POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING

Master of Science in Computer Engineering

Master Degree Thesis

Deep Learning on Polito Knowledge Graph

Leveraging Relational GCN for link prediction between nodes of a newly built publications graph

Supervisors
Prof. Antonio Vetrò
Prof. Juan Carlos De Martin

Candidate
Giovanni Garifo

Academic Year 2018-2019

To Monia
To my Grandfather

Abstract

Summary here, one page

Acknowledgements

Acknowledgements here, half page

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Graphs are used to empower some of the most complex IT services available today, an example among all being the Google search engine [1]. They can be used to represent almost any kind of information, and they are particurlarly capable of representing the structure of complex systems and describe the relationships between their elements.

Over the last decade, much effort has been made in trying to leverage the power of graphs to represent human knowledge and to build search tools capable of querying and understanding the semantic relations within them. RDF graphs are a particular class of graphs that can be used to build knowledge repositories. Given a domain and an ontology, they allows to build a structured representation of the knowledge in such domain.

Modern machine learning techniques can be used to mine latent information from such graphs. One of the main challenges in this field is how to learn meaningful representations of entities and relations that embed the underlying knowledge. Such representations can then be used to evaluate new links inside the graph or to classify unseen nodes. Deep learning techniques have proved to be first class citizens when dealing with representation learning tasks, being able to learn latent representations without any prior knowledge other than the graph structure, so as not to require any feature engineering.

[1] https://blog.google/products/search/introducing-knowledge-graph-things-not/

1.2 Thesis structure

1.2.1 Chapter 2

1.2.2 Chapter 3

1.2.3 Chapter 4

Chapter 2

Background

2.1 Semantic Web

2.1.1 From a Web of contents to a Web of data

The World Wide Web has been developed as a tool to easily access documents and to navigate through them by following hyperlinks. This simple description already resembles the structure of a graph: we can think of documents as nodes, and of hyperlinks as edges. The unstoppable growth of the *Web graph* led to the emergence of new tools to extricate in such complexity. Search engines have been developed to easily navigate such a giant graph, initially by scoring search results based on trivial statistics, such as the number of times a document has been linked, as in the case of the PageRank [1] algorithm developed by Google.

The Web rapidly became one of the most innovative technology ever built, allowing to retrive information quickly and easily as never before. The next evolutionary step has been to think about a Web not only exploitable by human beings but also by machines. In order to build such a comprehensive system, where information can be not only machine-readable, but machine-understandable, the World Wide Web had to move from a web of content, to a web of data.

The World Wide Web Consortium (W3C) introduced the Semantic Web as an extention to the prior standard of the WWW. Its primary goal has been to define a framework to describe and query semantic information

contained in the documents available on the web, so as to allow machines to understand the semantic information contained in web pages. In the vision of Tim Berners-Lee, the father of WWW, this would bring to the transition from a World Wide Web to a Giant Global Graph [1], where a web page contains metadata that provides to a machine the needed information to understand the concepts and meanings expressed in it.

2.1.2 The Semantic Web building blocks

The three key components of the Semantic Web standard are:

1. OWL: the Web Ontology Language
2. RDF: the Resource Description Framework
3. SPARQL: The SPARQL Protocol and RDF Query Language

OWL is a language used to define ontologies. In this context, an ontology is defined as a collection of concepts, relations and constraints between these concepts that describes an area of interest or a domain. OWL allows to classify things in terms of their meaning by describing their belonging to classes and subclasses defined by the ontology: if a thing is defined as member of a class, this means that it shares the same semantic meaning as all the other members of such class. The result of such classification is a taxonomy that defines a hierarchy of how things are semantically interrelated in the domain under analysis. The instances of OWL classes are called individuals, and can be related with other individuals or classes by means of properties. Each individual can be characterized with additional information using literals, that represent data values like strings, dates or integers.

RDF is a XML-based framework that defines a standard model for the description, modelling and interchange of resources on the Web.

