x}_j$, $\mathbf{c}_\ell$ and $\theta_{j,\ell}$ employing a different neural network g and using the hyperparameters ξ and λ_p accordingly.
- We now sample a text associated with an edge $e_{i,j}$. Draw each word $w_{i,j,v}$ in the text as follows:

$$\begin{aligned} \mathbf{z}_{i,v} &\sim \text{Categorical}(\theta_i) \\ \mathbf{z}_{j,v} &\sim \text{Categorical}(\theta_j) \\ \mathbf{w}_{i,j,v} &\sim \text{Categorical}(\beta, \mathbf{z}_{i,v}, \mathbf{z}_{j,v}) \end{aligned} \tag{1}$$

Where β is a 3d tensor representing the probabilistic patterns among user clusters, product clusters and words. In particular, $\beta_{k,\ell,:}$ specifies a categorical word distribution conditioned on the user cluster k and the product cluster ℓ. It lies in a $(V-1)$-dimensional simplex Δ^{V-1}, i.e., $\sum_{v=1}^{V} \beta_{k,\ell,v} = 1$ and $\beta_{k,\ell,v} > 0$. V denotes the number of words. The parameter $\beta_{k,\ell,v}$ is computed as:

$$\beta_{k,\ell,v} = \text{sparsemax}(\psi(\mathbf{c}_k, \mathbf{c}_\ell, \mathbf{x}_v; \rho)) \tag{2}$$

The function ψ defines a neural network with parameters ρ. $\mathbf{x}_v$ denotes the pre-trained word vector from Word2Vec [40]. The word sampling is inspired by the topic models. Here we assume every relation and word follow their distributions with distinct parameters computed with functions of the embedding vectors of the involved nodes. Fig. 1 depicts a schematic representation of the model.

B. Generation of Textual Explanations of Node Embeddings

Given the iGNN model, we can now generate textual explanations of node embeddings. For a node, e.g. a user i, the textual explanation is formulated as a node-specific word distribution $p(\mathbf{w}_v|\mathbf{x}_i)$ conditioned on its embedding vector $\mathbf{x}_i$.

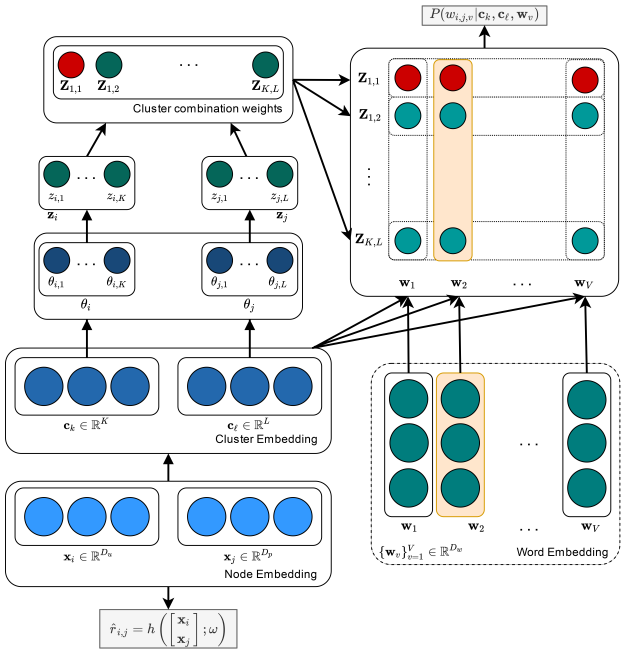


Fig. 1. The schematic view of the iGNN model. Dashed boxes represent input (non-trainable) data. The line connections depict the dependencies between the variables mentioned in Sec. III-A.

In particular, the probability of a word v to be used to explain the embedding $\mathbf{x}_i$ is computed as:

$$p(\mathbf{w}_v|\mathbf{x}_i) = \frac{1}{L} \sum_{k,\ell} \theta_{i,k} \ \beta_{k,\ell,v} \tag{3}$$

This is a marginal distribution over all possible user and product clusters. Since the target distribution is not related to any specific products, the product clusters are equally distributed, i.e. the term $\frac{1}{L}$ in (3). Textual explanations for product nodes can be generated in an equivalent manner.

