ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Master Thesis
in
Computer Science

Neural Graph Representations and their Application to Link Prediction

Sotiris Kotitsas

Committee: Ion Androutsopoulos (Supervisor)

Dimitris Pappas (Supervisor)

Iordanis Koutsopoulos

Haris Papageorgiou

November 2020

Acknowledgements

I would like to sincerely thank my supervisor Ion Androutsopoulos for the opportunity he gave me to work in this interesting field, his support and his valuable advice. I would also like to express my heartfelt thanks to my second supervisor, Dimitris Pappas for the guidance he offered me as well as the time he invested and his positive energy. In addition, I would also like to thank Causaly for their valuable advice and the data they provided. Finally, a big thanks goes to my family for their support and especially, my friends for always being there and believing in me.

Abstract

In this thesis, we experiment with the task of Link Prediction using Network Embedding (NE) methods. NE methods map network nodes to low-dimensional feature vectors and have wide applications in network analysis and bioinformatics. We consider separately the task of Link Prediction in graphs with only one type of relationship and in graphs with more than one type of relationship. The ultimate goal is to create methods capable of making novel predictions and helping in the Biomedical domain, e.g. COVID-19 related predictions. To that end, we create a biomedical dataset containing Coronavirus related information complemented by entities and relationships acquired from the UMLS ontology. Secondly, we note that the NE methods can be categorized to methods that utilize only the structure of the graphs and to methods that also try to exploit metadata associated with graphs, e.g. textual descriptors of the nodes. We utilize the idea of incorporating textual with structural information and propose several novel architectures which try to tackle the problem of simple and multi-relational link prediction. We evaluate these approaches to several benchmark datasets and also show that our multi-relational methods are competitive against the state-of-the-art in two benchmark datasets. We also show that our approach yields the same results and even outperforms the state-of-the-art in some metrics in our COVID-related graph. Finally, we do predictions regarding the COVID-19 concept and try to show their novelty, by examing if we are discovering information that had not been published when the COVID-related graph was constructed.

Περίληψη

Στην παρούσα διπλωματική εργασία, πειραματιζόμαστε με το πρόβλημα της πρόβλεψης ακμών (link prediction) σε γράφους, χρησιμοποιώντας μεθόδους που παράγουν ενσωματώσεις δικτύων (network embeddings, NE). Οι NE μέθοδοι αντιστοιχούν τους κόμβους ενός γράφου σε χαμηλών διαστάσεων αναπαραστάσεις και έχουν αρκετές εφαρμογές σε ανάλυση δικτύων και στη βιοπληροφορική. Εξετάζουμε ξεχωριστά το πρόβλημα της πρόβλεψης ακμών σε γράφους, οι οποίοι έχουν μόνο ένα είδος σχέσης και σε γράφους οι οποίοι έχουν παραπάνω από ένα τύπο σχέσεων. Ο απώτερος σκοπός είναι να δημιουργήσουμε μεθόδους οι οποίες είναι ικανές να κάνουν πρωτοποριακές προβλέψεις και να βοηθήσουν στο βιοιατρικό τομέα, π.χ. προβλέψεις σχετικές με τον COVID-19. Γι' αυτό το σκοπό, δημιουργούμε ένα βιοιατρικό σύνολο δεδομένων, το οποίο περιέχει πληροφορίες σχετικές με κορωνοϊούς και έχει συμπληρωματική πληροφορία για οντότητες και σχέσεις από την οντολογία UMLS. Δεύτερον, χωρίζουμε τις μεθόδους NE, σε μεθόδους οι οποίες αξιοποιούν μόνο τη δομή του γράφου και σε μεθόδους που προσπαθούν να εκμεταλλευτούν και μεταδεδομένα του γράφου, π.χ. κειμενικές περιγραφές των κόμβων. Αξιοποιούμε την ιδέα του να χρησιμοποιούμε πληροφορία σχετική με την δομή και το κείμενο του γράφου και προτείνουμε αρκετές πρωτοποριακές μεθόδους που προσπαθούν να λύσουν το πρόβλημα της πρόβλεψης ακμών με έναν και πολλούς τύπους σχέσεων. Αξιολογούμε αυτές τις προσεγγίσεις σε πολλαπλά σύνολα δεδομένων και επίσης δείχνουμε ότι οι πολυ-σχεσικές μέθοδοί μας είναι ανταγωνιστικές με μια από τις κορυφαίες (state of the art) μεθόδους σε δυο σύνολα δεδομένων. Επίσης, δείχνουμε ότι η προσέγγισή μας αποδίδει τα ίδια αποτελέσματα και ακόμα ξεπερνάει την κορυφαία μέθοδο σε ορισμένες μετρικές στον COVID γράφο μας. Τέλος, κάνουμε προβλέψεις σχετικές με την οντότητα του COVID-19 και προσπαθούμε να αξιολογήσουμε την χρησιμότητά τους, εξετάζοντας αν οι πληροφορίες που ανακαλύπτουμε είχαν δημοσιευτεί την περίοδο που φτιάχτηκε ο COVID γράφος μας.

Contents

1 Introduction

Knowledge extracted from graphs (networks) is used as input to a variety of applications. As a result, graph analysis has attracted a lot of research interest in the past few years. One typical application is link prediction (Lu and Zhou, 2010). Link prediction can facilitate knowledge discovery and it is the task of predicting missing links or links that are likely to occur in the future. It is particularly evident in the biomedical domain, where it can be used to predict protein-protein interactions (PPIs) (Grover and Leskovec, 2016) and also polypharmacy side effects (Zitnik et al., 2018), that are unwanted and unknown interactions. Furthermore, link prediction can be utilized for Drug-Drug Interactions (DDIs) where traditional work relies on in vitro and in vivo experiments and has laboratory limitations (Karim et al., 2019) and Drug-Target Interactions (DTIs) where the graph represents drugs/chemicals and the proteins which they interact with (Y. Wang and Zeng, 2013).

Past approaches on the above problem include models that operate on the original adjacency matrix, like similarity based (Adamic and Adar, 2001) and probabilistic models (Friedman et al., 1999). Working with these kind of methods is difficult, so recent approaches attempt to represent the graphs based on a learned vector representation (embedding) (Perozzi et al., 2014; Grover and Leskovec, 2016; J.Tang et al., 2015). This vector representation can represent the entire graph or each individual node (node embedding). In this paper, we focus on the latter. Existing work includes DeepWalk (Perozzi et al., 2014), Node2Vec (Grover and Leskovec, 2016), Graph Convolution Networks (GCNs) (Kipf and Welling, 2017), SDNE (D. Wang et al., 2016), NETRA (Yu et al., 2018). These frameworks learn node embeddings exploiting the network structure and try to capture higher-order proximities.

1.1 Motivation and Contribution

However, Knowledge Graphs are associated with metadata (textual descriptors, images, definitions etc). Our motivation is to utilize this additional information to create more informative node representations that can predict links that would be helpful to the biomedical domain. Methods focused in domains other than the biomedical one, like CANE (Tu et al., 2017), WANE (Shen et al., 2018), CENE (Sun et al., 2016) have already been proposed but they consider local neighborhoods only. Content-Aware Node2Vec (Kotitsas et al., 2019), on the other hand, tries to embed nodes in biomedical datasets exploiting the

wider network contexts created by Node2Vec, together with textual descriptors associated with each node.

Our contributions include the extension of Kotitsas et al., 2019 by addressing its most important limitations and improving its performance. Despite the improved results, that particular model cannot be used in heterogeneous graphs. Most graphs that are used for doing DDIs, DTIs, PPIs come in the form of heterogeneous graphs, since they contain different types of nodes and relationships. One advantage of Kotitsas et al., 2019 is that the model is agnostic to the node embedding framework. So to apply this idea to heterogeneous graphs, one could simply remove the NODE2VEC framework and replace it with a framework that can be applied to heterogeneous graphs. As a result, the main contribution of this thesis is the application of neural text encoders with Graph Convolutional Network (GCN) and Graph Attention Network (GAT) layers and their impact on drawing useful conclusions about biomedical link prediction.

1.2 Thesis Structure

The rest of the thesis is organized as follows:

- Chapter 2 describes the background theory that the methods in this thesis use.
- Chapter 3 describes the datasets.
- Chapter 4 describes the methods used in this thesis that exploit only the structure of the graph.
- Chapter 5 describes the methods used in this thesis that exploit the structure and the textual descriptors associated with the nodes of the graph.
- Chapter 6 describes a COVID related analysis, where we discuss the ability of our model to predict links associated with COVID-19.
- Chapter 7 discusses related work.
- Chapter 8 concludes and proposes directions for future work.

2 Background

In this chapter, we provide a brief introduction to background topics that our methods are based upon. We will discuss some of the theory behind node embedding models, such as random walk and graph convolution models.

2.1 Spatial-Based Models

We can view node embedding (NE) methods as learning the aggregation rule between first order neighbors. In other words, these methods focus on learning the weights of the nodes and their neighbors, in order to reveal and utilize structural properties of the graph (Chen et al., 2020). More formally, we can write this process as an update function, where we can get a lookup table for the node embeddings, called H. The node embedding of node u_i can be obtained through $H(u_i)$:

$$H(u_i) = W^{(1)}h(u_i) + \sum_{u_j \in N(u_i)} W^{(2)}h(u_j) \tag{2.1}$$

where u_j is a neighbor of node u_i, $h(\cdot)$ is their current node embedding and $W^{(1)}, W^{(2)}$ are weight matrices. $W^{(1)}$ (self-loop weights) is used to help take into account the node embedding of u_i. The first product on the right hand side of the equation denotes the hidden node representation of node u_i and the second product under the sum, denotes the hidden representations of the neighbors of node u_i. The methods generalize better if we apply a degree normalization to eq. 2.1 (Kipf and Welling, 2017; Johnson and Zhang, 2007). We can apply two normalization techniques. One denoted as random walk normalization and the other as symmetric normalization. For each normalization, eq. 2.1 becomes:

$$H(u_i) = W^{(1)}h(u_i) + \sum_{u_j \in N(u_i)} W^{(2)}\frac{h(u_j)}{d_{u_i}}, \tag{2.2}$$

$$H(u_i) = W^{(1)}h(u_i) + \sum_{u_j \in N(u_i)} W^{(2)}\frac{h(u_j)}{\sqrt{d_{u_i}d_{u_j}}} \tag{2.3}$$

where d_{u_i} is the degree of some node u_i, eq. 2.2 is for the random walk normalization and eq. 2.3 for the symmetric one. We can rewrite the above functions in matrix form as follows (Chen et al., 2020):

$$H = W^{(1)}H + W^{(2)}D^{-1}AH = (W^{(1)}I + W^{(2)}D^{-1}A)H, \tag{2.4}$$

$$H = W^{(1)}H + W^{(2)}D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H = (W^{(1)}I + W^{(2)}D^{-\frac{1}{2}}AD^{-\frac{1}{2}})H \tag{2.5}$$

where D is the degree matrix, I is the identity matrix and A is the adjacency matrix. We can replace $D^{-1}A$ and $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ with $\hat{A}$ and we get the generalized function of updating the weights:

$$H = (W^{(1)}I + W^{(2)}\hat{A})H \tag{2.6}$$

The main difference between the NE methods that we investigate resides on their approach of using matrix $\hat{A}$.

2.2 Random Walk Models

According to the previous section, the adjacency matrix A can help utilize the first order neighbors of a node i. The motivation behind random walk NE methods is to collect richer graph context, by capturing higher order dependencies between the nodes, since first order neighbors may not always be sufficient. For that purpose, DEEPWALK (Perozzi et al., 2014) was proposed. DEEPWALK first generates a fixed number of random walks and then applies the SKIPGRAM (Mikolov et al., 2013) model to extract node features. DEEPWALK follows eq.2.6 with $\hat{A} = D^{-1}A$ being the transition probabilities from one node to another during the random walk exploration. It can capture a k-th-order neighborhood by setting $\hat{A} = (I + \hat{A} + \hat{A}^2 + \cdots + \hat{A}^k)$, where $\hat{A}^k$ represents all the paths of length k from a node i to a node j.

However, Li et al., 2018 argue that if a method utilizes a large order of neighbors, it can lead to aggregating all the node embeddings, resulting in over-smoothing the embeddings and losing sight of the neighborhoods. So NODE2VEC tries to control the balance between low-order and high-order neighborhoods py proposing two variables p and q, which let the random walks switch between Breadth-First search (BFS) and Depth-First search DFS exploration (more on that in chapter 4).

