Embedding Biomedical Ontologies by Jointly Encoding Network Structure and Textual Node Descriptors

Sotiris Kotitsas[1], Dimitris Pappas[1,2], Ion Androutsopoulos[1], Ryan McDonald[1,3] and Marianna Apidianaki[4]

[1]Department of Informatics, Athens University of Economics and Business, Greece
[2]Institute for Language and Speech Processing, Research Center 'Athena', Greece
[3]Google Research
[4]CNRS, LLF, Univ. Paris Diderot, France

{p3150077, pappasd, ion}@aueb.gr
ryanmcd@google.com, marianna@limsi.fr

Abstract

Network Embedding (NE) methods, which map network nodes to low-dimensional feature vectors, have wide applications in network analysis and bioinformatics. Many existing NE methods rely only on network structure, overlooking other information associated with the nodes, e.g., text describing the nodes. Recent attempts to combine the two sources of information only consider local network structure. We extend NODE2VEC, a well-known NE method that considers broader network structure, to also consider textual node descriptors using recurrent neural encoders. Our method is evaluated on link prediction in two networks derived from UMLS. Experimental results demonstrate the effectiveness of the proposed approach compared to previous work.

1 Introduction

Network Embedding (NE) methods map each node of a network to an embedding, meaning a low-dimensional feature vector. They are highly effective in network analysis tasks involving predictions over nodes and edges, for example link prediction (Lu and Zhou, 2010), and node classification (Sen et al., 2008).

Early NE methods, such as DEEPWALK (Perozzi et al., 2014), LINE (Tang et al., 2015), NODE2VEC (Grover and Leskovec, 2016), GCNs (Kipf and Welling, 2016), leverage information from the network structure to produce embeddings that can reconstruct node neighborhoods. The main advantage of these *structure-oriented* methods is that they encode the network context of the nodes, which can be very informative. The downside is that they typically treat each node as an atomic unit, directly mapped to an embedding in a look-up table (Fig. 1a). There is no attempt to model information other than the network structure, such as textual descriptors (labels) or other meta-data associated with the nodes.

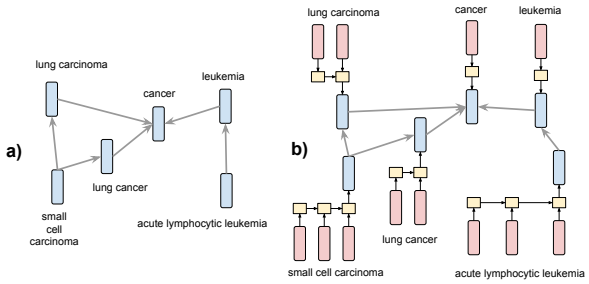


Figure 1: Example network with nodes associated with textual descriptors. a) A model where each node is represented by a vector (node embedding) from a look-up table. b) A model where each node embedding is generated compositionally from the word embeddings of its descriptor via an RNN. The latter model can learn node embeddings from both the network structure and the word sequences of the textual descriptors.

More recent NE methods, e.g., CANE (Tu et al., 2017), WANE (Shen et al., 2018), produce embeddings by combining the network structure and the text associated with the nodes. These *content-oriented* methods embed networks whose nodes are rich textual objects (often whole documents). They aim to capture the compositionality and semantic similarities in the text, encoding them with deep learning methods. This approach is illustrated in Fig. 1b. However, previous methods of this kind considered impoverished network contexts when embedding nodes, usually single-edge hops, as opposed to the non-local structure considered by most structure-oriented methods.

When embedding biomedical ontologies, it is important to exploit both wider network contexts and textual node descriptors. The benefit of the latter is evident, for example, in 'acute leukemia' IS-A 'leukemia'. To be able to predict (or reconstruct) this IS-A relation from the embeddings of 'acute leukemia' and 'leukemia' (and the word embeddings of their textual descriptors in Fig. 1b),

a NE method only needs to model the role of 'acute' as a modifier that can be included in the descriptor of a node (e.g., a disease node) to specify a sub-type. This property can be learned (and encoded in the word embedding of 'acute') if several similar IS-A edges, with 'acute' being the only extra word in the descriptor of the sub-type, exist in the network. This strategy would not however be successful in 'p53' (a protein) IS-A 'tumor suppressor', where no word in the descriptors frequently denotes sub-typing. Instead, by considering the broader network context of the nodes (i.e. longer paths that connect them), a NE method can detect that the two nodes have common neighbors and, hence, adjust the two node embeddings (and the word embeddings of their descriptors) to be close in the representation space, making it more likely to predict an IS-A relation between them.