C. Inference and Learning

The learning of the node embeddings and the corresponding textual explanations is driven by two learning objectives. The first objective is the log-likelihood of the review corpus. The parameters to be learned include embedding vectors of clusters $\{\mathbf{c}_k\}_{k=1}^K$ and $\{\mathbf{c}_\ell\}_{\ell=1}^L$, and parameters ϕ, ξ, γ and ρ that define the neural networks f, g and ψ. Thus, the log-likelihood of an edge and the corresponding text is:

$$\begin{aligned} \mathcal{L}_1 = &\log p(e_{i,j}|\mathbf{x}_i, \mathbf{x}_j, \gamma) + \\ &+ \sum_{v=1}^{S} \log \left(\sum_{k=1}^{K} \sum_{\ell=1}^{L} p(\mathbf{z}_i = k|\mathbf{x}_i, \mathbf{c}_k, \phi) \right. \\ &\left. p(\mathbf{z}_j = \ell|\mathbf{x}_j, \mathbf{c}_\ell, \xi) p(\mathbf{w}_{i,j,v}|\mathbf{c}_k, \mathbf{c}_\ell, \mathbf{x}_v, \rho) \right) \end{aligned} \tag{4}$$

where the first term represents the probability that an edge exists between two nodes. This is implicitly included in the computation of the textual information since we suppose that every edge, if exists, has some associated text.

The second objective is the error of the predictions. In our study case, this will be the rating prediction error.

$$\mathcal{L}_2 = \frac{1}{R}\sum(\hat{r}_{i,j} - r_{i,j})^2 \tag{5}$$

with

$$\hat{r}_{i,j} = h\left(\begin{bmatrix}\mathbf{x}_i\\ \mathbf{x}_j\end{bmatrix};\omega\right) \tag{6}$$

where $h(\cdot)$ can be any complex function. In our case, $h(\cdot)$ defines a deep neural network with the concatenation of the node embeddings $\mathbf{x}_i$ and $\mathbf{x}_j$ as input and ω as hyperparameters.

Finally, we can define the complete objective function as:

$$\min_{\Theta} \mathcal{L} = \min_{\Theta}(\mathcal{L}_1 + \mu\mathcal{L}_2) \tag{7}$$

where Θ represents the parametric space and μ is a hyperparameter to trade-off the importance of the prediction accuracy and the log-likelihood of the corpus. Even if our final goal is to generate textual explanations for node embedding, the prediction accuracy is crucial in the learning phase. Looking at (3), one can see that the textual explanation is conditioned on the node embedding vectors and thus we want these embeddings to perform well at score prediction (Eqs. (5) and (6)). Therefore, by jointly learning embedding vectors, rating predictions and textual explanations, we strengthen the interpretability of the results since the two objectives simultaneously influence each other during the training phase. Given the loss defined in (7), we can use backpropagation to efficiently optimize the model.

IV. Experiments

In this section, we verify the performance of our model with real text-labeled graphs. Firstly, we evaluate the capacity of the model on learning node embeddings that perform well in a downstream link prediction task. Then, we evaluate the generated textual explanations of the node vectors from multiple perspectives.

A. Data sets

We evaluate our method with a popular benchmark dataset: *Amazon Product Data*[1]. The dataset includes millions of product reviews and metadata divided per category collected from May 1999 to July 2014. We focus on the *5-core* version of the data sets, where each user and item has at least 5 associated reviews. The ratings (labels of links between users and items) are integer values between 1 and 5. We use data from 12 categories for our evaluation.

To preprocess the texts associated with the review graphs, we embed the words into a $200-$dimensional vector space using Word2Vec [40]. We train the word vectors with the raw reviews to capture the semantic and syntactic structure of the corpus. We normalize the text via the following steps: (a) set the maximum length of a raw review to 300; (b) lower cased letters; (c) remove stopwords, numbers and special characters; (d) remove non-existing words using an English vocabulary as filter; (e) lemmatization; (f) remove reviews with just one word.

[1]http://jmcauley.ucsd.edu/data/amazon

TABLE I
Statistics of the preprocessed data sets.