2.3 Graph Convolutional Networks

Graph Convolutional Networks (GCNs) will be discussed in-depth later on in this thesis (chapters 4, 5). In general, GCNs are a simplification of ChebNet (Defferrard et al., 2016). ChebNet includes a neural network layer for the convolutional operator (Chen et al., 2020). GCNs use eq. 2.6, but first add self-loops to nodes. By adding self-loops D becomes $\hat{D} = \sum_j (A + I)_{ij}$ and the weight matrix $W^{(1)}$ of eq. 2.6 is no longer needed. We can rewrite eq. 2.6 as:

$$H = W^{(2)} \hat{A} H \tag{2.7}$$

where now $\hat{A}$ is equal to $\hat{A} = \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$. The output of a GCN layer according to eq. 2.7 is the sum of the node embedding of the current node and the average of its neighbors. Later on, we will see that we can also add information based on the relationship of the current node and its neighbors.

3 Datasets

In this chapter we will describe the graphs (datasets) used for our experiments. Graphs come in many forms. Some are undirected and unweighted, others are directed with weights. Some graphs contain nodes and edges of only one type and other contain nodes and edges of different types (Heterogeneous Graphs). In the present work we will conduct experiments with various graphs, based on the models of each respective chapter. Simpler models are tested on undirected and unweighted graphs and more complex models are tested on heterogeneous graphs, where we can represent more complex information and interactions, making these models more easily applicable to real-world problems.

3.1 UMLS Datasets

Our starting point, regarding our graphs, are two biomedical datasets extracted from UMLS (Bodenreider, 2004). These two datasets were created during our previous work (Kotitsas et al., 2019). The UMLS ontology contains approx. 3.8 million biomedical concepts and 54 semantic relationships. The relationships become edges in the graphs, and the concepts become nodes. Each concept (node) is associated with a textual descriptor. We extract two types of semantic relationships, creating two graphs. The first, and smaller one, consists of PART-OF relationships where each node represents a part of the human body. The second graph contains IS-A relationships, and the concepts represented by the nodes vary across the spectrum of biomedical entities (diseases, proteins, genes,etc.). Both of these graphs are undirected, unweighted and contain nodes and edges of only one type. Statistics for the two datasets are shown in Table 3.1 and examples of instances from the datasets are shown in Table 3.2.

3.2 Benchmark MultiRelational Datasets

Moving away from simple graphs and onto more complex graphs and models, we use two standard benchmark datasets in multirelational link prediction.

The first benchmark dataset we used is derived from Wordnet (Glorot et al., 2013). Wordnet is a knowledge base designed to be used as a machine- and human-readable dictionary, thesaurus and to support text analysis. It incorporates its knowledge into its graph structure.

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
Min. descriptor length	1 word	1 word

Tab. 3.1: Statistics of the two datasets (IS-A, PART-OF). The true positive and true negative edges are used in the link prediction experiments.

Examples of Edges
Ectoplasm **IS-A** Subdivision of Cytoplasm
Dental Pulp **PART-OF** Tooth

Tab. 3.2: Examples of edges of the two datasets (IS-A, PART-OF).

The entities correspond to nodes (also called synsets) and they represent senses. The edges represent lexical relations between synsets, through different relation types. The dataset was created by considering the entities that were connected with 18 particular relation types (e.g. hypernym, hyponym etc). The original graph contained 40.943 nodes along with 18 relation types. Wordnet includes words with different meanings. In order to avoid ambiguity in the dataset the entities are described by their synset name, their part-of-speech tag ('NN' for noun, 'VB' for verb, 'JJ' for adjective and 'RB' for adverb) and an index indicating which sense it refers to.

However, the original version of this dataset is not suitable for research, since it suffers from test set leakage. A lot of training instances (source node, relation type, target node) exist as inverse relationships in the test set and a simple rule-based model can achieve state-of-the-art results. A new version of the dataset WN18RR (Dettmers et al., 2018) was created, were all of these inverse relationships were removed and this is the dataset used on this thesis.

The second multirelational link prediction benchmark dataset used in this thesis is FB15k (Bordes et al., 2013) which is a subset of Freebase and contains about 14.951 nodes and 1,345 different relationship types. Unlike WN18RR, content in this graph describes facts about movies, actors, awards, sports, and sport teams. However, the original dataset suffered from the same problem as WN, where the inverse relationships of the train set existed in the test set and simple models could achieve great results. FB15K-237 (Toutanova et al., 2015) was proposed where these types of edges where removed from the splits and also the new dataset has a reduced number of relationships (237 in total).

Statistics	WN18RR	FB15K-237
Nodes	40,943	14,541
Total Edges	93,003	310,116
Training Edges	86,335	272,115
Validation Edges	3034	17,535
Test Edges	3134	20,466
Avg. descriptor length	1 words	3 words
Max. descriptor length	8 words	17 words
Min. descriptor length	1 word	1 word

Tab. 3.3: Statistics of the two datasets (WN18RR, FB15K-237).

WN18RR	FB15K-237
clupea **_hypernym** fish genus	dominican republic **/location/country/form_of_government** republic
evening **_has_part** sunset	wendee lee **/people/person/profession** voice actor
nyctaginaceae **_member_meronym** allionia	american history x **/film /film /genre** action film

Tab. 3.4: Examples of edges of the two datasets (WN18RR, FB15K-237).

The complete statistics of these benchmark datasets can be seen in Table 3.3. Examples of the two datasets can be seen in Table 3.4.

3.3 Causaly Dataset

To investigate the effectiveness of models that operate on heterogeneous graphs, we construct a graph that incorporates knowledge for the Coronavirus diseases and supplementary information extracted from the UMLS. In this sub chapter we will analyze how we constructed this particular dataset and discuss how it can be used to discover knowledge for the Coronaviridae family (and particularly covid-19)which is a matter of great importance.

First of all we need to acquire information in the aforementioned form and also we need this information to be about the Coronaviridae family. That data was provided by Causaly[1]. Our whole graph was constructed based on the initial graph provided by Causaly. The graph contains relationships of 4 types (UNIDIRECTIONAL, BIDIRECTIONAL, UPREGULATE and DOWNREGULATE). These 4 relations indicate a reaction from some target, compound, other disease to a coronavirus disease. The initial graph containing data relevant to coronaviridae family has 1878 nodes and 2587 edges. For every edge (source - relationship - target) Causaly has found evidence indicating the type of the relationship. Some pairs

[1]https://www.causaly.com/

of source - target interact with more than one of the four relation types. For that reason Causaly aggregates this information by keeping the most probable relationship type based on the number of evidence for that particular type. For our experiments we use the aggregated version of the graph. So far, we have no additional information for the entities that exist in our initial graph. For example, for a certain entity like cyclosporine, we are missing the fact that cyclosporine is an immunotherapeutic agent or that cyclosporine is an enzyme inhibitor. To enrich our graph with this type of information we extract information from the Unified Medical Language System (UMLS) ontology. For every entity in our initial graph we map it to the corresponding entity in UMLS and then through the MRREL file of UMLS we get the entities at distance 2 (hops) from the original entity. As a result, we now have in our graph edges like (cyclosporine IS-A immunotherapeutic agent and cyclosporine IS-A enzyme inhibitor).

However, we have the following use cases. We would like to be able to model interactions from compounds to targets and from targets to the diseases of our interest and also from compounds to diseases, since it could be possible that a compound directly interacts with a disease. In effect, for our first use case, we will be able to find information like for example: Compound A interacts with Target A which interacts with Disease A, so from Compound A we can find other relevant Targets that might help with Disease A. From the graph created through Causaly and UMLS, we have a very small number of edges from compounds to targets, targets to diseases and also from compounds to diseases.

To obtain edges of this type we used the CHEMBL Database. CHEMBL is a manually curated database of bioactive molecules with drug-like properties. CHEMBL contains 2 million compounds and 13 thousand targets. For each compound, CHEMBL provides drug mechanisms and drug indications. Drug mechanisms are interactions between a compound and a target. This interaction comes in the form of Mechanism Of Action (MOA). Each MOA can be mapped to one of our 4 relationship types of interest. All the MOAs that concern us, along with the mappings are presented in the Appendix in Table 3.7[2]. Furthermore, drug indications can be mapped directly to DOWNREGULATE and they model an interaction between a compound and a disease. For each entity in our graph, if it is a compound we search it in CHEMBL and if it exists we get its drug mechanisms and its drug indications. As a result, through our mapping we get edges between compound-targets and compound-diseases, which are also linked with one of our 4 relationship types.

The final graph however, contains more than the 4 aforementioned relationships since it was enriched with information through the UMLS ontology. Our training protocol will allow the models to also learn information from these extra relationship types but learn only to predict on the 4 basic ones. The complete statistics of the graph can be seen in Table 3.5 and examples of edges in Table 3.6.

[2]The mapping was done manually through manual search

Statistics	COVID GRAPH
Nodes	262,649
Edges	609,947
Training	5474
Testing	350
Validation	350
Avg. descriptor length	8 words
Max. descriptor length	641 words
Min. descriptor length	1 word

Tab. 3.5: Statistics of the dataset COVID GRAPH.

Examples of Edges
Cyclosporine **DOWNREGULATE** Genus Coronavirus
Sars Coronavirus **CAUSATIVE AGENT OF** Severe Acute Respiratory Syndrome

Tab. 3.6: Examples of edges of COVID GRAPH

Mechanisms of Action	Type of edge
RELEASING AGENT	UPREGULATE
POSITIVE ALLOSTERIC MODULATOR	UPREGULATE
BINDING AGENT	BIDIRECTIONAL
CROSS-LINKING AGENT	BIDIRECTIONAL
MODULATOR	UPREGULATE
SEQUESTERING AGENT	DOWNREGULATE
POSITIVE MODULATOR	UPREGULATE
ANTAGONIST	DOWNREGULATE
RNAI INHIBITOR	DOWNREGULATE
ACTIVATOR	UPREGULATE
OTHER	UNIDIRECTIONAL
PARTIAL AGONIST	UPREGULATE
CHELATING AGENT	DOWNREGULATE
ANTISENSE INHIBITOR	DOWNREGULATE
NEGATIVE MODULATOR	DOWNREGULATE
OPENER	UPREGULATE
AGONIST	UPREGULATE
BLOCKER	DOWNREGULATE
NEGATIVE ALLOSTERIC MODULATOR	DOWNREGULATE
INHIBITOR	DOWNREGULATE
OXIDATIVE ENZYME	BIDIRECTIONAL
HYDROLYTIC ENZYME	BIDIRECTIONAL
PROTEOLYTIC ENZYME	BIDIRECTIONAL

Tab. 3.7: Mechanisms of Actions (MOAs) and their mapping to the relationship types of interest.

4 Graph Representations explicitly using Structure

Graph representation models, also known as, Network Embedding (NE) models, try to map each node of a network to an embedding, meaning a low-dimensional feature vector. They are highly effective in network analysis tasks involving link prediction over nodes and edges (our task at hand) (Lu and Zhou, 2010) and node classification (Sen et al., 2008).

A lot of early network embedding work, like DEEPWALK (Perozzi et al., 2014), LINE (J.Tang et al., 2015), NODE2VEC (Grover and Leskovec, 2016), GCN (Kipf and Welling, 2017), leverage information from the graph structure to produce node embeddings that can reconstruct node neighborhoods. The main advantage of these methods, which explicitly use the structure of the graph during learning, is that they encode the graph context of the nodes, which can be very informative.

In this chapter, we will discuss methods that work as mentioned above and we will evaluate them on certain datasets that were discussed in Chapter 3. The methods discussed here constitute the basis of the methods we will discuss in later chapters.

4.1 Node2Vec

In this section we will discuss NODE2VEC (Grover and Leskovec, 2016). NODE2VEC is a NE that tries to learn node embeddings only by the structure of the graph. NODE2VEC is based on DEEPWALK (Perozzi et al., 2014). DEEPWALK learns node embeddings by applying WORD2VEC's SKIPGRAM (Mikolov et al., 2013) model to node sequences generated via random walks on the network. NODE2VEC adopts the same learning strategy, but also explores different strategies to perform random walks, introducing hyper-parameters to guide them and generate more flexible neighborhoods. More formally NODE2VEC introduces return parameters p and q.

Parameter p controls the likelihood of immediately revisiting a node in the random walk. If we set p to a high value then we are less likely to sample an already visited node in the next 2 hops (unless the node that we end up to has no other neighbors and we must revisit the previous node). By controlling p, we can encourage the random walks to perform a

moderate exploration of the graph structure. If we set parameter p to a low value then we encourage the random walks to stay always close to the previous visited node.