We propose a new NE method that leverages the strengths of both structure and content-oriented approaches. To exploit wide network contexts, we follow NODE2VEC (Grover and Leskovec, 2016) and generate random walks to construct the network neighborhood of each node. The SKIPGRAM model (Mikolov et al., 2013) is then used to learn node embeddings that successfully predict the nodes in each walk, from the node at the beginning of the walk. To enrich the node embeddings with information from their textual descriptors, we replace the NODE2VEC look-up table with various architectures that operate on the word embeddings of the descriptors. These include simply averaging the word embeddings of a descriptor, and applying recurrent deep learning encoders. The proposed method can be seen as an extension of NODE2VEC that incorporates textual node descriptors. We evaluate several variants of the proposed method on link prediction, a standard evaluation task for NE methods. We use two biomedical networks extracted from UMLS (Bodenreider, 2004), with PART-OF and IS-A relations, respectively. Our method outperforms several existing structure and content-oriented methods on both datasets. We make our datasets and source code available.[1]

2 Related work

Network Embedding (NE) methods, a type of representation learning, are highly effective in network analysis tasks involving predictions over nodes and edges. Link prediction has been extensively studied in social networks (Wang et al., 2015), and is particularly relevant to bioinformatics where it can help, for example, to discover interactions between proteins, diseases, and genes (Lei and Ruan, 2013; Shojaie, 2013; Grover and Leskovec, 2016). Node classification can also help analyze large networks by automatically assigning roles or labels to nodes (Ahmed et al., 2018; Sen et al., 2008). In bioinformatics, this approach has been used to identify proteins whose mutations are linked with particular diseases (Agrawal et al., 2018).

A typical structure-oriented NE method is DEEPWALK (Perozzi et al., 2014), which learns node embeddings by applying WORD2VEC's SKIPGRAM model (Mikolov et al., 2013) to node sequences generated via random walks on the network. NODE2VEC (Grover and Leskovec, 2016) explores different strategies to perform random walks, introducing hyper-parameters to guide them and generate more flexible neighborhoods. LINE (Tang et al., 2015) learns node embeddings by exploiting first- and second-order proximity information in the network. Wang et al. (2016) learn node embeddings that preserve the proximity between 2-hop neighbors using a deep autoencoder. Yu et al. (2018) encode node sequences generated via random walks, by mapping the walks to low dimensional embeddings, through an LSTM autoencoder. To avoid overfitting, they use a generative adversarial training process as regularization. Graph Convolutional Networks (GCNs) are a graph encoding framework that also falls within this paradigm (Kipf and Welling, 2016; Schlichtkrull et al., 2018). Unlike other methods that use random walks or static neighbourhoods, GCNs use iterative neighbourhood averaging strategies to account for non-local graph structure. All the aforementioned methods only encode the structural information into node embeddings, ignoring textual or other information that can be associated with the nodes of the network.

Previous work on biomedical ontologies (e.g., Gene Ontology, GO) suggested that their terms, which are represented through textual descriptors, have compositional structure. By modeling it, we can create richer representations of the data encoded in the ontologies (Mungall, 2004; Ogren et al., 2003, 2004). Ogren et al. (2003) strengthen

[1] https://github.com/SotirisKot/Content-Aware-N2V

the argument of compositionality by observing that many GO terms contain other GO terms. Also, they argue that substrings that are not GO terms appear frequently and often indicate semantic relationships. Ogren et al. (2004) use finite state automata to represent GO terms and demonstrate how small conceptual changes can create biologically meaningful candidate terms.

In other work on NE methods, CENE (Sun et al., 2016) treats textual descriptors as a special kind of node, and uses bidirectional recurrent neural networks (RNNs) to encode them. CANE (Tu et al., 2017) learns two embeddings per node, a text-based one and an embedding based on network structure. The text-based one changes when interacting with different neighbors, using a mutual attention mechanism. WANE (Shen et al., 2018) also uses two types of node embeddings, text-based and structure-based. For the text-based embeddings, it matches important words across the textual descriptors of different nodes, and aggregates the resulting alignment features. In spite of performance improvements over structure-oriented approaches, these content-aware methods do not thoroughly explore the network structure, since they consider only direct neighbors.

By contrast, we utilize NODE2VEC to obtain wider network neighborhoods via random walks, a typical approach of structure-oriented methods, but we also use RNNs to encode the textual descriptors, as in some content-oriented approaches. Unlike CENE, however, we do not treat texts as separate nodes; unlike CANE, we do not learn separate embeddings from texts and network structure; and unlike WANE, we do not align the descriptors of different nodes. We generate the embedding of each node from the word embeddings of its descriptor via the RNN (Fig. 1), but the parameters of the RNN, the word embeddings, hence also the node embeddings are updated during training to predict NODE2VEC's neighborhoods.

Although we use NODE2VEC to incorporate network context in the node embeddings, other neighborhood embedding methods, such as GCNs, could easily be used too. Similarly, text encoders other than RNNs could be applied. For example, Mishra et al. (2019) try to detect abusive language in tweets with a semi-supervised learning approach based on GCNs. They exploit the network structure and also the labels associated with the tweets, taking into account the linguistic behavior of the authors.