	Reviews	Users	Items
Amazon Instant Videos	36575	4178	1686
Automotive	20303	2277	1835
Baby	157447	17589	7050
Grocery, Gourmet Food	149383	12210	8709
Office Products	53085	4276	2421
Patio, Lawn and Garden	13223	1484	963
Pet Supplies	152705	17079	8510
Tools and Home Improv.	132360	14109	10218
Toys and Games	161775	15541	11919
Beauty	192562	19284	12089
Digital Music	64202	4766	3569
Cell Phones and Acc.	190186	24628	10420

The textual information shows strong diversity with respect to product category. For example, words frequently occurring in the *Baby* category would not occur in *Office product* reviews. Therefore, the keywords selection will be performed independently for each category. Let $\mathcal{C}$ denote the review collection for a given category, we consider each review $s_{ij} \in \mathcal{C}$ as a document. For each word $w \in s_{ij}$ we compute the corresponding *tf-idf* index and extract the top 10% words (at most 10 for longer reviews) with respect to such index. Then, we merge all the top-words extracted from the corpus $\mathcal{C}$ and we sort them with respect to the *tf-idf* score. The first V words in this ranking denote the vocabulary for the given category. Finally, we further filter the reviews to remove the ones without any word belonging to the vocabulary. To tackle the word co-occurrence sparsity problem over short texts, we extract biterms for each document [41]. For a node-related document with v words, the resulting biterm extraction results in $v(v-1)/2$ unordered word-pairs. In this way, we can pattern the textual information by means of biterms co-occurring in the same document. Some statistics of the preprocessed data sets are summarized in Table I.

B. Experimental Setting

We set the embedding vector dimensionality $D = 200$ for all users, products and clusters embeddings. We evaluated the robustness of our method to changes in the hyper-parameter D but did not observe any significant performance difference. The number of user clusters $K = 50$ and product clusters $L = 30$; we evaluated the values $[10, 20, 30, 40, 50]$ and we observed that, generally, larger values lead to more sparse cluster assignments. We set the coefficients for the *sparsegen-lin* function as $\lambda_u = 0.9$ and $\lambda_p = 0.75$. The function $h(\cdot)$ defined in (6) is a neural network with four hidden fully-connected layers $[128, 64, 64, 32]$. The experiment was run for 200 learning iterations and validated every 2 iterations. A single epoch performs RMSProp with a learning rate set to 2e-6 and batch size of 256.

TABLE II
MSE FOR IGNN AND STATE-OF-THE-ART APPROACHES.

Category	Offset	Attn+CNN	NMF	SVD	HFT	DeepCoNN	TransRev	iGNN
Amazon Instant Videos	1.180	0.936	0.946	0.904	0.888	0.943	**0.884**	0.923
Automotive	0.948	0.881	0.876	0.857	0.862	**0.753**	0.855	0.827
Baby	1.262	1.176	1.171	1.108	1.104	1.154	1.100	**1.094**
Grocery, Gourmet Food	1.165	1.004	0.985	0.964	0.961	0.973	0.957	**0.924**
Office Products	0.876	0.726	0.742	0.727	0.727	0.738	0.724	**0.665**
Patio, Lawn and Garden	1.156	0.999	0.958	0.950	0.956	1.070	**0.941**	**0.941**
Pet Supplies	1.354	1.236	1.241	1.198	1.194	1.281	1.191	**1.186**
Tools and Home Improv.	1.017	0.938	0.908	0.884	0.884	0.946	**0.879**	**0.879**
Toys and Games	0.975	-	0.821	0.788	0.784	0.851	0.784	**0.775**
Beauty	1.322	-	1.204	1.168	1.165	1.184	1.158	**1.073**
Digital Music	1.137	-	0.805	0.797	0.793	0.835	**0.782**	0.855
Cell Phones and Acc.	1.451	-	1.357	1.290	1.285	1.365	1.279	**1.161**

C. Rating Prediction Results

We compare our method with several baselines considering both factorization-based approaches, like SVD and NMF [42], and review-based approaches (i.e. methods that take advantage of the textual information for improving the rating prediction performance), including HFT [43], DeepCoNN [44], TransRev [45] and Attn+CNN [46]. We also compare our method with a simple baseline that uses the average of the training ratings as prediction.