Parameter q can allow us to interchange between "inwards" and "outwards" exploration of the random walks. If we set $q > 1$ then the random walk is biased towards the initial node of the random walk. Such behavior approximates BFS-like exploration keeping the random walks close to the starting node. On the other hand, if we set $q < 1$ then the random walk is more biased to visit nodes further away from its starting point. This behavior leads to DFS-like exploration since we encourage the random walks to move "onwards" visiting a larger context of the graph structure.

Regarding its training protocol, as it was mentioned above NODE2VEC applies WORD2VEC's SKIPGRAM (Mikolov et al., 2013) model to learn node embeddings. In essence to incorporate structural information of the graph into the node embeddings, SKIPGRAM maximizes 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. Through NODE2VEC we 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 ($v_{i,j} \in V$). 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):

$$p(u|v) = \frac{\exp(f'(u) \cdot f(v))}{\sum_{u' \in V} \exp(f'(u') \cdot f(v))} \tag{4.1}$$

where it is assumed that each node v has two different node embeddings, $f(v), f'(v)$, used when v is the focus node or the predicted neighbor, respectively, and $\cdot$ denotes the dot product. NODE2VEC minimizes the following objective function:

$$L = -\sum_{v \in V} \sum_{i=1}^{r} \sum_{j=2}^{k} \log p(v_{i,j} | v_{i,1} = v) \tag{4.2}$$

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. 4.1) is computationally inefficient. We apply negative sampling instead, as in WORD2VEC (Mikolov et al., 2013). Thus, NODE2VEC is analogous to SKIPGRAM WORD2VEC, but using random walks from each focus node, instead of using a context window around each focus word in a corpus.

4.2 Graph Convolutional Networks

In this section we will discuss Graph Convolutional Networks (GCN) (Kipf and Welling, 2017). GCNs have been shown to be effective at aggregating and encoding features from network neighborhoods, and has led to significant improvements in areas such as semi-supervised classification (Kipf and Welling, 2017) and multi-relational link prediction (Zitnik et al., 2018; Schlichtkrull et al., 2018).

Graph Convolutional Networks, aggregate the embeddings of the neighbors of a node and then apply a propagation function on the obtained embedding. Aggregating the embeddings of only local-neighborhoods, makes GCNs scalable and fast for large graphs. By applying multiple aggregations for the neighborhood of a node, we allows the learned node embedding to characterize a global neighborhood (Goyal and Ferrara, 2018). Every GCN layer in its simplest form can be written as

$$h_i^{l+1} = \sigma(W_0^{(l)} h_i^l + \sum_{j \in N_i} c_{i,j} W_1^{(l)} h_j^l) \tag{4.3}$$

where σ is a non-linear activation function such as the ReLU. W_0^l and W_1^l are learnable parameter matrices of dimensions $d^{l+1} \times d^l$, with d^{l+1} being the output dimensions of layer l and d^l the input dimensions of layer l. W_0^l can be considered as a weight matrix for selfloops and W_1^l a weight matrix for the set of neighbors of i, the set N_i. $c_{i,j}$ is a normalization constant typically chosen to be $c_{i,j} = \frac{1}{\sqrt{D_{i,i} D_{j,j}}}$, where $D_{i,i}$ the degree of node i. This constant originates from the symmetric normalization of the adjacency matrix, which helps the GCN layers to account for the degree of each node and in effect avoid exploding numbers in the embeddings of nodes with large degrees.

The former equation can be written in a matrix form as follows

$$H^{l+1} = \sigma(H^l W_0^l + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^l W_1^l), \tag{4.4}$$

where the sum over the neighbors of node i, along with the normalization constant $c_{i,j}$, has been replaced by the multiplication of the normalized adjacency matrix $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ as mentioned on the previous paragraph. Furthermore, H^l is the output of the l-th layer of the model and $H^0 = X$, with $X \in \mathbb{R}^{N \times d}$ being the node features. In this version of GCN the node features can be randomly initialized embeddings or the average of word embeddings of some textual descriptor associated with the node. In later chapters, we will see the application of RNN encoders under the GCN layers, in order to feed the GCN with more informative node features and also learn word embeddings as a side effect.

In the above defined model, we used a seperate weight matrix for modeling the selfloops. According to Kipf and Welling, 2017 this can lead to overfitting in nodes with very few

neighbors and when the graph has very few nodes in general. To address this issue Kipf and Welling, 2017 proposed to use a model with one single parameter weight matrix. As a result, we add selfloops on the adjacency matrix by adding the identity matrix I_N. The normalization of the adjacency matrix can be written as $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$, where $\hat{A} = A + I_N$ and $\hat{D}_{ii} = \sum_{j \in N_i} \hat{A}_{i,j}$. Equation 4.4 becomes

$$H^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W_1^l), \tag{4.5}$$

4.3 Experimental Results

In this section we will present and discuss results regarding the two methods described in the previous sections. We test these two methods on the task of link prediction, which is the primary task of this thesis.

The aforementioned methods, namely NODE2VEC and GCN, cannot perform multi-relational prediction. So, we evaluate them on simple link prediction, where all the nodes of the graph have a single relationship among them. The two datasets of this thesis that have only one type of relationship between nodes, are the PART-OF and IS-A datasets of section 3.1. To test our models on these two datasets, first we need to remove a fraction of the edges. Then given an incomplete network, we need to infer these missing links. Since, we are working with a random walk method (NODE2VEC), we need to ensure that during the removal of the edges (which we will try to predict), the graph remains conncected, so we can perform random walks over it. Each removed edge, connecting two nodes u_1, u_2 is treated as a true positive, meaning that our methods should infer that these two nodes should be connected. However, we also need a set of true negatives, which are pairs of nodes $u_1^{'}, u_2^{'}$, that are not connected in the original graph and we should infer that these two nodes should not be connected.

We devise two approaches of generating true negative samples for testing our methods. In Random Negative Sampling, we simply choose two random nodes and ensure that are not connected with an edge on the original graph. The other approach is Close Proximity Sampling, where we consider each node of the graph as a focus node and we iterate over them. For each focus node u, we want to find another node v, in close proximity, making it harder for the link prediction method to infer that these two nodes should not be linked. However, we do not want v to be an ancestor or descendent of u, since the two datasets used in this chapter are hierarchies and someone could argue that if u is a grandparent of v then it is not wrong to predict an edge between them. Taking these limitations into consideration, we first find the ancestors of u that are between 2 to 5 hops

away from u in the original network.[1] We randomly choose one of these ancestors and as the candindate node v, we select one of the ancestor's children, making sure that u and v are not connected in the original network. The complete statistics of these two datasets, along with the statistics of the train and evaluation sets, can be seen in Section 3.1.

We evaluate our methods using two link predictors described below.

Cosine similarity link predictor (CS): We are given two nodes u_1, u_2 (they can be the end-points of either a true positive or a true negative edge). CS calculates the cosine similarity between their corresponding node embeddings. By clipping the negative cosine scores to zero, we can treat the cosine similarity as a probability score. More formally, $score(u_1, u_2) = max(0, cos(f(u_1), f(u_2)))$, where $f(u_i)$ the node embedding of node i. CS, predicts an edge between these two nodes if $score(u_1, u_2) \geq t$, where t is some threshold. To alleviate the need of choosing a theshold t beforehand, we evaluate the effectiveness of the CS predictor by computing AUC (Area under the ROC Curve), considering in effect the precision and recall for various t.

Logistic regression link predictor (LR): Same as the CS predictor, we are given two nodes u_1, u_2. However, now we compute the Hadamard (element-wise) product of the two node embeddings $f(v_1) \odot f(v_2)$ and feed it to a Logistic Regression classifier to obtain a probability estimate p the two nodes should be connected. LR will predict an edge if $p \geq t$, where t is again a threshold value. Just like in the CS predictor we compute the AUC score to evaluate with multiple thresholds. 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.

An important observation, is that NODE2VEC optimizes the SKIPGRAM objective, meaning its training is unsupervised and it cannot be trained end-to-end. So first we train the node embeddings and then we can evaluate them. This is the reason why the LR predictor when used together with NODE2VEC needs to be trained. On the other hand, when using the GCN model, we can train end-to-end by using either the CS or LR predictor. We expect the GCN model to perform better since it optimizes the node embeddings directly for the task of link prediction.

For the experiments in this section, we used 30 dimensions for the final node embeddings, for both models and the number of GCN layers used is 2. These hyper-parameters where chosen mostly to speed up the experiments. Now following Kotitsas et al., 2019 in the random walks of NODE2VEC, we set r = 5, l = 40, k = 10[2] for IS-A, and r = 10, l = 40, k = 10

[1] The edges of these two datasets are not directed. Hence, looking for descendents would be equivalent. We convert the datasets to directed to complete this process.

[2] r is the number of walks, l is the length of each walk and k is the context-size window

for PART-OF. We train the models using 250 as a maximum number of epochs and we utilize early stopping of 20 epochs, i.e., we stop the training if the AUC score on the development set does not increase for 20 consecutive epochs. The optimizer used is Adam (Kingma and Ba, 2015) with learning rate of 0.01. For the implementation of the methods, we used PyTorch (Paszke et al., 2017).

Link Prediction Results: Link prediction results for the IS-A and PART-OF networks are reported in Tables 4.1 and 4.2.

NE Method + Link Predictor	Random Negative Sampling	Close Proximity Sampling
NODE2VEC + CS	**75.0**	57.5
GCN + CS	71.0	**63.0**
NODE2VEC + LR	77.0	58.0
GCN + LR	**77.3**	**67.3**

Tab. 4.1: 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	**77.9**	**65.2**
GCN + CS	77.4	60.2
NODE2VEC + LR	**78.9**	64.6
GCN + LR	66.4	**65.2**

Tab. 4.2: AUC scores (%) for the PART-OF dataset. Best scores per link predictor (CS, LR) shown in bold.

The NODE2VEC and GCN models that were described in this chapter, function as the building blocks for the models described in the next chapter of this thesis. So it is natural to asses their performance. Additionally, the later models utilize the structure and the textual descriptors of the nodes and we can compare the benefits that the textual information provides.

In both datasets, all methods perform much worse in Close Proximity Sampling, which indicates that this type of generating true negative edges is indeed difficult. As stated before, these models only utilize structure. The nodes of these negative edges are in close proximity and having only the structure as a way to discriminate between positive and negative edges does not offer much.

In IS-A, apart from the CS predictor in Random Negative Sampling, the GCN model outperforms NODE2VEC in both negative sampling approaches, presumably because it trains end-to-end and directly for the task at hand (link prediction). However, this is not the case in the PART-OF dataset, since NODE2VEC outperforms GCN in every setting apart from the

LR predictor in Close Proximity Sampling. This anomaly presumably can be attributed to the fact that the PART-OF dataset is small and has a tree-like structure. With only a few hops the concept of the neighborhood can be shifted. GCNs aggregate all the information in the present neighborhood with the same importance. Stacking GCN layers in a small graph can lead to over-smoothing the node embeddings, losing the notion of the neighborhood, which is the case here.

5 Graph Representations that jointly use Textual and Structural Information

As mentioned in the previous chapter, network embedding (NE) methods, try to map each node to a low dimensional representation (node embedding). Early work, consisted of methods that utilize only the structure properties and the topology of the graph. The advantage is that this type of information can be very useful. However, the downside is that they typically treat each node as an atomic unit, directly mapped to an embedding in a look-up table (Fig. 5.1a). 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. Knowledge Graphs are associated with metadata (textual descriptors, images, definitions etc). More recent work, 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 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. 5.1b.

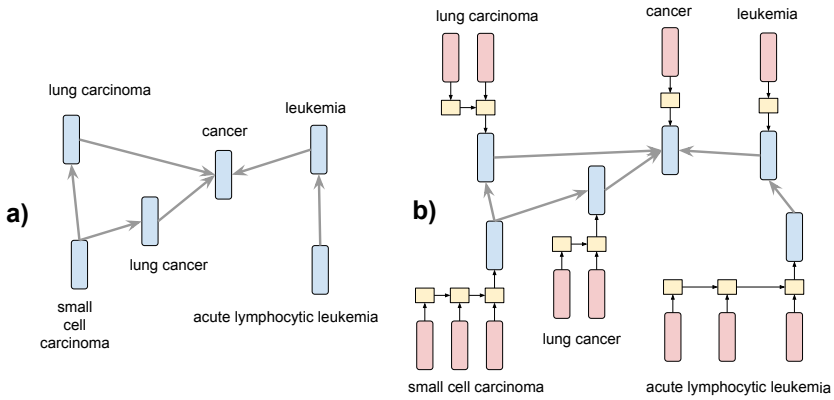

Fig. 5.1: Example network with nodes associated with textual descriptors. a) A model where each node is represented by a vector (node embedding) from a look-up table. b) A model where each node embedding is generated compositionally from the word embeddings of its descriptor via an RNN. The latter model can learn node embeddings from both the network structure and the word sequences of the textual descriptors. The figure was taken from (Kotitsas et al., 2019).