3 Proposed Node Embedding Approach

Consider a network (graph) $G = \langle V, E, S \rangle$, where V is the set of nodes (vertices); $E \subseteq V \times V$ is the set of edges (links) between nodes; and S is a function that maps each node $v \in V$ to its textual descriptor $S(v) = \langle w_1, w_2, \ldots, w_n \rangle$, where n is the word length of the descriptor, and each word w_i comes from a vocabulary W. We consider only undirected, unweighted networks, where all edges represent instances of the same (single) relationship (e.g., IS-A or PART-OF). Our approach, however, can be extended to directed weighted networks with multiple relationship types. We learn an embedding $f(v) \in \mathbb{R}^d$ for each node $v \in V$. As a side effect, we also learn a word embedding $e(w)$ for each vocabulary word $w \in W$.

To incorporate structural information into the node embeddings, we maximize the *predicted* probabilities $p(u|v)$ of observing the *actual* neighbors $u \in N(v)$ of each 'focus' node $v \in V$, where $N(v)$ is the neighborhood of v, and $p(u|v)$ is predicted from the node embeddings of u and v. The neighbors $N(v)$ of v are not necessarily directly connected to v. In real-world networks, especially biomedical, many nodes have few direct neighbors. We use NODE2VEC (Grover and Leskovec, 2016) to obtain a larger neighborhood for each node v, by generating random walks from v. For every focus node $v \in V$, we compute r random walks (paths) $P_{v,i} = \langle v_{i,1} = v, v_{i,2}, ..., v_{i,k} \rangle$ $(i = 1, \ldots, r)$ of fixed length k through the network $(v_{i,j} \in V)$.[2] The predicted probability $p(v_{i,j} = u)$ of observing node u at step j of a walk $P_{v,i}$ that starts at focus node v is taken to depend only on the embeddings of u, v, i.e., $p(v_{i,j} = u) = p(u|v)$, and can be estimated with a softmax as in the SKIPGRAM model (Mikolov et al., 2013):

$$p(u|v) = \frac{\exp(f'(u) \cdot f(v))}{\sum_{u' \in V} \exp(f'(u') \cdot f(v))} \quad (1)$$

where it is assumed that each node v has two different node embeddings, $f(v), f'(v)$, used when

[2] Our networks are unweighted, hence we use uniform edge weighting to traverse them. NODE2VEC has two hyperparameters, p, q, to control the locality of the walk. We set $p = q = 1$ (default values). For efficiency, NODE2VEC actually performs r random walks of length $l \geq k$; then it uses r sub-walks of length k that start at each focus node.

v is the focus node or the predicted neighbor, respectively, and $\cdot$ denotes the dot product. NODE2VEC minimizes the following objective function:

$$L = -\sum_{v \in V} \sum_{i=1}^{r} \sum_{j=2}^{k} \log p(v_{i,j} | v_{i,1} = v) \quad (2)$$

in effect maximizing the likelihood of observing the actual neighbors $v_{i,j}$ of each focus node v that are encountered during the r walks $P_{v,i} = \langle v_{i,1} = v, v_{i,2}, ..., v_{i,k} \rangle$ $(i = 1, \ldots, r)$ from v. Calculating $p(u|v)$ using a softmax (Eq. 1) is computationally inefficient. We apply negative sampling instead, as in WORD2VEC (Mikolov et al., 2013). Thus, NODE2VEC is analogous to SKIP-GRAM WORD2VEC, but using random walks from each focus node, instead of using a context window around each focus word in a corpus.

As already mentioned, the original NODE2VEC does not consider the textual descriptors of the nodes. It treats each node embedding $f(v)$ as a vector representing an atomic unit, the node v; a look-up table directly maps each node v to its embedding $f(v)$. This does not take advantage of the flexibility and richness of natural language (e.g., synonyms, paraphrases), nor of its compositional nature. To address this limitation, we substitute the look-up table where NODE2VEC stores the embedding $f(v)$ of each node v with a neural sequence encoder that produces $f(v)$ from the word embeddings of the descriptor $S(v)$ of v.

More formally, let every word $w \in W$ have two embeddings $e(w)$ and $e'(w)$, used when w occurs in the descriptor of a focus node, and when w occurs in the descriptor of a neighbor of a focus node (in a random walk), respectively. For every node $v \in V$ with descriptor $S(v) = (w_1, \ldots, w_n)$, we create the sequences $T(v) = \langle e(w_1), \ldots, e(w_n) \rangle$ and $T'(v) = \langle e'(w_1), \ldots, e'(w_n) \rangle$. We then set $f(v) = \text{ENC}(T(v))$ and $f'(v) = \text{ENC}(T'(v))$, where ENC is the sequence encoder. We outline below three specific possibilities for ENC, though it can be any neural text encoder. Note that the embeddings $f(v)$ and $f'(v)$ of each node v are constructed from the word embeddings $T(v)$ and $T'(v)$, respectively, of its descriptor $S(v)$ by the encoder ENC. The word embeddings of the descriptor and the parameters of ENC, however, are also optimized during back-propagation, so that the resulting node embeddings will predict (Eq. 1) the actual neighbors of each focus node (Fig. 2).