Following previous works, the data are randomly split by reviews into training (80%), validation (10%) and test (10%) sets. We independently repeat each experiment on five different random splits and report the averaged Mean Squared Error (MSE) to quantitatively evaluate the results. Table II summarizes the results of the compared approaches. Note that we primarily focus on the generation of the textual explanation for node embeddings. This experiment is a direct consequence of how we define the generation of textual explanations (recall (3)) and the loss (as in (7)). We can observe that iGNN outperforms other models on the majority of the data sets, and gets comparable results on the remaining. This clearly confirms the capacity of our proposed method on learning node embedding that competitively perform on a rating prediction task.

D. Analysis of the Probabilistic Patterns

We now investigate the capacity of our model on generating textual explanations for the learned node vector representations in order to create a human-understandable explanation of them. We first evaluate the probabilistic patterns between user clusters, product clusters and words by analyzing the learned word distributions $\beta_{\cdot,\cdot,v}$. As defined in (2), β specifies the word distributions for each combination of user and product cluster. For each category, to visualize the learned word-vector distributions, the TSNE method [47] is employed for dimensionality reduction. Fig. 2 illustrates the cluster organization for different categories. One can find that the clusters are well structured and mapped into the 2-dimensional embedding space with different distributions. For further analysis, we select a pair of clusters from two different categories, and we compute the corresponding average word distribution. In theory, one could infer the main *topic* of each cluster looking at the most probable words of the averaged cluster word distribution. Table III reports the most probable words for each of the selected clusters. The results validate our hypothesis since, for each cluster, we can infer the sub-category of interest. For instance, the first cluster in the *Pet Supplies* category refers to the *grooming* sub-category, while the second one focuses on *aquariums*. Thus, the word distributions $\beta_{\cdot,\cdot,v}$ capture the latent structures of the data and help to find the patterns between user and product clusters, enhancing the interpretability of the results. Indeed, knowing the user and product cluster assignments one can find the *sub-categories* highly correlated with the given items.

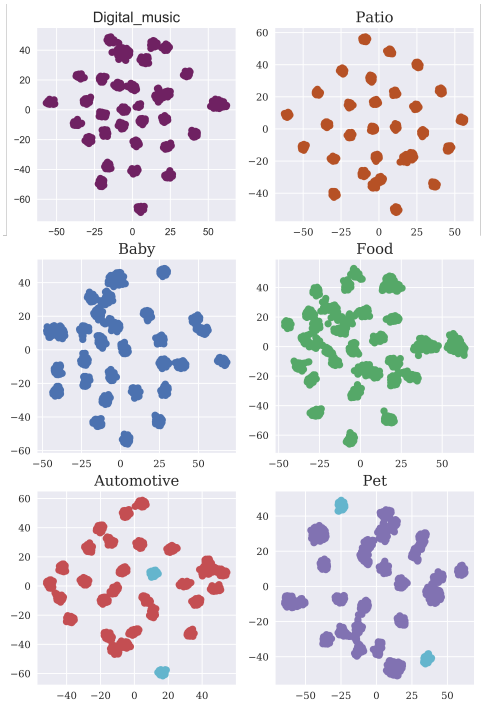


Fig. 2. Cluster organization of the word-vector distributions $\beta_{\cdot,\cdot,v}$ for different categories. The clusters analyzed in Table III are highlighted in cyan.

TABLE III
MOST PROBABLE WORDS OF DIFFERENT AVERAGED CLUSTER WORD-DISTRIBUTIONS.

Automotive		Pet Supplies	
Cluster 1	Cluster 2	Cluster 1	Cluster 2
battery	trailer	work	filter
charge	power	hair	fish
charger	gas	brush	water
power	away	look	gallon
plug	compressor	thing	plant
code	large	fur	heater
cord	pressure	smell	turtle
lead	jeep	quality	flow
transmission	price	wet	clean
phone	guy	stuff	pump
volt	hole	comb	algae
change	weight	shed	shrimp
adapter	fast	groom	work
smell	fuel	flea	gravel
connect	space	shampoo	tube
electronics	*performances*	*grooming*	*aquariums*