The first component of the framework is the *RDF Model and Syntax*, which defines a data model that describes how the RDF resources should be represented. The basic model consist of only three object types: resource, property, and statement. A resource is uniquely identified by an

[1] https://web.archive.org/web/20160713021037/http://dig.csail.mit.edu/breadcrumbs/node/215

Uniform Resource Identifier (URI). A property can be both a resource attribute or a relation between resources. A statement describes a resource property, and is defined as a triple between a subject (the resource), a predicate (the property) and an object (a literal or another resource).

The second component of the framework is the *RDF Schema* (RDFS), that defines a basic vocabulary for describing RDF resources and the relationships between them. Many vocabularies have been built on top of RDFS, such as the Friend of a Friend (FOAF) vocabulary [2], for describing social networks, or the one maintained by the Dublin Core Metadata Initiative [2], that defines common terms used in the definition of metadata for digital resources.

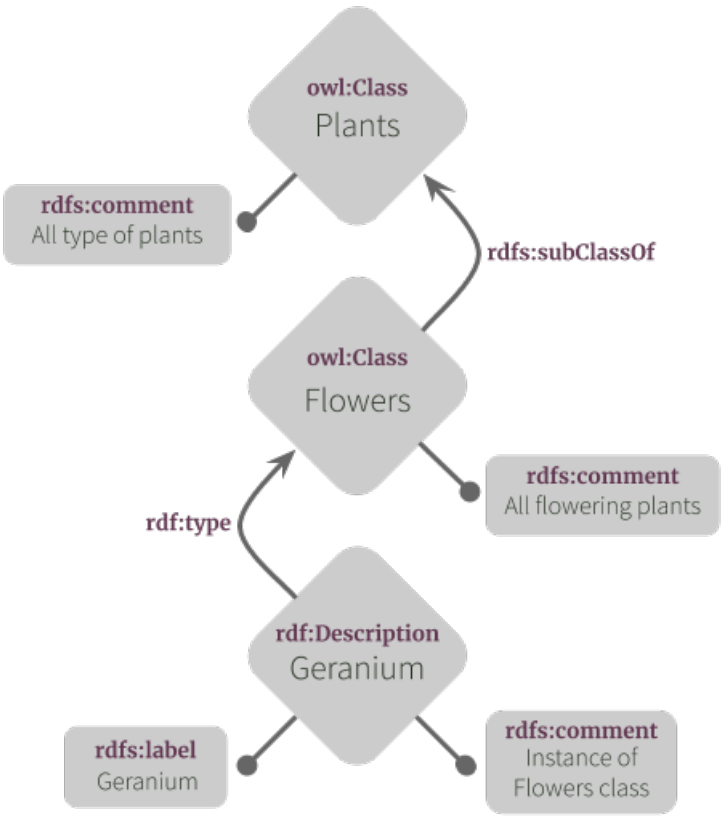

Figure 2.1. An example of ontology defined using OWL and RDF Schema.

[2] https://www.dublincore.org/

SPARQL is a query language for triplestores, a class of Database Management Systems (DBMS) specialized in storing RDF databases. Such DBMS often expose endpoints that can be used to query the database and obtain results. Given the complexity of the data stored, the query language has been designed to be as simple as possible, in example by allowing the use of variables, whose definition is preceded by a question mark.

The syntax of SPARQL is heavily derived from SQL, with some minor adaptations to be more suited for querying graphs data. The following is an example of query which selects all the labels (human-readable description of a resource) of all the entities that matches the given resource type.

```
PREXIF plants:<http://example.org/plants/>

SELECT ?name
WHERE {
    ?subject rdf:type plants:flowers .
    ?subject rdfs:label ?name .
}
```