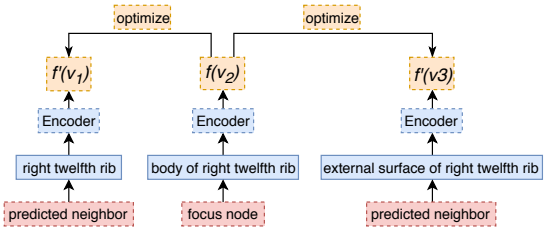


Figure 2: Illustration of the proposed NE approach.

For simplicity, we only mention $f(v)$ and $T(v)$ below, but the same applies to $f'(v)$ and $T'(v)$.

AVG-N2V: For every node $v \in V$, this model constructs the node's embedding $f(v)$ by simply averaging the word embeddings $T(v) = \langle e(w_1), \ldots, e(w_n) \rangle$ of $S(v) = (w_1, w_2, \ldots, w_n)$.

$$f(v) = \frac{1}{n} \sum_{i=1}^{n} e(w_i) \quad (3)$$

GRU-N2V: Although averaging word embeddings is effective in text categorization (Joulin et al., 2016), it ignores word order. To account for order, we apply RNNs with GRU cells (Cho et al., 2014) instead. For each node $v \in V$ with descriptor $S(v) = \langle w_1, \ldots, w_n \rangle$, this method computes n hidden state vectors $H = \langle h_1, \ldots, h_n \rangle = \text{GRU}(e(w_1), \ldots, e(w_n))$. The last hidden state vector h_n is the node embedding $f(v)$.

BIGRU-MAX-RES-N2V: This method uses a bidirectional RNN (Schuster and Paliwal, 1997). For each node v with descriptor $S(v) = \langle w_1, w_2, \ldots, w_n \rangle$, a bidirectional GRU (BIGRU) computes two sets of n hidden state vectors, one for each direction. These two sets are then added to form the output H of the BIGRU:

$$H_f = \text{GRU}_f(e(w_1), \ldots, e(w_n)) \quad (4)$$
$$H_b = \text{GRU}_b(e(w_1), \ldots, e(w_n)) \quad (5)$$
$$H = H_f + H_b \quad (6)$$

where $_f$, $_b$ denote the forward and backward directions, and $+$ indicates component-wise addition. We add residual connections (He et al., 2015) from each word embedding $e(w_t)$ to the corresponding hidden state h_t of H. Instead of using the final forward and backward states of H, we apply max-pooling (Collobert and Weston, 2008; Conneau et al., 2017) over the state vectors h_t of H. The output of the max pooling is the node embedding $f(v)$. Figure 3 illustrates this method.

Additional experiments were conducted with several variants of the last encoder. A unidirectional GRU instead of a BIGRU, and a BIGRU with

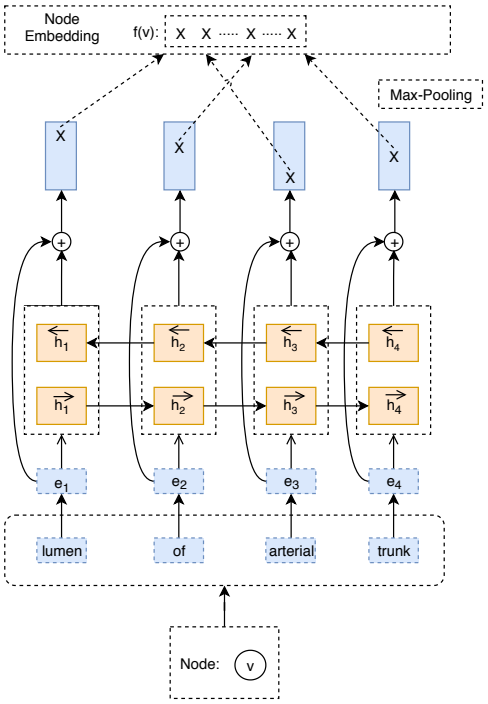


Figure 3: Obtaining the embedding of a node v by applying a BIGRU encoder with max-pooling and residuals to the embeddings of v's textual descriptor.

self-attention (Bahdanau et al., 2015) instead of max-pooling were also tried. To save space, we described only the best performing variant.

4 Experiments

We investigate the effectiveness of our proposed approach by conducting link prediction experiments on two biomedical datasets derived from UMLS. Furthermore, we devise a new approach of generating negative edges for the link prediction evaluation – beyond just random negatives – that makes the problem more difficult and aligns more with real-world use-cases. We also conduct a qualitative analysis, showing that the proposed framework does indeed leverage both the textual descriptors and the network structure.