In this chapter, we will cover our previous work at Kotitsas et al., 2019, describing how the method of that particular work learns node embeddings by incorporating textual with

structural information. Furthermore, we will talk about extensions/improvements of that work. Finally, we will discuss new approaches based on GCN and GAT models and their application on more complex problems (heterogeneous graphs).

5.1 Content-Aware NODE2VEC

This section describes the method at Kotitsas et al., 2019. The motivation behind this work is that despite the efforts of previous works to learn node embeddings based on textual descriptors and structural properties, these methods considered impoverished network contexts when embedding nodes, usually single-edge hops, as opposed to the non-local structure considered by most structure-oriented methods. The importance of modeling rich neighborhood context while taking into account the textual descriptors, becomes evident if we consider the following example. Suppose we have the edge 'acute leukemia' IS-A 'leukemia'. To be able to predict this IS-A relation from the embeddings of both its corresponding nodes (and the word embeddings of their textual descriptors like in Fig. 5.1b), a NE method would only need to model the word 'acute' as a modifier of the word 'leukemia', with its only purpose to denote a sub-type of 'leukemia'. This property can be easily 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.

The above strategy would not work for an edge like 'p53' (a protein) IS-A 'tumor suppressor', where there is no overlap between the textual descriptors and not a word that would indicate 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.

Content-Aware NODE2VEC is 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. The aforementioned methodology was explained in Chapter 4. The original NODE2VEC does not account for textual descriptors associated with each node. 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, Content-Aware NODE2VEC

substitutes 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. 4.1) the actual neighbors of each focus node (Fig. 5.2). For simplicity, we only mention $f(v)$ and $T(v)$ below, but the same applies to $f'(v)$ and $T'(v)$. In the original paper Kotitsas et al., 2019 we experimented with several textual encoders, from averaging the word embeddings of the descriptors to using simple one-directional GRU cells (Cho et al., 2014). In this thesis, we will describe the final textual encoder that is used on Content-Aware NODE2VEC.

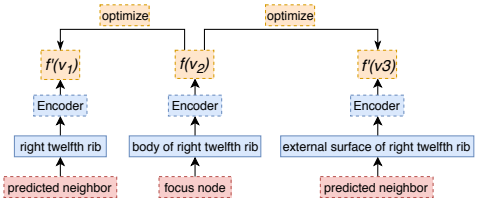

Fig. 5.2: Illustration of the proposed NE approach. Figure taken from Kotitsas et al., 2019.

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:

$$H_f = \text{GRU}_f(e(w_1), \ldots, e(w_n)) \tag{5.1}$$

$$H_b = \text{GRU}_b(e(w_1), \ldots, e(w_n)) \tag{5.2}$$

$$H = H_f + H_b \tag{5.3}$$

where $_f$, $_b$ denote the forward and backward directions, and $+$ indicates component-wise addition. We add residual connections (He et al., 2016) 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 5.3 illustrates this method. A textual encoder utilizing self-attention instead of max-pooling was also tried but without improving the results.

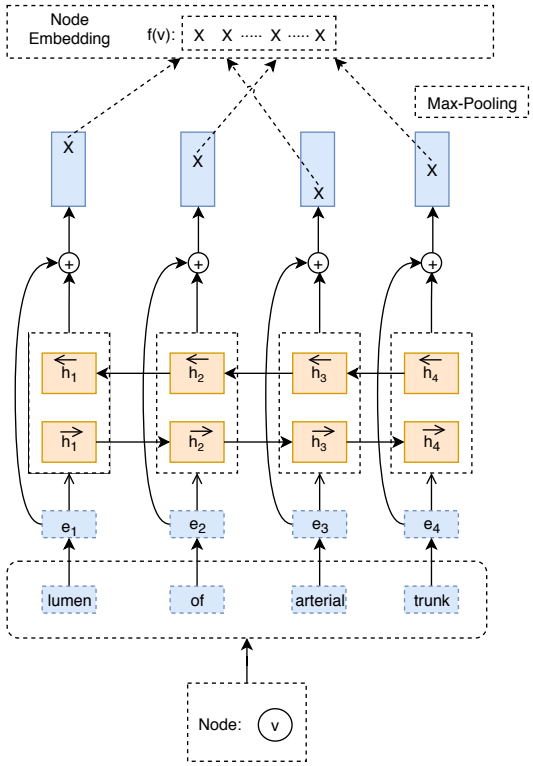

Fig. 5.3: Obtaining the embedding of a node v by applying a BIGRU encoder with max-pooling and residuals to the embeddings of v's textual descriptor. Figure taken from Kotitsas et al., 2019.

5.2 Improvements on Content-Aware NODE2VEC

Content-Aware NODE2VEC yielded very good results on two biomedical datasets (PART-OF, IS-A) described in section 3.1 and also outperforming the models compared against it (more on section 5.6). However, we can notice some technicalities in the model that need improvement and also we can propose simple improvements that yield even better results. As mentioned before, Content-Aware NODE2VEC uses neural text encoders to create node embeddings compositionally from the word embeddings of the textual descriptors associated with each node. Furthermore, one node has two node embeddings, one when it is in the start of a random walk ('focus' node) and one when this node is being predicted from the 'focus' node of another walk. In the original Content-Aware NODE2VEC paper, the neural textual encoder was one Gated Recurrent Unit (GRU) (Cho et al., 2014) (particularly a BIGRU with residual connections and max-pooling mechanism on top) for generating both node embeddings (through the textual descriptor) of a node.

A problem that we can notice with the above methodology is that this architecture might not be suitable for capturing parent/child relationships, since the single GRU cell maps the

'focus' and predicted node embeddings to the same embedding space. In this section we discuss about two key changes to the architecture of Content-Aware NODE2VEC.

First, we utilize two GRU cells for generating node embeddings for each node. One is used for the 'focus' node and the other for the predicted node, unlike our original model in Kotitsas et al., 2019, where the same RNN encoder was used for both the 'focus' and the predicted node. Secondly, to further assist the GRU cells and the resulting node embeddings, we utilize pretrained biomedical Word2Vec (Mikolov et al., 2013) word embeddings which are kept frozen, i.e., they are not updated when training our model. The trainable word embeddings are concatenated with the corresponding pretrained word embedding and are used as an input to the GRU cells. For words we do not have a pre-trained Word2Vec embedding for, we use a randomly generated embedding instead.. Figure 5.4 illustrates the proposed extensions.

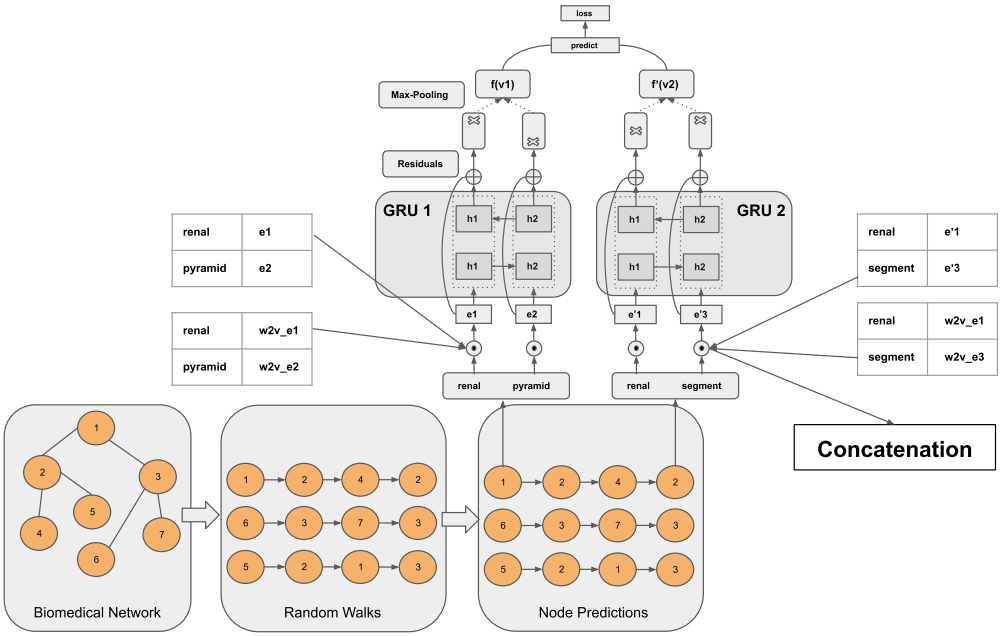

Fig. 5.4: Proposed extensions of Content-Aware NODE2VEC. For each random walk, Content-Aware NODE2VEC takes as input the word embeddings of the texts of the first node and the correct node to be predicted. We now concatenate the pretrained word embeddings (denoted `w2v_e1` for the word renal) with the trainable word embeddings (denoted e_i) and pass them through two bidirectional GRUs with residuals (GRU 1 for the focus node and GRU 2 for the predicted node). A max-pooling layer is applied to obtain the embeddings of the first and the correct predicted node of the random walk.

5.3 Content-Aware Graph Convolutional Networks

Despite the changes that were implemented to improve the performance of the Content-Aware NODE2VEC, one significant issue still remains. Unfortunately, training Content-

Aware NODE2VEC on large graphs is impractical. Each node participates in multiple random walks, and whenever we encounter a node during a random walk, we need to recompute its embedding by passing its textual descriptor through the text encoder, which is the main bottleneck. We cannot cache and re-use the embeddings of the nodes, because the parameters of the encoder (and the word embeddings) change after each prediction. A single training epoch of Content-Aware NODE2VEC on the IS-A (295k nodes) or the PART-OF (17k nodes) graph of section 3.1 takes approximately 40 or 6.5 hours, respectively. One advantage of incorporating textual with structural information with the textual encoder under the structural framework, is that these two modules are offering a way of abstraction, meaning that we can easily change the textual encoder or the structural encoder without affecting each other. So, a quick way to speed up the training process and make the methods more scalable is to change NODE2VEC. As already mentioned, the GCN framework aggregates the embeddings of the neighbors of a node and then applies a propagation function on the obtained embedding. Aggregating the embeddings of only local-neighborhoods, makes GCNs scalable and fast for large graphs. Also, by applying multiple aggregations for the neighborhood of a node, allows the learned node embedding to characterize a global neighborhood. By utilizing GCNs we can still exploit large neighborhood contexts of the graph and also speed up the training by a large margin. Compared to Content-Aware NODE2VEC, training on the IS-A, PART-OF datasets takes 4.7 hours or 1 minute respectively.

Recall, that the operations performed by a GCN layer can be summed in the following equation

$$H^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{l}W_{1}^{l}), \tag{5.4}$$

where $H^0 = X$, with $X \in \mathbb{R}^{|V|\times d}$, being the input to the GCN layer, where $|V|$ the number of nodes and d the dimensions of each node embedding. Typical GCN layers randomly initialize X and later train the node embeddings. In this section, to account for textual information, we propose to utilize the encoder of Content-Aware NODE2VEC (fig. 5.3). For each node we get its textual representation through our textual encoder, creating the matrix $X \in \mathbb{R}^{|V|\times d}$. X then is given as input node features to our GCN framework. Note that for large graphs, we need to pass each node through our encoder to create matrix X, which can lead to GPU memory problems. One solution is to pass from the encoder only the nodes that will play a role to the output predictions of the model for the current batch, significantly reducing the GPU memory requirements. Figure 5.5 illustrates only the encoding part of the nodes. Notice that the Predict box at Fig. 5.5, can be any encoding method that makes a prediction. From simply taking two node embeddings and calculating the cosine similarity between them, to a Logistic Regression Layer. More on the decoding part of our models in section 5.6.