2.1.3 Knowledge Bases as knowledge repositories

Even if the raise of the Semantic Web has suffered a slowdown in its growth due to the complexity of its vision, many new projects were born from its enabling technologies. Efforts have been put by profit and non-profit organizations in trying to build complex knowledge repositories starting from the knowledge already present in the Web. An example among all is the DBpedia[3] project, which developed a structured knowledge base from the unstructured data available on Wikipedia. Another example is the *Google Knowledge Graph*[4], which is used to enhance the Google search engine and virtual assistant capabilities, allowing to retrieve punctual information about everything that has been classified in its ontology and described

[3] https://wiki.dbpedia.org/

[4] https://blog.google/products/search/introducing-knowledge-graph-things-not/

in its knowledge base, or the *Open Academic Graph*[5], a Scientific Knowledge Graph that contains, describes and links more then three hundred million academic papers.

From an implementation perspective, knowledge bases can be created to describe a specific domain by defining an ontology and a vocabulary for such domain using OWL and RDF Schema, and then by describing the concepts of such domain using the RDF Model and Syntax. The RDF document obtained can then be stored in a triplestore and queryed using SPARQL. The biggest effort when building knowledge bases is to have a correct understanding and prior knowledge of the domain of interest, to avoid the risk of mischaracterizing and misrepresenting concepts.

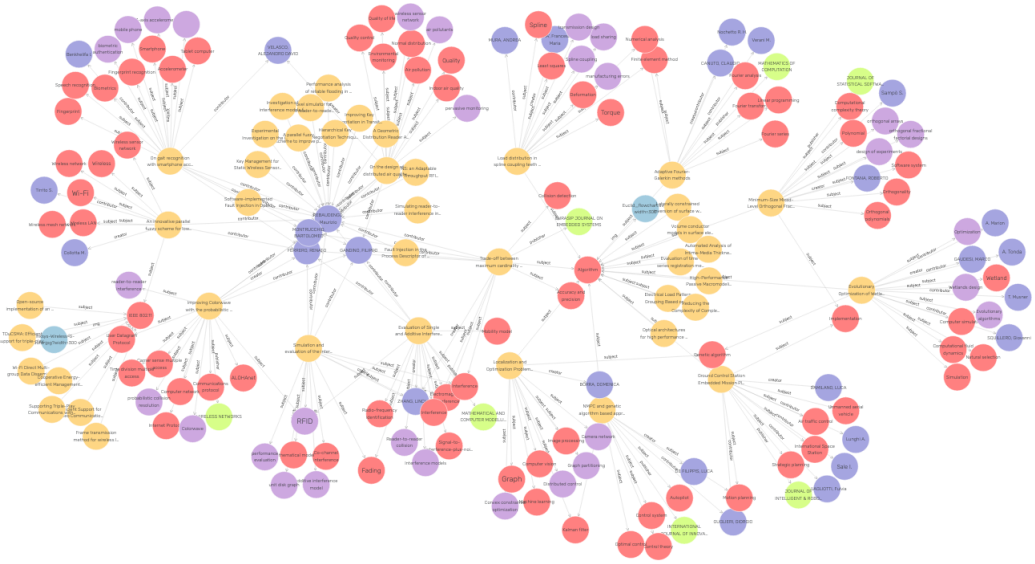

Figure 2.2. An extract of the Polito Knowledge Graph that will be introduced and described in the following chapters.

If all the requirements and cautions are met, a well formed knowledge base may prove to be a critical resource for an organization. It allows not only to build new services upon it, but also to improve the existing knowledge inside the organization by performing reasoning upon the available knowledge, thus to discover implicit facts that can be derived from existing relationships. Another field of applications is the development of Expert

[5] https://www.openacademic.ai/oag/

Systems, AI software that emulates the behavior of a human decision-making process by navigating the knowledge base and taking decisions like in a rule-based system.

Today's knowledge bases are commonly composed by tens of thousands nodes and by hundreds of thousands of edges, such giant data structures pose many challenges. Not only storing and querying giant graphs requires the adoption of specialized DBMS that are capable of efficiently store and query the RDF input representation, but also doing analysis and gathering statistics from such giant graphs requires the adoption of highly efficient algorithms in order to retrieve the desired output in an acceptable time.