4.1 Datasets

We created our datasets from the UMLS ontology, which contains approx. 3.8 million biomedical concepts and 54 semantic relationships. The relationships become edges in the networks, and the concepts become nodes. Each concept (node) is associated with a textual descriptor. We extract two types of semantic relationships, creating two networks. The first, and smaller one, consists of PART-OF relationships where each node represents a part of the human body. The second network contains IS-A relationships, and the concepts represented by the nodes vary across the spectrum of biomedical entities (diseases, proteins, genes, etc.). To our knowledge, the IS-A network is one of the largest datasets employed for link prediction and learning network embeddings. Statistics for the two datasets are shown in Table 1.

Statistics	IS-A	PART-OF
Nodes	294,693	16,894
Edges	356,541	19,436
Training true positive edges	294,692	16,893
Training true negative edges	294,692	16,893
Test true positive edges	61,849	2,543
Test true negative edges	61,849	2,543
Avg. descriptor length	5 words	6 words
Max. descriptor length	31 words	14 words

Table 1: Statistics of the two datasets (IS-A, PART-OF). The true positive and true negative edges are used in the link prediction experiments.

4.2 Baseline Node Embedding Methods

We compare our proposed methods to baselines of two types: *structure-oriented* methods, which solely focus on network structure, and *content-oriented* methods that try to combine the network structure with the textual descriptors of the nodes (albeit using impoverished network neighborhoods so far). For the first type of methods, we employ NODE2VEC (Grover and Leskovec, 2016), which uses a biased random walk algorithm based on DEEPWALK (Perozzi et al., 2014) to explore the structure of the network more efficiently. Our work can be seen as an extension of NODE2VEC that incorporates textual node descriptors, as already discussed, hence it is natural to compare to NODE2VEC. As a *content-oriented* baseline we use CANE (Tu et al., 2017), which learns separate text-based and network-based embeddings, and uses a mutual attention mechanism to dynamically change the text-based embeddings for different neighbors (Section 2). CANE only considers the direct neighbors of each node, unlike NODE2VEC, which considers larger neighborhoods obtained via random walks.

4.3 Link Prediction

In link prediction, we are given a network with a certain fraction of edges removed. We need to infer these missing edges by observing the incomplete network, facilitating the discovery of links (e.g., unobserved protein-protein interactions).

Concretely, given a network we first randomly remove some percentage of edges, ensuring that

the network remains connected so that we can perform random walks over it. Each removed edge e connecting nodes v_1, v_2 is treated as a *true positive*, in the sense that a link prediction method should infer that an edge should be added between v_1, v_2. We also use an equal number of *true negatives*, which are pairs of nodes v'_1, v'_2 with no edge between v'_1, v'_2 in the original network. When evaluating NE methods, a link predictor is given true positive and true negative pairs of nodes, and is required to discriminate between the two classes by examining only the node embeddings of each pair. Node embeddings are obtained by applying a NE method to the pruned network, i.e., after removing the true positives. A NE method is considered better than another one, if it leads to better performance of the same link predictor.

We experiment with two approaches to obtain true negatives. In *Random Negative Sampling*, we randomly select pairs of nodes that were not directly connected (by a single edge) in the original network. In *Close Proximity Negative Sampling*, we iterate over the nodes of the original network considering each one as a focus. For each focus node v, we want to find another node u in close proximity that is not an ancestor or descendent (e.g., parent, grandparent, child, grandchild) of v in the IS-A or PART-OF hierarchy, depending on the dataset. We want u to be close to v, to make it more difficult for the link predictor to infer that u and v should not be linked. We do not, however, want u to be an ancestor or descendent of v, because the IS-A and PART-OF relationships of our datasets are transitive. For example, if u is a grandparent of v, it could be argued that inferring that u and v should be linked, is *not* an error. To satisfy these constraints, we first find the ancestors of v that are between 2 to 5 hops away from v in the original network.[3] We randomly select one of these ancestors, and then we randomly select as u one of the ancestor's children in the original network, ensuring that u was not an ancestor or descendent of v in the original network. In both approaches, we randomly select as many true negatives as the true positives, discarding the remaining true negatives.

We experimented with two link predictors:

Cosine similarity link predictor (CS): Given a pair of nodes v_1, v_2 (true positive or true negative edge), CS computes the cosine similarity (ignoring negative scores) between the two node embeddings as $s(v_1, v_2) = \max(0, \cos(f(v_1), f(v_2)))$, and predicts an edge between the two nodes if $s(v_1, v_2) \geq t$, where t is a threshold. We evaluate the predictor on the true positives and true negatives (shown as 'test' true positives and 'test' true negatives in Table 1) by computing AUC (area under ROC curve), in effect considering the precision and recall of the predictor for varying t.[4]

Logistic regression link predictor (LR): Given a pair of nodes v_1, v_2, LR computes the Hadamard (element-wise) product of the two node embeddings $f(v_1) \odot f(v_2)$ and feeds it to a logistic regression classifier to obtain a probability estimate p that the two nodes should be linked. The predictor predicts an edge between v_1, v_2 if $p \geq t$. We compute AUC on a test set by varying t. The test set of this predictor is the same set of true positives and true negatives (with Random or Close Proximity Negative Sampling) that we use when evaluating the CS predictor. The training set of the logistic regression classifier contains as true positives all the other edges of the network that remain after the true positives of the test set have been removed, and an equal number of true negatives (with the same negative sampling method as in the test set) that are not used in the test set.