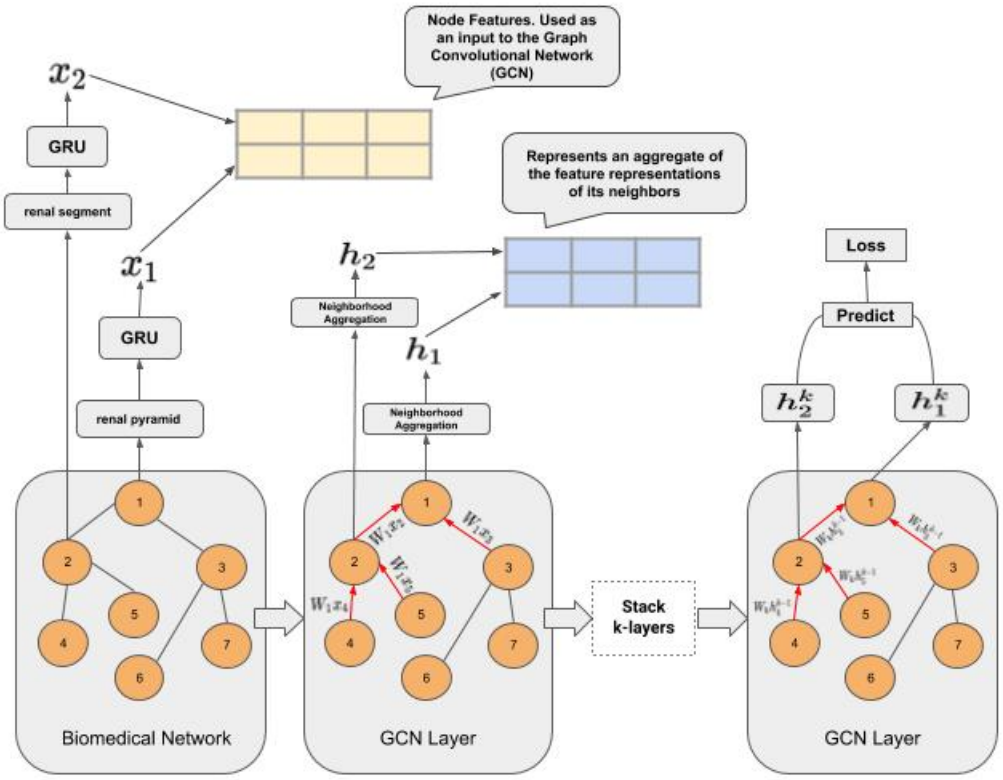

Fig. 5.5: Proposed framework that incorporates textual descriptors with Graph Convolutional Networks. As already mentioned, we pass each node's textual descriptor through the neural text encoder described in (Fig. 5.3). The result of the text encoder (x_1, x_2) can be used as input to the GCN layer (h_2,h_1). By applying the GCN propagation rule (Eq. 5.4) we create the hidden representations of the current layer. By stacking k-GCN layers we can capture information k-hops away.

5.4 Multi-Relational Content-Aware Graph Convolutional Networks on Directed Networks

The described method in the previous section can be utilized on graphs that are undirected and have a single relationship between their nodes. Real-world graphs are not so simple. Most of them are heterogeneous, meaning that they have multiple node types and multiple relationships types between the nodes, modeling complex interactions, such as in biomedical networks and in networks that represent interactions between drugs etc. GCN layers have been shown to be effective at aggregating and encoding features from network neighborhoods, and have led to significant improvements in areas such as semi-supervised classification (Kipf and Welling, 2017) and especially in multi-relational link prediction (Zitnik et al., 2018; Schlichtkrull et al., 2018), where their significance can be important, since they were used for predicting side-effects of drugs (Schlichtkrull et al., 2018).

Motivated by the work and the results in multi-relational link prediction (Zitnik et al., 2018; Schlichtkrull et al., 2018) we propose a framework that operates on directed graphs with multiple relationships and incorporates node metadata (information from textual descriptors) into the node embeddings.

First, we must rewrite the propagation rule for simple GCN layers to account for multiple relationships between the nodes. Writing equation 5.4 in its non-matrix form, we get

$$h_i^{l+1} = \sigma(\sum_{j \in N_i} \frac{1}{c_{i,j}} W^l h_j^l), \tag{5.5}$$

where h_i^{l+1} is the output of the l-th GCN layer for node i, N_i is the neighborhood of i and $c_{i,j}$ is a normalization constant as described in section 4.2. Following Schlichtkrull et al., 2018, we can now transform eq. 5.5 in order to account for multiple relationships, meaning that every relationship will be associated with its own weight matrix W_r^l with $r \in R$, where R the set of relationships. So eq. 5.5 becomes:

$$h_i^{l+1} = \sigma(\sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,j}} W_r^l h_j^l + W_0^l h_i^l), \tag{5.6}$$

with N_i^r now being the neighborhood of i under relationship r and W_0^l the self-loop weights of the l-th layer. Furthermore, we would like our model to operate on directed graphs. We want to take into account the fact that we have ancestors and descedants and model their impact differently. For example, the node embedding of a node appearing in a relationship of is_ingredient_of, must be updated differently than the node embedding of the same node appearing in a relationship of has_ingredient_of. The final propagation rule of our model becomes:

$$h_i^{l+1} = \sigma(\sum_{r \in R} \sum_{j \in N_i^{r,f}} \sum_{k \in f} \frac{1}{c_{i,j}} W_{r,k}^l h_j^l + W_0^l h_i^l), \tag{5.7}$$

with $f \in \{a, d\}$, where $W_{r,a}^l$ the weights for relationship r when taking into account the ancestors and $W_{r,d}^l$ when looking at descendants.

According to Schlichtkrull et al., 2018, applying eq. 5.7 to highly multi-relational data can lead to rapid growth in number of parameters in accordance to the number of relations in the graph. As a result, this can easily lead to overfitting on rare relationships and models of very large size. We apply a basis-decomposition to the weight matrices of each GCN layer to address this issue. More formally, with basis-decomposition each weight matrix W_r^l is defined as:

$$W_r^l = \sum_{b=1}^{B} a_{rb}^l V_b^l \tag{5.8}$$

where we can view eq. 5.8 as a linear combination of basis transformations $V_b^l \in \mathbb{R}^{d^{l+1} \times d^l}$, with coefficients a_{rb}^l, so that only the coefficients depend on the relationship r.

As in the original GCN framework the first layer takes as input a matrix $X \in \mathbb{R}^{|V| \times d}$, where X are the node features and again here we can create this matrix through the textual encoder of fig. 5.3. After we pass the node features through our encoding framework we get the final hidden representations for each node. We can then predict the possibility of an edge occurring or not. To achieve that we must calculate a score for each edge, indicating how likely it is that these two nodes are connected through a relationship of a certain type. In order to do that, following previous work we employ the Distmult (Yang et al., 2015) factorization as the scoring function, which already performs well on its own on link prediction tasks. In Distmult, each relationship is associated with a weight matrix $R_r \in \mathbb{R}^{d \times d}$ and a given triple for evaluation (s, r, t) can be scored as follows:

$$score(s, r, t) = e_s^T R_r e_o \tag{5.9}$$

where e_s, e_o are the hidden representations of the source and target nodes respectively.

Note that the above framework trains end-to-end for the multi-relational link prediction task, alleviating the need of using separate algorithms of training the node embeddings and then evaluating them (Perozzi et al., 2014; Grover and Leskovec, 2016; Busbridge et al., 2019).

We also experimented with a different architecture of the aforementioned method, where the input to the GCN layers are random initialiazed embeddings, which will be trained based only on the structure. After the GCN layers, we pass from the textual encoder of fig. 5.3 the textual descriptors of the nodes of the edges we want to classify and concat them with the corresponding structural embeddings, passing the concatenated result from an MLP to get the final node embedding. We do not present results from this variant, because the results were not improved compared to our initial framework (described in this section), presumambly because in the main framework, the node embeddings of each node are constructed from the word embeddings of its textual descriptor. The word embeddings of the textual descriptor and the parameters of the textual encoder are also updated during backpropagation to construct more informatives node embeddings along with the parameters of the GCN.

5.5 Multi-Relational Content-Aware Graph Attention Networks

So far in this thesis we have talked about methods that learn node embeddings based on random walks (NODE2VEC) and methods that iterably apply convolutions, aggregating the node features of nodes in a neighborhood and then applying a propagation function.

Typically every graph convolutional layer applies a shared node-wise feature tranformation as in eq. 5.5 (to get latent representations for each node), specified by matrix W^l.

Let's consider that the latent representation of node i at layer l is $g_i^l = W^l h_i^l$. Recall from eq. 5.5 that we also apply a normalization constant, to account for certain structural properties of the graph (i.e. the degree of each node), typically set to $c_{i,j} = \sqrt{D_{i,i} D_{j,j}}$. Let's consider $a_{i,j} = \frac{1}{c_{i,j}}$ and then equation 5.5 becomes:

$$h_i^{l+1} = \sigma(\sum_{j \in N_i} a_{i,j} g_j), \tag{5.10}$$

The motivation behind Graph Attention Layers, is that instead of explicitly defining $a_{i,j}$, we would like to treat it as a learnable parameter. As a result, GAT layers let $a_{i,j}$ to be implicitly defined, by employing self-attention mechanisms over the node features of the neighborhoods of the nodes. Self-Attention mechanisms (Vaswani et al., 2017) have been shown to be very effective and a lot of work tries to investigate the effects they have on graph representations and even solving TSP problems (Velickovic et al., 2018; Kool and Welling, 2018; Busbridge et al., 2019).

Vanilla GAT layers (Velickovic et al., 2018) let $a_{i,j}$ be a byproduct of an attention mechanism a, which can be any attention mechanism (here we focus on self-attention). a computes unnormalized coefficients $e_{i,j}$ between pair of nodes, based on their features. More formally, $e_{i,j}$ can be defined as

$$e_{i,j} = a(h_i, h_j) \tag{5.11}$$

where h_i, h_j are the node features. Each GAT layers takes as input a feature representation of each node. In order to map this node features into another embedding space and model latent interactions, we need at least one linear transformation. For that reason, every GAT layer is associated with a learnable weight matrix $W^j \in \mathbb{R}^{d_{in} \times d_{out}}$, just like in GCN layers. Equation 5.11 now becomes

$$e_{i,j}^l = a(W^l h_i, W^l h_j) \tag{5.12}$$

We can view $e_{i,j}$ as the importance of node j's features over node i's features. However, as of right now eq. 5.11 has no structural information, since it attends over all the pairs i, j. In order to account for structure, we must mask certain pairs of nodes and only calculate $e_{i,j}$ for nodes j that belong in the set of neighbors of node i, meaning $j \in N_i$, with N_i

being the neighborhood of i. We will follow Velickovic et al., 2018 and the neighbors of i will be the first-order neighbors.

As already mentioned, the coefficients $e_{i,j}$ are unnormalized. To make them comparable between the nodes of a neighborhood N_i, we normalize them using the softmax function, under the condition that the normalized coefficients of all $j \in N_i$ will sum up to 1. So the unnormalized coefficients become,

$$a_{i,j} = \text{softmax}_j(e_{i,j}) = \frac{\exp(e_{i,j})}{\sum_{k \in N_i} \exp(e_{i,k})}, \forall i : \sum_{j \in N_i} a_{i,j} = 1 \tag{5.13}$$

We still need to define the attention mechanism a. In our experiments and following (Velickovic et al., 2018) the attention mechanism a will be a feed-forward network of one single-layer, with learnable parameters $\vec{a} \in \mathbb{R}^{2d_{out}}$ and LeakyReLU as the non-linearity function (with negative slope of 0.2). Now the full equation of calculating the normalized coefficients $a_{i,j}$ of GAT layer l is:

$$a_{i,j}^l = \frac{\exp(LeakyReLU(\vec{a}^{T^l}[W^l h_i^l \| W^l h_j^l]))}{\sum_{k \in N_i} \exp(LeakyReLU(\vec{a}^{T^l}[W^l h_i^l \| W^l h_k^l]))} \tag{5.14}$$

where $\|$ is the concatenation operation.

After, the normalized coefficients are calculated we can re-write the propagation rule of GAT layers(5.15). So the hidden representations of the nodes of the $l+1$-th GAT layer are

$$h_i^{l+1} = \sigma(\sum_{j \in N_i} a_{i,j}^l W^l h_j^l), \tag{5.15}$$

with $a_{i,j}$ now being the normalized attention coefficients.

As in the work of Vaswani et al., 2017, multi-head attention has been beneficial towards experimental results and also on stabilizing the training process. The multi-head attention mechanism consists of K independent attention mechanisms that implement equation 5.15. The resulting hidden representations can be concatenated, added or averaged. We hope that the K attention heads will capture different aspects of the same neighborhood. The representations of the intermediate GAT layers are concatenated, so the output of the $l+1$-th layer is:

$$h_i^{l+1} = \|_{k=1}^K \sigma(\sum_{j \in N_i} a_{i,j}^{lk} W^{lk} h_j^l), \tag{5.16}$$

where $\|$ represents concatenation. Note, that every attention head has its own attention coeficients and weight matrix W^{lk} and concatenating each head will result in representations of $\mathbb{R}^{Kd_{out}}$.