E. Evaluation of Generated Textual Explanations

To investigate whether our model is able to generate textual explanations as node-specific word distributions, we attempt to provide both quantitative and qualitative evaluation of the node-related explanations. As reported in [48], lacking common metrics for interpretability quality is problematic to the research community to make progress in this area and, in an unsupervised setting, the problem is even more clear. Indeed, it is common for researchers to simply rely on the human visualization of the results, e.g. employing attention mechanisms or heat maps, in order to make the results human-explainable [26]. However, although our case study can be evaluated by just relying on human perception of the highlighted relevant words, we try to introduce some metrics that could further help on assessing the quality of the results. To visualize and evaluate the correlation between the generated word distributions and the node-related words in the data, we proceed as follows. Given a sampled node, we first extract the *top*-15 words in the generated word distribution, i.e. the 15 words having the highest probability in the distribution. Second, we extract the set of words associated with this specific node in the data. Let denote with A and B respectively, these two sets.

The Jaccard similarity is used to measure similarity between two sets of words A and B, which is defined as follows:

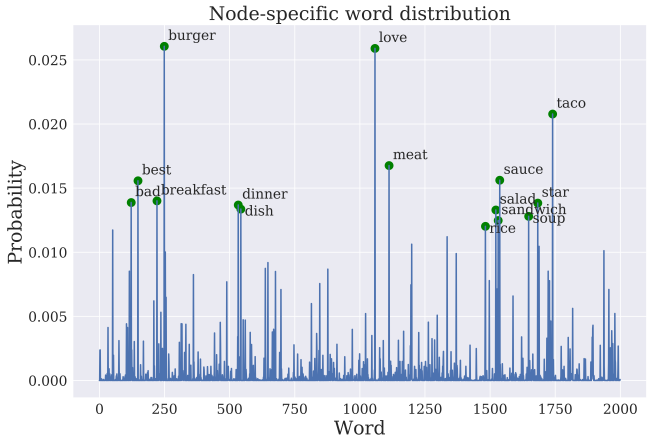
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{8}$$

To further capture the semantic similarity of the two sets, we integrate the Word Mover's Distance (WDM), introduced by [49]; this metric takes into account the word similarities in the word embedding space and computes the minimum distance that the words in text A need to *travel* in the semantic space to reach the words in text B. Since Jaccard and WDM have different scales and behaviors, we apply MinMaxScaler to Jaccard, and transform WDM as follows:

$$\text{t-}WDM(A,B) = 1 - \left(\frac{d_i - min(\mathbf{d})}{max(\mathbf{d}) - min(\mathbf{d})} \right) \tag{9}$$

where d_i is the distance between the sets A and B for a node i, and $\mathbf{d}$ represents all the distances between the two sets for each node in the graph. In this way, the transformed distance score is in the range [0,1] and, opposite to the definition of semantic distance, the higher the value the closer are sets A and B.

We investigate the quality of the generated textual explanations by computing these scores for different data sets. Fig. 3 shows an example of the generated word distribution of a sampled node. Fig. 4 illustrates the distribution of values of these measures across the nodes in the data sets. The Jaccard values mostly vary in the range $[0.4 - 0.7]$, while the WDM scores are highly concentrated in the range $[0.6 - 0.8]$.



TOP-15 WORDS	bad **best breakfast** **burger** dinner dish **love meat** **rice salad sandwich** **sauce** soup star **taco**
NODE-RELATED WORDS	ambiance authentic **best breakfast** choice cilantro decision guacamole **love meat** pas plain satisfied **sauce** spacious staple **taco**
	t-J(A,B) = 0.696 t-WDM(A,B) = 0.718

Fig. 3. Intepretability case study on a random node. The figure depicts the node-specific word distribution; the 15 highest probabilities are highlighted by green points. TOP-15 WORDS and NODE-RELATED WORDS refer to the sets A and B defined in (8)-(9). Black bold represents the overlapping words; blue bold highlights words that may explain further characteristics of the node.

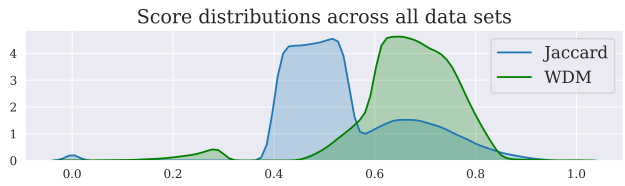


Fig. 4. Evaluation of the metric distributions across all data sets.