4.4 Implementation Details

For NODE2VEC and our NE methods, which can be viewed as extensions of NODE2VEC, the dimensionality of the node embeddings is 30. The dimensionality of the word embeddings (in our NE methods) is also 30. In the random walks, we set r = 5, l = 40, k = 10 for IS-A, and r = 10, l = 40, k = 10 for PART-OF; these hyper-parameters were not particularly tuned, and their values were selected mostly to speed up the experiments. We train for one epoch with a batch size of 128, setting the number of SKIPGRAM's negative samples to 2. We use the Adam (Kingma and Ba, 2015) optimizer in our NE methods. We implemented our NE methods and the two link predictors using PyTorch (Paszke et al., 2017) and Scikit-Learn (Pedregosa et al., 2011). For NODE2VEC and CANE, we used the implementations provided.[5]

[3] The edges of the resulting datasets are not directed. Hence, looking for descendents would be equivalent.

[4] We do not report precision, recall, F1 scores, because these require selecting a particular threshold t values.

[5] See https://github.com/aditya-grover/node2vec, https://github.com/thunlp/CANE.

NE Method + Link Predictor	Random Negative Sampling	Close Proximity Sampling
Node2Vec + CS	66.6	54.3
CANE + CS	94.1	69.6
Avg-N2V + CS	95.0	78.6
GRU-N2V + CS	**98.7**	**79.2**
BiGRU-Max-Res-N2V + CS	98.5	79.0
Node2Vec + LR	77.2	56.3
CANE + LR	95.3	70.0
Avg-N2V + LR	97.6	73.9
GRU-N2V + LR	99.0	79.6
BiGRU-Max-Res-N2V + LR	**99.3**	**82.1**

Table 2: AUC scores (%) for the IS-A dataset. Best scores per link predictor (CS, LR) shown in bold.

NE Method + Link Predictor	Random Negative Sampling	Close Proximity Sampling
Node2Vec + CS	76.8	61.8
CANE + CS	93.9	75.3
Avg-N2V + CS	95.9	81.8
GRU-N2V + CS	98.0	83.1
BiGRU-Max-Res-N2V + CS	**98.5**	**83.3**
Node2Vec + LR	85.2	66.5
CANE + LR	94.4	76.3
Avg-N2V + LR	97.6	79.4
GRU-N2V + LR	99.0	85.6
BiGRU-Max-Res-N2V + LR	**99.5**	**88.6**

Table 3: AUC scores (%) for the PART-OF dataset. Best scores per link predictor (CS, LR) shown in bold.

For CANE, we set the dimensionality of the node embeddings to 200, as in the work of Tu et al. (2017). We also tried 30-dimensional node embeddings, as in NODE2VEC and our NE methods, but performance deteriorated significantly.

4.5 Link Prediction Results

Link prediction results for the IS-A and PART-OF networks are reported in Tables 2 and 3. All content-oriented NE methods (CANE and our extensions of NODE2VEC) clearly outperform the structure-oriented method (NODE2VEC) on both datasets in both negative edge sampling settings, showing that modeling the textual descriptors of the nodes is critical. Furthermore, all methods perform much worse with Close Proximity Negative Sampling, confirming that the latter produces more difficult link prediction datasets.

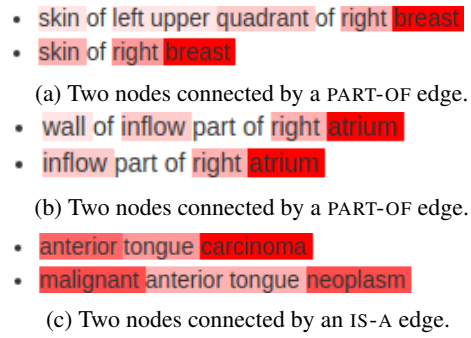
All of our NE methods (content-aware extensions of NODE2VEC) outperform NODE2VEC and CANE in every case, especially with Close Proximity Negative Sampling. We conclude that it is important to model not just the textual descriptor of a node or its direct neighbors, but as much non-local network structure as possible.

For PART-OF relations (Table 3), BIGRU-MAX-RES-N2V obtains the best results with both link predictors (CS, LR) in both negative sampling settings, but the differences from GRU-N2V are very small in most cases. For IS-A (Table 2), BIGRU-MAX-RES-N2V obtains the best results with the LR predictor, and only slightly inferior results than GRU-N2V with the CS predictor. The differences of these two NE methods from AVG-N2V are larger, indicating that recurrent neural encoders of textual descriptors are more effective than simply averaging the word embeddings of the descriptors.