In the last GAT layer of any architecture, according to Velickovic et al., 2018 concatenation will not help, so instead we average the output hidden representations of each head:

$$h_i^{l+1} = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N_i} a_{i,j}^{lk} W^{lk} h_j^l), \tag{5.17}$$

We have described the Graph Attention Layers, but we would like to be able to use them in heterogeneous graph where we need to also take into account the type of relationship between the two nodes, something that current GAT layers ignore. Given a triple (e_i, r, e_j), we want to model the impact of node j's features to node i's features but also account for relation r.

Recall from eq. 5.11 that each GAT layer is associated with a weight matrix W^l. In contrast to applying a linear transformation of W^l with the node features (eq. 5.12), following Nathani et al., 2019, we define for each triple (e_i, r, e_j) the representation:

$$c_{ijr}^l = W_1^l[h_i^l \| h_j^l \| g_r^l] \tag{5.18}$$

g_r^l is an embedding vector associated with relationship r, instead of having a weight matrix W_r^l associated with each relationship (5.4), since now we concatenate the representation of relationship r with the hidden representations of the nodes. After the concatenation we then perform the linear transformation as in the original GAT layer, here denoted by the weight matrix W_1^l (5.18). Again, similar to the original GAT layer, we need to first calculate the unnormalized importance of a triple (e_i, r, e_j), so we use the same attention mechanism as before (a single-layer feed forward network followed by an activation function). So the unnormalized coefficients are given by:

$$e_{ijr}^l = LeakyReLU(W_2^l c_{ijr}^l) \tag{5.19}$$

where $W_2^{(l)}$ the weight matrix of the feed forward layer.

In accordance with the original GAT layer, we apply a softmax function on the unnormalized coefficients, so that we can get comparable attention values between the nodes of a neighborhood. So,

$$a_{ijr}^{l} = \text{softmax}_{jr}(e_{ijr}) = \frac{\exp(e_{ijr})}{\sum_{n \in N_i} \sum_{k \in R_{in}} \exp(e_{ink})} \tag{5.20}$$

where R_{ij} the set of relationships between nodes i, j. The hidden representations of each layer are calculated as in the GAT layer, where we concatenate the representations of the intermediate layers and average the representation of the final layer:

$$h_i^{l+1} = \|_{k=1}^{K} \sigma\Big(\sum_{j \in N_i} \sum_{r \in R_{i,j}} a_{ijr}^{lk} c_{ijr}^{lk}\Big), \tag{5.21}$$

$$h_i^{l+1} = \sigma\Big(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} \sum_{r \in R_{i,j}} a_{ijr}^{lk} c_{ijr}^{lk}\Big) \tag{5.22}$$

with eq. 5.21 referring to the representations of the intermediate layers and eq. 5.22 referring to the last layer.

Finally, same as in 5.4, we pass the textual descriptors of the nodes through our encoder 5.3 to create the node features that will be used as an input to the multi-relational GAT framework. After we get the final node representations we use the same decoding strategy as in multi-relational GCN 5.9.

5.6 Experimental Results

In this section we will present and discuss results regarding the methods described in this chapter. We test these methods on the tasks of link prediction and multi-relational link prediction. The main idea of this chapter was about methods that incorporate structural with textual information. We will present results with the same methods and datasets as in section 4, with the only difference that we will also utilize text now. Finally, we will present results with the methods devised for multi-relational link prediction.

For link prediction with one relationship type between the nodes, we will use the IS-A and PART-OF datasets as in section 4. For the task of multi-relational link prediction, we will present results on WN18RR, FB15K-237, COVID GRAPH (as described in sections 3.2, 3.3).

Node Embedding Methods: We split our methods to those that are able to perform multi-relational link prediction and to those that can only be evaluated in single-relation link prediction.

For link prediction with one relationship type, we will present results for the extensions of Content-Aware NODE2VEC (CA-N2V) and for Content-Aware GCN (CA-GCN). We will compare

with the original CA-N2V and with CANE. CANE learns separate text-based and network-based embeddings, and uses a mutual attention mechanism to dynamically change the text-based embeddings for different neighbors. CANE only considers the direct neighbors of each node, unlike NODE2VEC, which considers larger neighborhoods obtained via random walks.

For multi-relational link prediction, we will present results with Multi-Relational Graph Convolutional Networks (MULTIREL-GCN) (to validate whether the use of textual information also helps in this particular task), Multi-Relational Content-Aware GCN (MULTIREL-CA-GCN), Multi-Relational Content-Aware Graph Attention Networks (MULTIREL-CA-GAT) and CONVE. CONVE uses convolutions over node and relationship embeddings to predict links. The full CONVE model consists of a convolutional layer, a fully connected network and an inner product to output the final prediction probabilities (chapter. 7).

In the multi-relational link prediction task, the goal is to predict a triple (s, r, t). By embedding the triple we get (e_s, R_r, e_o). Given these representations we want to predict the triple with e_s or e_o missing, i.e., predict e_s given (R_r, e_o) or predict e_o given $(e_s, R_r$. So, given a triple in an evaluation set (e_s, R_r, e_o), we generate $N-1$ (where N the number of the nodes) other corrupted triples, by replacing e_s, with every other entity e_i in the graph[1]. We then, calculate a score for every such triple, sort them in descending order and get the rank of the correct triple (e_s, R_r, e_o). Following previous work, (Bordes et al., 2013; Dettmers et al., 2018; Nguyen et al., 2018) we apply a filtered setting, meaning that during ranking we mask the corrupt triples that that happen to be true edges in the graph. The above procedure is repeated but now replacing the tail of the triple e_o, instead of the head and the average metrics are reported. We calculate Mean Reciprocal Rank (MRR) and HITS@K with $K = 1, 3$ and 10. MRR is defined as:

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}, \tag{5.23}$$

where $|Q|$ is the number of the correct triples in the evaluation sets and rank_i is the rank of each correct triple. We use MRR since we have one correct triple for every sample in the evaluation sets and we care about ranking that triple as high as possible. HITS@K is the percentage of the correct triples of the evaluation sets that are ranked in the top-k positions.

For the experiments on IS-A and PART-OF datasets, we used 30-dimensional embeddings for CA-N2V, CA-GCN, CANE for predicting the edges and 2 GCN layers for CA-GCN. Regarding the random walks of CA-N2V (and its extensions) we used the same settings as in Section 4.3. The learning rate used in these experiments is 0.01 (same as in Section 4.3) and the optimizer

[1]For the COVID GRAPH, 80000 random nodes where chosen per edge in the evaluation sets, since the number of the nodes in the graph is too large

is Adam. For WN18RR and FB15K-237 200-dimensional embeddings were used for both the entity and relationship embeddings in the decoding phase and 60-dimensional embeddings for COVID GRAPH. Due to the size of the graph, a higher number of dimensions could not be chosen. For the same reason the number of GCN layers in our models (MULTIREL-CA-GCN, MULTIREL-GCN) is 1 for COVID GRAPH and FB15K-237 datasets (in the FB15K-237 dataset we have a large number of relationships) and 2 layers for the WN18RR dataset. The optimizer was Adam (Kingma and Ba, 2015) with a learning rate of 0.001. For all the experiments, the maximum number of epochs were 250 with an early stopping of 20 epochs. In the IS-A and PART-OF datasets AUC was considered and respectively for multi-relational link prediction the metric for early stopping was MRR. PyTorch was used for all the implementations.

NE Method + Link Predictor	Random Negative Sampling	Close Proximity Sampling
CA-N2V + CS	99.5	83.3
EXT-CA-N2V + CS	**99.6**	84.5
EXT-CA-N2V-W2V + CS	99.5	**85.3**
CANE + CS	94.1	70.0
CA-GCN + CS	97.0	78.6
CA-N2V + LR	99.7	84.5
EXT-CA-N2V + LR	**99.8**	86.6
EXT-CA-N2V-W2V + LR	**99.8**	**88.4**
CANE + LR	95.3	70.0
CA-GCN + LR	98.0	80.0

Tab. 5.1: 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
CA-N2V + CS	97.5	81.4
EXT-CA-N2V + CS	97.8	82.2
EXT-CA-N2V-W2V + CS	**98.2**	**82.6**
CANE + CS	94.0	75.3
CA-GCN + CS	96.0	81.5
CA-N2V + LR	99.2	86.7
EXT-CA-N2V + LR	99.5	89.1
EXT-CA-N2V-W2V + LR	**99.6**	**91.7**
CANE + LR	94.4	76.3
CA-GCN + LR	92.0	80.0

Tab. 5.2: AUC scores (%) for the PART-OF dataset. Best scores per link predictor (CS, LR) shown in bold.

Single-Relation and Multi-Relational Link Prediction Results: Experimental results for the IS-A and PART-OF networks are reported in Table 5.1 and 5.2. For multi-relational link prediction results for the WN18RR, FB15K-237, COVID GRAPH networks can be seen in Tables 5.3.

IS-A and PART-OF datasets: These two datasets refer to single-relation link prediction. Recall the results in section 4.3. We can observe that all the methods in this section (which

also use the textual information together with structural properties), clearly outperform the models in Section 4.3, which use only structural information. So we can conclude that also modeling textual information can be crucial to learning better representations for the nodes. Furthermore, observe that all the methods perform much worse in the Close Proximity Sampling setting, indicating that indeed this approach can generate more difficult evaluation sets, reflecting better real-world problems.

CA-N2V, its extensions (EXT-CA-N2V, EXT-CA-N2V-W2V) and CA-GCN, all outperform CANE, especially in Close Proximity Sampling. Recall that CANE models only direct neighbors and does not try to model richer neighborhoods, unlike the NODE2VEC models and the GCN model (when used with more than one layer). As a result, we can propose that modeling as much graph-structure as possible, is of great importance. However, we must be careful not to model too much wide network structure, to avoid the fact of losing the notion of neighborhoods (again we can observe this phenomenon with CA-GCN in the PART-OF dataset, where the CA-GCN with LR performs the worst).

For both datasets IS-A and PART-OF, the extensions of CA-N2V obtain the best results and particularly in Close Proximity Sampling, where the differences from the other models are larger. Combining the power of textual encoding and the benefits of random walks, prove to be very efficient. However, CA-N2V and its extensions are not implemented for datasets such as WN18RR, FB15K-237, COVID GRAPH, which reflect real-world Knowledge Bases. Also, as already mentioned, the overhead of training CA-N2V and the resources needed, are large enough to limit its real-world application possibilities (training cannot be performed in a graph with more than 1 million edges). CA-GCN is a lot more scalable and faster and it yields competitive results. Additionally, CA-GCN can be extended easily in MULTIREL-CA-GCN and used in multi-relational link prediction (also trying to extend CA-N2V to multi-relational link prediction, would also make its training even slower). The importance of CA-GCN is evident as it can be used in more complex graphs and also make predictions that can be proven useful in real-world applications (as reported in chapter ??).

Finally, apart from CA-GCN, we can observe that all the models perform better with the LR predictor, rather than with the CS predictor, with the differences being larger with Close Proximity Sampling. This was expected, 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.

WN18RR, FB15K-237 and COVID GRAPH datasets: The datasets WN18RR, FB15K-237 and COVID GRAPH are graphs with directed edges and with more than one type of relationship. These datasets reflect real world use cases, where having multiple relationship types is pretty common, since in that way the graphs can represent more complex interactions and useful information. WN18RR and FB15K-237 are two common datasets for benchmarking

NE Method	MRR	HITS@K		
		@1	@3	@10
CONVE	**45.0**	**41.0**	**47.0**	**53.0**
MULTIREL-GCN	38.0	32.0	42.0	48.0
MULTIREL-CA-GCN	40.0	34.0	42.0	49.0
MULTIREL-CA-GAT	40.0	31.0	44.0	**53.0**

(a) WN18RR dataset. Best scores shown in bold.

NE Method	MRR	HITS@K		
		@1	@3	@10
CONVE	**31.0**	**22.0**	**34.0**	**49.0**
MULTIREL-GCN	28.0	19.0	30.0	48.0
MULTIREL-CA-GCN	29.0	20.0	31.0	48.0
MULTIREL-CA-GAT	28.0	19.0	31.0	47.0

(b) FB15K-237 dataset. Best scores shown in bold.