The lower Jaccard scores do not affect the performances of the model, instead, confirms that our model generates textual explanations that are not redundant as would have been with too high similarity scores. Indeed, as written in [26], two sets of words can be semantically similar even with low lexical overlapping. Moreover, this proves that our approach is not solely based on lexical similarity, but takes into account the semantic similarity of the words generating textual explanations that are not too similar to the original data. We further evaluate the results on the *Baby*, *Gourmet Food* and *Pet Supplies* data sets. The example nodes in Table IV demonstrate the effectiveness of iGNN to capture the most probable words associated with a given node. In particular, not only direct correspondences with the words in the data but also highly related words. For example, looking at the sampled node from the *Baby* category, we can observe that the generated explanation might be used to better capture further characteristics of the given node. Specifically, it looks like the node is related to toys and strollers and, analyzing the blue bold highlighted words, we can infer that the items should also be safe and made with good quality materials. Additionally, we can notice the impact of the scores on the quality of the generated word distributions. The first two examples, taken from the *Gourmet Food* data set, have different Jaccard scores while similar distances. Nevertheless, the first example looks semantically good. This shows the importance of considering the word vector representations during the training phase since they can *push* the probabilities to words semantically closer.

We can also observe that: (a) the model does not take into account the polarity of the words. An extension could be achieved by exploiting the sentiment (or rating) information to get *positive/negative* word distributions; (b) the quality of vocabulary could be improved by using more sophisticated techniques that might potentially lead to better performances.

V. Conclusions

In this paper we present interpretable graph neural networks (iGNN), a neural generative model that uses the extra-textual information associated with a graph to learn human-understandable explanations for node embedding. To strengthen the interpretability of the results, the model learns simultaneously node embeddings and textual explanations during the training phase. In addition, the introduction of node cluster embeddings enables the model to learn the patterns among nodes, ratings and textual information. In this way, iGNN learns the discrete structure of the graph data and the text-based explanations in an elegant way. Finally, recalling (5) and (6), our model offers a flexible framework since it can be integrated with different learning tasks. In our case, given the availability of the rating information, we perform a rating prediction task, but the second term of the complete objective function defined in (7) may be changed according to the desired task. The promising results in the qualitative analysis show the capacity of the model on generating human-understandable explanations while competitively performing well on a prediction task. Indeed, the model is able to explain the meaning of the learned node representations in a human interpretable way. The additional quantitative analysis of the textual explanations helps at assessing the quality of the results. In contrast with the simple visualization of sampled outputs, the proposed investigation enables to comprehensively understand the general model behavior in terms of textual explainability. For future work, it may be interesting to extend the approach by integrating the polarity of the words and employing the interpretation for downstream tasks, such as human-understandable recommendations.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 766186.

TABLE IV
EVALUATION OF RETRIEVED NODES FROM DIFFERENT DATA SETS. BLACK BOLD REPRESENTS THE MATCHING WORDS; BLUE BOLD HIGHLIGHTS WORDS THAT MAY EXPLAIN FURTHER CHARACTERISTICS OF THE NODE.

Gourmet Food	
Top-15 Words	**butter buy cereal chocolate** cracker **delicious** eat **energy honey milk nut package** price **say time**
Node-related Words	almond alternative arrive **butter buy** container creamy date deal **delicious** healthy jar mess **nut** oil oily order **package** peanut plastic pull **say** size stir **time** variety way
	t-J(A,B) = 0.539 t-WDM(A,B) = 0.748
Top-15 Words	**best** buy **chocolate cup dark delicious** eat mix nice **organic price say strong tea time**
Node-related Words	agree arrive bad **best** big **chocolate** cookie crack **cup delicious** disappear early equal expect forever heat package picky **price** quite **say** shelf sleeve staff **tea time** week
	t-J(A,B) = 0.640 t-WDM(A,B) = 0.753
Baby	
Top-15 Words	**color** cup cute **daughter** hard **item material nice perfect product** pump **quality safe stroller toy**
Node-related Words	bouncer **color** comfy **daughter** flop happy insert issue **item** lot **perfect product** return **stroller** swing tiny **toy**
	t-J(A,B) = 0.777 t-WDM(A,B) = 0.731
Pet Supplies	
Top-15 Words	**clean color** eat **filter good** help **long look** order **size small thing water week work**
Node-related Words	bottle brush care chip **clean** daily drink excite **good** last **long look** nice puppy **small** smell spray stay **thing** triple type **water week** white **work**
	t-J(A,B) = 0.943 t-WDM(A,B) = 0.893

REFERENCES

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[2] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of SIGKDD*, 2016.