Target Node: Left Eyeball (PART-OF)		
Most Similar Embeddings	**Cos**	**Hops**
equator of left eyeball	99.3	1
episcleral layer of left eyeball	99.2	4
cavity of left eyeball	99.1	1
wall of left eyeball	99.0	1
vascular layer of left eyeball	98.9	1
Target Node: Lung Carcinoma (IS-A)		
Most Similar Embeddings	**Cos**	**Hops**
recurrent lung carcinoma	97.6	1
papillary carcinoma	97.1	2
lung pleomorphic carcinoma	97.0	3
ureter carcinoma	96.6	2
lymphoepithelioma-like lung carcinoma	96.6	3

Table 4: Examples of nodes whose embeddings are closest (cosine similarity, Cos) to the embedding of a target node in the PART-OF (top) and IS-A (bottom) datasets. We also show the distances (number of edges, Hops) between the nodes in the networks.

Finally, we observe that the best results of the LR predictor are better than those of the CS predictor, in both datasets and with both negative edge sampling approaches, with the differences being larger with Close Proximity Sampling. This is as one would expect, because the logistic regression classifier can assign different weights to the dimensions of the node embeddings, depending on their predictive power, whereas cosine similarity assigns the same importance to all dimensions.



(a) Two nodes connected by a PART-OF edge.

(b) Two nodes connected by a PART-OF edge.

(c) Two nodes connected by an IS-A edge.

Figure 4: Visualization of the importance that BIGRU-MAX-RES-N2V assigns to the words of the descriptors of the nodes of three edges. Edges (a) and (b) are from the PART-OF dataset. Edge (c) is from the IS-A dataset.

4.6 Qualitative Analysis

To better understand the benefits of leveraging both network structure and textual descriptors, we present examples from the two datasets.

Most similar embeddings: Table 4 presents the five nearest nodes for two target nodes ('Left Eyeball' and 'Lung Carcinoma'), based on the cosine similarity of the corresponding node embeddings in the PART-OF and IS-A networks, respectively. We observe that all nodes in the PART-OF example are very similar content-wise to our target node. Furthermore, the model captures the semantic relationship between concepts, since most of the returned nodes are actually parts of 'Left Eyeball'. The same pattern is observed in the IS-A example, with the exception of 'ureter carcinoma', which is not directly related with 'lung carcinoma', but is still a form of cancer. Finally, it is clear that the model extracts meaningful information from both the textual content of each node and the network structure, since the returned nodes are closely located in the network (Hops 1–4).

Heatmap visualization: BIGRU-MAX-RES-N2V can be extended to highlight the words in each textual descriptor that mostly influence the corresponding node embedding. Recall that this NE method applies a max-pooling operator (Fig. 3) over the state vectors $h_1, \ldots, h_n$ of the words $w_1, \ldots, w_n$ of the descriptor, keeping the maximum value per dimension across the state vectors. We count how many dimension-values the max-pooling operator keeps from each state vector h_i, and we treat that count (normalized to $[0, 1]$) as the importance score of the corresponding word w_i.[6] We then visualize the importance scores as heatmaps of the descriptors. In the first two example edges of Fig. 4, the highest importance scores are assigned to words indicating body parts, which is appropriate given that the edges indicate PART-OF relations. In the third example edge, the highest importance score of the first descriptor is assigned to 'carcinoma', and the highest importance scores of the second descriptor are shared by 'malignant' and 'neoplasm'; again, this is appropriate, since these words indicate an IS-A relation.

[6] We actually obtain two importance scores for each word in the descriptor of a node, since each node v has two embeddings $f(v), f'(v)$, used when v is the focus or a neighbor (Section 3), i.e., there are two results of the max-pooling operator. We average the two importance scores of each word.

Edges/Descriptors	BN2V	CANE	N2V	Hops
(a) bariatric surgery (b) bypass gastrojejunostomy	82.7	38.0	56.2	11
(a) anatomical line (b) anterior malleolar fold	82.3	29.0	50.0	22
(a) zone of biceps brachii (b) short head of biceps brachii	93.0	70.0	61.6	13

Table 5: Examples of true positive edges, showing how structure and textual descriptors affect node embeddings. The first two edges are IS-A, the third one is PART-OF. The NE methods used are BIGRU-MAX-RES-N2V (BN2V), CANE and NODE2VEC (N2V). We report cosine similarities between node embeddings and the distances between the nodes (number of edges, Hops) in the networks after removing true positive edges.