NE Method	MRR	HITS@K		
		@1	@3	@10
CONVE	**23.0**	**15.0**	25.0	39.0
MULTIREL-GCN	15.0	7.0	16.0	32.0
MULTIREL-CA-GCN	**23.0**	13.0	**27.0**	**43.0**
MULTIREL-CA-GAT	19.0	11.0	20.0	35.0

(c) COVID GRAPH dataset. Best scores shown in bold.

Tab. 5.3: MRR and HITS@K results for the datasets as described in 3.2, 3.3.

our models. We conduct experiments on these two graphs, to verify if our model is competitive or even superior to one the state-of-the-art models in these two datasets, namely CONVE. Note that we also try to capitalize on the textual information available for these two datasets. The results are shown in Table 5.3.

Our MULTIREL-CA-GCN and its variants (MULTIREL-GCN, MULTIREL-CA-GAT) do not outperform CONVE, however MULTIREL-CA-GCN remains pretty competitive and MULTIREL-CA-GAT performs the same in HITS@K with $K = 10$, which means that it ranks the same percentage of positive triples in the top 10 ranks, but lacks at ranking them in the top 3. Consequently, the differences in MRR are also larger. This presumably can be attributed to the textual information in the WN18RR dataset, since most of the entities are plain words and do not display a compositional structure, e.g "acute heart failure" and "heart failure" where "acute" can be used to specify a subtype of "heart failure", like several biomedical entities in PART-OF and IS-A datasets, where the text clearly helps. The fact that MULTIREL-GCN, which uses no text, is pretty close to the text-based models (MULTIREL-CA-GCN, MULTIREL-CA-GAT) makes the previous argument more evident. Regarding the FB15K-237 dataset, our methods still do not outperform CONVE but the differences here are even smaller. Again here we can observe that the no-text model (MULTIREL-GCN) yields the same results with the text-based models, again suggesting that the text in this dataset is not informative enough to help the models perform better.

COVID GRAPH is a biomedical dataset, where the textual descriptors of the entities are mostly phrases and not single words. This means that given a triple (s, r, o), the textual descriptor of s might have some common words or near-synonyms with the textual descriptor of o and presumably indicating some relationship. Furthermore, if we try to predict the triple (s, r, o) and the textual descriptors of s and o are similar to the textual descriptors of $s^{'}$ and $o^{'}$ of another known triple $(s^{'}, r, o^{'})$, then predicting (s, r, o) might be easier. MULTIREL-CA-GCN performs the same as CONVE in the MRR setting. However, it outperforms CONVE in HITS@K ($K = 3, 10$), meaning that MULTIREL-CA-GCN ranks more positive edges in first top-10 ranks, but it lacks at HITS@K ($K = 1$). MULTIREL-GCN performs much worse than it did in the WN18RR and FB15K-237 datasets, indicating that in COVID GRAPH the textual descriptors can be useful.

Finally, in all three datasets MULTIREL-CA-GAT performs worse than its counterparts, which was unexpected. This needs further examination, but it may be attributed to the simplistic incorporation of multiple relationships in GATs (section 5.5). More complex or different attention mechanisms can be used as future work.

6 Covid-Related Predictions

In this chapter, we conduct an evaluation of new predictions. As already mentioned, the ultimate goal of this paper is to do novel predictions using our approach and show that it is useful in the Biomedical domain. COVID-19 is an emerging and evolving situation. As a result, it would be of great importance if we could show that our approach can contribute to the fight against it and also motivate further research, regarding our NE methods and the evolution or even curation of our COVID GRAPH.

In particular, we want to identify entities of our graph that may have some relationship with the concept of COVID-19. To that extent, we ask our model, namely MULTIREL-CA-GCN, to make a prediction for every entity in the graph, and the particular relationships we are interested in (UNIDIRECTIONAL, BIDIRECTIONAL, UPREGULATE and DOWNREGULATE), that is not directly connected to COVID-19. We use these predictions to construct a ranked list of (entity, relationship, COVID-19) triples. The triples are ranked based on the probability score given by our model. After the ranking of our predictions, we filter every relationship that occurs in the cause and effect analysis provided by Causaly between UMLS entities (e.g Drugs, genes etc) and COVID-19, since these are predictions that already have been inferred. We investigate the top-100 predictions. Ideally, we would like to discover relationships that they were not present at the time we constructed our COVID GRAPH, but later on research has found that indeed the entity at question is related to COVID-19. The problem could also be formulated as a Knowledge-Base Completion task, since we discover new edges that could be added to the COVID GRAPH. To evaluate our predictions, we asked Causaly to investigate them, by searching in their Knowledge Graph for evidence and even in Pubmed. We also asked an expert to evaluate them, providing scores from 1-5, depending on how significant the prediction might be according to the expert. Based on the evaluation done by Causaly and the expert, we can divide our predictions into three classes. The first class, represents predictions predictions for which evidence has been found and the expert deemed them useful. The second class, represents predictions that are not useful and no evidence has been found about them. Finally, the third class represents predictions that we have no evidence about, but the expert thinks they might be useful, so they get a score greater than 4. Table 6.1 reports the performance of our model in the top-100 predictions.

In Table 6.2, we provide some relevant predictions that exist in our top-10 ranked predictions, two class 3 and two class 2 predictions, along with evidence where is available, snippets that support the evidence and a significance score. For example, our model con-

	Class 1 (evidence and useful)	Class 2 (no evidence and not useful)	Class 3 (no evidence and useful)
Accuracy	41/100	47/100	12/100

Tab. 6.1: Hits for each class in our top-100 predictions of entities related to COVID-19. The predicted entities were not directly linked to COVID-19 in the graph used.

Rank	Entity i	Entity j	Evidence	Snippet	Rating
2	cholesterol	covid-19	Wei et al., 2020	"Fourth ,SARS-COV-2 infection may alter vascular permeability, causing a leakage of cholesterol molecules into tissues, such as alveolar spaces, to form exudate."	5
4	lipopolysaccharides	covid-19	Anderson and Reiter, 2020	"The alterations in butyrate and LPS can promote viral replication and host symptom severity via impacts on the melatonergic pathway." "Under conditions of increased gut permeability, intestinal epithelial cells increase HMGB1 release in exosomes, suggesting that both circulating LPS and exosomal HMGB1 may contribute to viral lethality via increased gut permeability."	4
5	il33 protein human	covid-19	Adeliño et al., 2020	"Viral-activated mast cells release inflammatory chemical compounds including protease, ILs such as IL-1, IL-6, or IL-33, and histamine."	5
7	pcsk9 gene	covid-19	Banach et al., 2020	"PCSK9 inhibitors potentially reduce inflammation as a result of downregulation of LDL receptors, reduction of proinflammatory mediators, reduced infiltration of monocytes into the subendothelial layer and monocyte migration, and amelioration of vascular inflammation."	4
67	cldn4 gene	covid-19	n/a	n/a	4
82	heat shock proteins	covid-19	n/a	n/a	5
13	CERLIPONASE ALFA	covid-19	n/a	n/a	0
14	Hu3S193	covid-19	n/a	n/a	0

Tab. 6.2: Covid-Related predictions. For each prediction, we provide the rank, according to the predicted score, the relationship that the model predicted, the two entities, the publication we found the evidence in, the snippet that supported the relationship, a comment if it is available from the expert and a significance score assigned by the expert, indicating how important the prediction might be. The last four predictions are class 3 and class 2 (the ones with scores greater than 4 are class 3) and the rest are class 1.

sidered very likely an interaction between *cholesterol* with *covid-19*, hence this prediction is ranked in second. Wei et al., 2020 is the publication providing the evidence and we also provide a snippet from the publication, supporting the evidence. Lastly, according to the expert, mechanisms between *covid-19* and *cholesterol* are currently being investigated, hence the rating of this prediction as $5/5$. Note, that this edge was not present in the COVID GRAPH, the prediction is specific (not something broad) and the evidence very unlikely to be found by randomly selecting an entity and deemed it related to covid-19. Note, that in the second and third predictions, lipopolysaccharides is the same as LPS and il33 protein human is IL-33. Furthermore, in Table 6.2, the last four predictions have no evidence. However, according to the expert, *cldn4 gene* would be interesting to investigate on humans, since in mice, it plays an important role in the respiratory tract. Cldn4 concentrations

increase during acute lung injury and appear to enable alveolar fluid clearance. Another prediction worth investigating is *heat shock proteins*. Drug re-positioning suggests a role for the Heat Shock Protein 90 inhibitor Geldanamycin in treating *covid-19* infection. Lastly, *CERLIPONASE ALFA* and *Hu3S193* are not relevant at all, since no evidence was available and the expert deemed them not useful.

7 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 (P. 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). Furthermore, link prediction can be utilized in Drug-Drug Interactions (DDIs). DDIs is a research problem where traditional work relies on in vitro and in vivo experiments and has laboratory limitations (Karim et al., 2019). Representing drugs as nodes in a graph and links as interactions, the task can be regarded as a link prediction problem. One way to perform link prediction is mapping the nodes of a graph to an embedding in a vector space, while preserving the properties of the graph. Then the embeddings are given to a model as input and the model is required to predict the likelihood of an edge.

Early methods of obtaining node embeddings try to preserve the properties of the graph by applying random walks. As already noted (section 4.1), Node2Vec (Grover and Leskovec, 2016) generates random walks and then applies the Skipgram model (Mikolov et al., 2013) to the node sequences created. Grover and Leskovec, 2016 evaluate link prediction results in protein-protein interactions. D. 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. In addition, as previously discussed in section 4.2 Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017) use iterative neighbourhood averaging strategies, meaning that the model iteratively aggregates the embeddings of neighbors for a node. Then GCNs apply a function over the obtained embedding and its embedding at previous iterations to obtain the new embedding (Goyal and Ferrara, 2018). By stacking iterations, GCNs can obtain representations for non-local neighborhoods.

The aforementioned methods learn node embeddings based on structural properties. Graphs however as already mentioned are associated with metadata that characterize each node. Hamilton et al., 2017 propose GraphSAGE, which leverages node features (textual descriptors, node degrees etc) to learn node embeddings that are used in inductive learning. Other work that utilizes structure with textual information is CENE (Sun et al., 2016) which 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, unlike our previous work in Kotitsas et al., 2019 where as already discussed, we proposed Content-Aware Node2Vec which uses Node2Vec to obtain non-local neighborhoods for each node and optimizes over these neighborhoods by also considering the textual descriptors of the nodes by encoding them with RNNs.

More work on biomedical link prediction includes the work of Karim et al., 2019, who first generate a large-scale drug knowledge graph and then embed the nodes by using a combination of ComplEx embeddings (Trouillon et al., 2016) and a Convolutional-LSTM network. Another model that utilizes Convolutional Networks is ConvE (Dettmers et al., 2018). ConvE, which has already been used in the experiments of this Thesis, uses convolutions over node and relationship embeddings to predict links. The model consists of a convolutional layer, a fully connected network and an inner product for the final predictions. By applying multiple convolutions different feature maps are created and the concatenation of these feature maps constitutes a triple.

Schlichtkrull et al., 2018 extend the GCN framework to aggregate information from multiple relationships (in heterogeneous graphs) in a neighborhood of a particular node. Decagon (Zitnik et al., 2018) utilized these extensions to try and improve results on biomedical link prediction by applying multi-relational GCNs on graphs containing drug-drug, protein-protein and drug-protein interactions. GCNs however accumulate information by equally considering all edges (all have the same importance), ignoring the fact that a *causative_agent_of* relationship is possibly more important than an *isa* relationship. For that purpose, Velickovic et al., 2018 proposed Graph Attention Networks (GATs), which learn attention coefficients that act as weights, modeling the importance of each edge differently. Although GATs have been proven useful they are not suitable for multi-relational link prediction, since they do not account for multiple relationships. Nathani et al., 2019 try to incorporate the type of relationship into GATs by concatenating the node representations with the embedding of the relationship of the corresponding triple. Busbridge et al., 2019 propose Relational-GAT which also tries to model the impact of different relationships onto the node embeddings, since different relationships convey distinct pieces of information. They use a Query, Key, Value attention mechanism such as in Vaswani et al., 2017. However, their model was not evaluated on multi-relational link prediction but rather in node classification.