[3] T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*, 2017.

[4] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of SIGKDD*, 2014.

[5] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto. Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In *Proceedings of IJCAI*, 2017.

[6] R. Socher, D. Chen, C. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, 2013.

[7] S. Rhee, S. Seo, and S. Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*, 2017.

[8] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur. Protein interface prediction using graph convolutional networks. In *NIPS*, 2017.

[9] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Simaan. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of EMNLP*, 2017.

[10] D. Beck, G. Haffari, and T. Cohn. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of ACL*, 2018.

[11] L. Song, Y. Zhang, Z. Wang, and D. Gildea. N-ary relation extraction using graph state lstm. *arXiv preprint arXiv:1808.09101*, 2018.

[12] Y. Zhang, P. Qi, and C. D. Manning. Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185*, 2018.

[13] M. Narasimhan, S. Lazebnik, and A. Schwing. Out of the box: Reasoning with graph convolution nets for factual visual question answering. In *Proceedings of NeurIPS*, 2018.

[14] D. Teney, L. Liu, and A. Den Hengel. Graph-structured representations for visual question answering. In *Proceedings of CVPR*, 2017.

[15] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *Proceedings of CVPR*, 2018.

[16] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of WWW*, 2015.

[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua. Neural collaborative filtering. In *Proceedings of WWW*, 2017.

[18] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *Proceedings of ICLR*, 2018.

[19] R. Ying, R. He, K. Chen, P. Eksombatchai, W. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of KDD*, 2018.

[20] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1722–1731, 2017.

[21] Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Improved semantic-aware network embedding with fine-grained word alignment. *arXiv preprint arXiv:1808.09633*, 2018.

[22] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[23] W. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[24] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[25] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[26] Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Sam Gershman, and Finale Doshi-Velez. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*, 2019.

[27] M. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you: Explaining the predictions of any classifier. In *Proceedings of KDD*, 2016.

[28] S. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

[29] Fulton Wang and Cynthia Rudin. Falling rule lists. In *Artificial Intelligence and Statistics*, pages 1013–1022, 2015.

[30] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

[31] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[32] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[33] Carolin Lawrence, T. Sztyler, and Mathias Niepert. Explaining neural matrix factorization with gradient rollback. *ArXiv*, abs/2010.05516, 2020.

[34] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[35] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[36] Saba Al-Sayouri, Ekta Gujral, Danai Koutra, Evangelos E Papalexakis, and Sarah S Lam. t-pine: Tensor-based predictable and interpretable node embeddings. *Social Network Analysis and Mining*, 10(1):1–11, 2020.

[37] Chi Thang Duong, Quoc Viet Hung Nguyen, and Karl Aberer. Interpretable node embeddings with mincut loss. 2019.

[38] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623, 2016.

[39] Anirban Laha, Saneem Ahmed Chemmengath, Priyanka Agrawal, Mitesh Khapra, Karthik Sankaranarayanan, and Harish G Ramaswamy. On controllable sparse alternatives to softmax. In *Advances in Neural Information Processing Systems*, pages 6422–6432, 2018.

[40] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[41] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456, 2013.

[42] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[43] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[44] Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 425–434, 2017.

[45] Alberto Garcia-Duran, Roberto Gonzalez, Daniel Onoro-Rubio, Mathias Niepert, and Hui Li. Transrev: Modeling reviews as translations from users to items. *arXiv preprint arXiv:1801.10095*, 2018.

[46] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Representation learning of users and items for review rating prediction using attention-based convolutional neural network. In *3rd international workshop on machine learning methods for recommender systems (MLRec)(SDM'17)*, 2017.

[47] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[48] Philipp Schmidt and Felix Biessmann. Quantifying interpretability and trust in machine learning systems. *arXiv preprint arXiv:1901.08558*, 2019.

[49] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966, 2015.