Case Study: In Table 5, we present examples that illustrate learning from both the network structure and textual descriptors. All three edges are true positives, i.e., they were initially present in the network and they were removed to test link prediction. In the first two edges, which come from the IS-A network, the node descriptors share no words. Nevertheless, BIGRU-MAX-RES-N2V (BN2V) produces node embeddings with high cosine similarities, much higher than NODE2VEC that uses only network structure, presumably because the word embeddings (and neural encoder) of BN2V correctly capture lexical relations (e.g., near-synonyms). Although CANE also considers the textual descriptors, its similarity scores are much lower, presumably because it uses only local neighborhoods (single-edge hops). The nodes in the third example, which come from the PART-OF network, have a larger word overlap. NODE2VEC

is unaware of this overlap and produces the lowest score. The two content-oriented methods (BN2V, CANE) produce higher scores, but again BN2V produces a much higher similarity, presumably because it uses larger neighborhoods. In all three edges, the two nodes are distant (>10 hops), yet BN2V produces high similarity scores.

5 Conclusions and Future Work

We proposed a new method to learn content-aware node embeddings, which extends NODE2VEC by considering the textual descriptors of the nodes. The proposed approach leverages the strengths of both structure- and content-oriented node embedding methods. It exploits non-local network neighborhoods generated by random walks, as in the original NODE2VEC, and allows integrating various neural encoders of the textual descriptors. We evaluated our models on two biomedical networks extracted from UMLS, which consist of PART-OF and IS-A edges. Experimental results with two link predictors, cosine similarity and logistic regression, demonstrated that our approach is effective and outperforms previous methods which rely on structure alone, or model content along with local network context only.

In future work, we plan to experiment with networks extracted from other biomedical ontologies and knowledge bases. We also plan to explore if the word embeddings that our methods generate can improve biomedical question answering systems (McDonald et al., 2018).

Acknowledgements

This work was partly supported by the Research Center of the Athens University of Economics and Business. The work was also supported by the French National Research Agency under project ANR-16-CE33-0013.

References

Monica Agrawal, Marinka Zitnik, and Jure Leskovec. 2018. Large-scale analysis of disease pathways in the human interactome. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 23:111–122.

Nesreen K. Ahmed, Ryan A. Rossi, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, Rong Zhou, and Hoda Eldardiry. 2018. Learning role-based graph embeddings. *CoRR*, abs/1802.02896.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

James Bergstra, Daniel L K Yamins, and David D. Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms.

Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(Database-Issue):267–270.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 160–167.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Chengwei Lei and Jianhua Ruan. 2013. A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity. *Bioinformatics*, 29(3):355–364.

Linyuan Lu and Tao Zhou. 2010. Link prediction in complex networks: A survey. *CoRR*, abs/1010.0725.

Ryan McDonald, Georgios-Ioannis Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1849–1860, Brussels, Belgium.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2019. Abusive language detection with graph convolutional networks. *CoRR*, abs/1904.04073.

Christopher J Mungall. 2004. Obol: Integrating language and meaning in bio-ontologies. *Comparative and Functional Genomics*, 5:509 – 520.

Philip V. Ogren, K. Bretonnel Cohen, George K. Acquaah-Mensah, Jens Eberlein, and Lawrence E Hunter. 2003. The compositional structure of gene ontology terms. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 214–25.

Philip V. Ogren, K. Bretonnel Cohen, and Lawrence E Hunter. 2004. Implications of compositionality in the gene ontology for its curation and usage. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 174–85.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jacob VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: online learning of social representations. In *KDD*, pages 701–710. ACM.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine*, 29(3):93–106.

Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Improved semantic-aware network embedding with fine-grained word alignment. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1829–1838.

Ali Shojaie. 2013. Link prediction in biological networks using multi-mode exponential random graph models.

Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A general framework for content-enhanced network representation learning. *CoRR*, abs/1610.02906.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1067–1077.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. CANE: context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1722–1731.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1225–1234.

Peng Wang, Baowen Xu, Yurong Wu, and Xiaoyu Zhou. 2015. Link prediction in social networks: the state-of-the-art. *SCIENCE CHINA Information Sciences*, 58(1):1–38.

Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C. Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. 2018. Learning deep network representations with adversarially regularized autoencoders. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2663–2671.

Appendix

A CANE Hyper-parameters

CANE has 3 hyper-parameters, denoted α, β, γ, which control to what extent it uses information from network structure or textual descriptors. We learned these hyper-parameters by employing the HyperOpt (Bergstra et al., 2013) on the validation set.[7] All three hyper-parameters had the same search space: $[0.2, 1, 0]$ with a step of 0.1. The optimization ran for 30 trials for both datasets. Table 6 reports the resulting hyper-parameter values.

Parameters	**PART-OF**	**IS-A**
α	0.2	0.7
β	1.0	0.7
γ	1.0	0.7

Table 6: Hyper-parameter values used in CANE.

[7] For more information on HyperOpt see: https://github.com/hyperopt/hyperopt/wiki/FMin. For a tutorial see: https://github.com/Vooban/Hyperopt-Keras-CNN-CIFAR-100.