In this thesis, we are motivated by the aforementioned work and try to incorporate the ideas of Kotitsas et al., 2019 to GCNs and GATs framework that are trying to solve multi-relational link prediction. Our main difference with the previous methods is that we first pass the textual descriptors of the nodes through our neural textual encoder (fig. 5.3). We obtain node representations that act as an input to the GCNs and GATs frameworks, that are more informative than using randomly initialized node embeddings (as experimental results have shown). Furthermore, by doing so, we get as a by-product word embeddings that have the graph structure incorporated in them and can be used as pretrained word embeddings (Pappas et al., 2020). More formally, we generate the embedding of each node from the word embeddings of its descriptor via the encoder in fig. 5.3, but the parameters of the RNN, the word embeddings, hence also the node embeddings are updated during training to predict the likelihood of an edge. In addition, this framework is also agnostic to the node embedding framework used on top of the textual encoder. Lastly, apart from pretrained embeddings, we also get pretrained node embeddings which can also be applied to several applications as in Chalkidis et al., 2020.

8 Conclusions and Future Work

In this thesis, we addressed the problem of link prediction, with the ultimate goal of creating methods that can contribute to the scientific community, by completing Knowledge Bases or discovering new information. We partitioned the problem of link prediction, to single-relation link prediction, which applies to graphs with only one type of relationship between the nodes and to multi-relational link prediction, which applies to graphs with multiple relationships between the nodes. The latter can be applied to to more real-world applications and a lot of important information can be efficiently described with these types of graphs. To that extent, we utilized two biomedical datasets for single-relation link-prediction and two benchmark datasets for multi-relational link prediction. Furthermore, based on data provided by Causaly, we created a multi-relational graph, which contains information for the Coronaviridae family.

Regarding, the NE methods, we extended the work of Kotitsas et al., 2019 (CA-N2V), which is a method that learns node embeddings by utilizing the structural properties of the graph along with the textual descriptors of the nodes. Improved results on the single-relation link prediction task show that our improvements are useful. A key observation about CA-N2V is that it is agnostic to the framework that utilizes structure. So we were able to change NODE2VEC with GCNS and GATS. Furthermore, this enabled us to more easily create methods that are capable of handling multiple relationships. We compared our methods to one of the state-of-the-art models, namely CONVE, in multi-relational link prediction and results on the two benchmark datasets showed that our approach is competitive. In our COVID GRAPH we remained competitive and we even outperformed CONVE in HITS@K. Finally, we used our best model, to infer predictions regarding COVID-19. Upon evaluation, we found that entities, which where not connected with COVID-19 in COVID GRAPH, were indeed related and provided evidence and expert comments. This is of great importance, since the edges that were true but absent, during the creation of our graph, could be added, helping completing our Knowledge Graph (COVID GRAPH). We could also apply the same process to other Knowledge Graphs as future work.

In future work, we plan to examine more predictions of our model, trying to find edges that were absent during the creation of our graph, but later on were discovered (or even not discovered at all), facilitating in that way Knowledge Discovery, which would be remarkable for the scientific community. Lastly, we plan to investigate the not so great performance of the GAT related methods and experiment with different variants of attention mechanisms.

Bibliography

Adamic, L. A. and E. Adar (2001). "Friends and neighbors on the Web". In: *Social Networks* 25, pp. 211–230 (cit. on p. 1).

Adeliño, R., J. F. Andrés-Cordón, and C. Aracelis De La Cruz Martínez (2020). "Acute urticaria with angioedema in the setting of coronavirus disease 2019". In: *The Journal of Allergy and Clinical Immunology. in Practice* 8, pp. 2386–2387 (cit. on p. 37).

Anderson, G. and R. J. Reiter (2020). "Melatonin: Roles in influenza, Covid-19, and other viral infections". In: *Reviews in Medical Virology* 30 (cit. on p. 37).

Banach, M., P. Penson, Z. Fras, et al. (2020). "Brief recommendations on the management of adult patients with familial hypercholesterolemia during the COVID-19 pandemic". In: *Pharmacological Research* 158, pp. 104891–104891 (cit. on p. 37).

Bodenreider, O. (2004). "The Unified Medical Language System (UMLS): integrating biomedical terminology". In: *Nucleic Acids Research* 32.Database-Issue, pp. 267–270 (cit. on p. 6).

Bordes, A., N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko (2013). "Translating Embeddings for Modeling Multi-relational Data". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held in Lake Tahoe, Nevada, United States*, pp. 2787–2795 (cit. on pp. 7, 31).

Busbridge, D., D. Sherburn, P. Cavallo, and N. Hammerla (2019). "Relational Graph Attention Networks". In: *CoRR* abs/1904.05811 (cit. on pp. 26, 27, 40).

Chalkidis, I., M. Fergadiotis, S. Kotitsas, et al. (2020). "An Empirical Study on Large-Scale Multi-Label Text Classification Including Few and Zero-Shot Labels". In: *CoRR* abs/2010.01653 (cit. on p. 41).

Chen, Z., F. Chen, L. Zhang, et al. (2020). "Bridging the Gap between Spatial and Spectral Domains: A Survey on Graph Neural Networks". In: *CoRR* abs/2002.11867 (cit. on pp. 3–5).

Cho, K., B. van Merrienboer, Ç. Gülçehre, et al. (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, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734 (cit. on pp. 20, 21).

Collobert, R. and J. 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*, pp. 160–167 (cit. on p. 20).

Conneau, A., D. Kiela, H. Schwenk, L. Barrault, and A. 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*, pp. 670–680 (cit. on p. 20).

Defferrard, M., X. Bresson, and P. Vandergheynst (2016). "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain*, pp. 3837–3845 (cit. on p. 5).

Dettmers, T., P. Minervini, P. Stenetorp, and S. Riedel (2018). "Convolutional 2D Knowledge Graph Embeddings". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA*, pp. 1811–1818 (cit. on pp. 7, 31, 40).

Friedman, N., L. Getoor, D. Koller, and A. Pfeffer (1999). "Learning Probabilistic Relational Models". In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31*. Pp. 1300–1309 (cit. on p. 1).

Glorot, X., A. Bordes, J. Weston, and Y. Bengio (2013). "A semantic matching energy function for learning with multi-relational data". In: *Machine Learning* 94, pp. 233–259 (cit. on p. 6).

Goyal, P. and E. Ferrara (2018). "Graph embedding techniques, applications, and performance: A survey". In: *Knowl. Based Syst.* 151, pp. 78–94 (cit. on pp. 13, 39).

Grover, A. and J. Leskovec (2016). "node2vec: Scalable Feature Learning for Networks". In: *KDD*, pp. 855–864 (cit. on pp. 1, 11, 19, 26, 39).

Hamilton, W. L., Z. Ying, and J. Leskovec (2017). "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA*, pp. 1024–1034 (cit. on p. 39).

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA*, pp. 770–778 (cit. on p. 20).

J.Tang, M. Qu, M. Wang, et al. (2015). "LINE: Large-scale Information Network Embedding". In: *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy*, pp. 1067–1077 (cit. on pp. 1, 11).

Johnson, R. and T. Zhang (2007). "On the Effectiveness of Laplacian Normalization for Graph Semi-supervised Learning". In: *J. Mach. Learn. Res.* 8, pp. 1489–1517 (cit. on p. 3).

Karim, Md. R., M. Cochez, J. B. Jares, et al. (2019). "Drug-Drug Interaction Prediction Based on Knowledge Graph Embeddings and Convolutional-LSTM Network". In: *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, BCB 2019, Niagara Falls, NY, USA*, pp. 113–123 (cit. on pp. 1, 39, 40).

Kingma, D. P. and J. Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Conference Track Proceedings* (cit. on pp. 16, 32).

Kipf, T. N. and M. Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, Conference Track Proceedings* (cit. on pp. 1, 3, 11, 13, 14, 24, 39).

Kool, W. and M. Welling (2018). "Attention Solves Your TSP". In: *CoRR* abs/1803.08475 (cit. on p. 27).

Kotitsas, S., D. Pappas, I. Androutsopoulos, R. McDonald, and M. Apidianaki (2019). "Embedding Biomedical Ontologies by Jointly Encoding Network Structure and Textual Node Descriptors". In: *Proceedings of the 18th BioNLP Workshop and Shared Task, BioNLP@ACL 2019, Florence, Italy*, pp. 298–308 (cit. on pp. 1, 2, 6, 15, 18–22, 40–42).

Lei, C. and J. Ruan (2013). "A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity". In: *Bioinformatics* 29.3, pp. 355–364 (cit. on p. 39).

Li, Q., Z. Han, and X. Wu (2018). "Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA*, pp. 3538–3545 (cit. on p. 4).

Lu, L. and T. Zhou (2010). "Link Prediction in Complex Networks: A Survey". In: *CoRR* abs/1010.0725 (cit. on pp. 1, 11).

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held in Lake Tahoe, Nevada, United States*, pp. 3111–3119 (cit. on pp. 4, 11, 12, 19, 22, 39).

Nathani, D., J. Chauhan, C. Sharma, and M. Kaul (2019). "Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, Volume 1: Long Papers*, pp. 4710–4723 (cit. on pp. 29, 40).

Nguyen, Dai Q., T. Nguyen, Dat Q. Nguyen, and D. Q. Phung (2018). "A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana, pp. 327–333 (cit. on p. 31).

Pappas, D., P. Stavropoulos, and I. Androutsopoulos (2020). "AUEB-NLP at BioASQ 8: Biomedical Document and Snippet Retrieval". In: *CLEF* (cit. on p. 41).

Paszke, A., S. Gross, S. Chintala, et al. (2017). "Automatic differentiation in PyTorch". In: (cit. on p. 16).

Perozzi, B., R. Al-Rfou, and S. Skiena (2014). "DeepWalk: online learning of social representations". In: *KDD*, pp. 701–710 (cit. on pp. 1, 4, 11, 26).

Schlichtkrull, M., T. N. Kipf, P. Bloem, et al. (2018). "Modeling Relational Data with Graph Convolutional Networks". In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, Proceedings*. Vol. 10843, pp. 593–607 (cit. on pp. 13, 24, 25, 40).

Schuster, M. and K. K. Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Trans. Signal Processing* 45, pp. 2673–2681 (cit. on p. 20).

Sen, P., G. Namata, M. Bilgic, et al. (2008). "Collective Classification in Network Data". In: *AI Magazine* 29.3, pp. 93–106 (cit. on p. 11).

Shen, D., X. Zhang, R. Henao, and L. 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*, pp. 1829–1838 (cit. on pp. 1, 18, 40).

Shojaie, A. (2013). "Link Prediction in Biological Networks using Multi-Mode Exponential Random Graph Models". In: (cit. on p. 39).

Sun, X., J. Guo, X. Ding, and T. Liu (2016). "A General Framework for Content-enhanced Network Representation Learning". In: *CoRR* abs/1610.02906 (cit. on pp. 1, 39).

Toutanova, K., D. Chen, P. Pantel, et al. (2015). "Representing Text for Joint Embedding of Text and Knowledge Bases". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal*, pp. 1499–1509 (cit. on p. 7).

Trouillon, T., J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard (2016). "Complex Embeddings for Simple Link Prediction". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA*. Vol. 48, pp. 2071–2080 (cit. on p. 40).

Tu, C., H. Liu, Z. Liu, and M. 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, Volume 1: Long Papers*, pp. 1722–1731 (cit. on pp. 1, 18, 40).

Vaswani, A., N. Shazeer, N. Parmar, et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA*, pp. 5998–6008 (cit. on pp. 27, 28, 40).

Velickovic, P., G. Cucurull, A. Casanova, et al. (2018). "Graph Attention Networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*. OpenReview.net (cit. on pp. 27–29, 40).

Wang, D., P. Cui, and W. 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*, pp. 1225–1234 (cit. on pp. 1, 39).

Wang, P., B. Xu, Y. Wu, and X. Zhou (2015). "Link prediction in social networks: the state-of-the-art". In: *SCIENCE CHINA Information Sciences* 58.1, pp. 1–38 (cit. on p. 39).

Wang, Y. and J. Zeng (2013). "Predicting drug-target interactions using restricted Boltzmann machines". In: *Bioinformatics* 29, pp. 126–134 (cit. on p. 1).

Wei, X., W. Zeng, J. Su, et al. (2020). "Hypolipidemia is associated with the severity of COVID-19". In: *Journal of Clinical Lipidology* 14, pp. 297–304 (cit. on p. 37).

Yang, B., W. Yih, X. He, J. Gao, and L. Deng (2015). "Embedding Entities and Relations for Learning and Inference in Knowledge Bases". In: *CoRR* abs/1412.6575 (cit. on p. 26).

Yu, W., C. Zheng, W. Cheng, et al. (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*, pp. 2663–2671 (cit. on pp. 1, 39).

Zitnik, M., M. Agrawal, and J. Leskovec (2018). "Modeling polypharmacy side effects with graph convolutional networks". In: *Bioinform.* 34.13, pp. i457–i466 (cit. on pp. 1, 13, 24, 25, 40).