The availability of such a complex and informative data structure leads to the opening of interesting scenarios, especially when thinking about the latent information that can be extracted from it. In fact, a knowledge base is a structured representation of the human knowledge in a specific field, thus its comprehensiveness is restricted by the human understanding.

2.2 Learning on Graphs

2.2.1 Representation learning

Machine learning (ML) algorithms are used to learn models from the available data, with the final goal to obtain a set of parameters that are fine-tuned to identify seen characteristics in the data used for training. The models obtained can be later used to recognize unseen inputs by leveraging the knowledge embedded in such parameters. ML algorithms require the input data to be available in a machine-understandable vector representation. An important task in the ML field is the learning of such representations, task known as representation learning.

Natural Language Processing (NLP) is one of the research branches that in the past years has made a great use of machine learning algorithms both for language recognition and for embedding words *meaning* into words *vectors*. One of the most successful algorithms when dealing with representation learning of words is Word2Vec [3], where the model obtained is trained to learn a vector representation for each word in a vocabulary. In Word2Vec, the concept of meaning of a word is related to the context in which such word is frequently used, so two words are recognized as similar if they're used in similar contexts, thus in the vector

space of the learnt representations words that have similar meaning have higher cosine similarity with respect to dissimilar ones. For instance, the cosine similarity between the word vectors of "Man" and "King" is roughly the same as the one between the words "Woman" and "Queen", since such words are used in similar contexts. This has open up new scenarios for language recognition and processing, since it allowed to perform vector operations on such words which brought interesting results, as can be seen in figure 2.3.

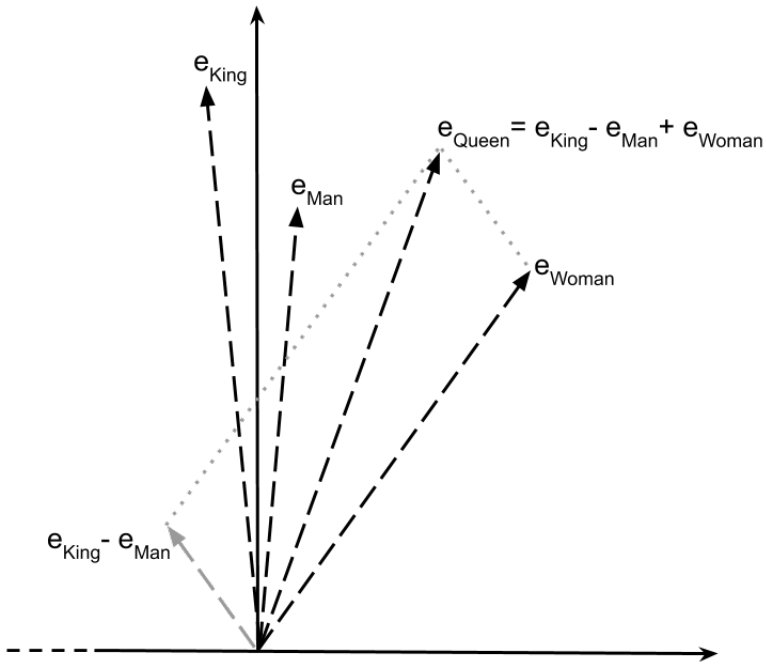

Figure 2.3. Word vectors allows to perform vector operations, the results obtained reflect the fact that Word2Vec is capable of embed the meaning of such words.

The idea that words can be characterized by the context in which they're used can be generalized and used into other fields of research, such as the field of representation learning on graphs.

Graphs are composed by nodes and edges, and are used to describe complex systems, such as social networks or the interactions in a molecular biology system. To apply machine learning algorithms to such data structures, in order to analyze the available data and predict new facts, vector representations of such nodes and edges are needed. Such vector

representations are often referred to as *embeddings*.

Early approaches required these representations to be learned from feature vectors that where handcrafted, task that required not only a relevant amount of effort, but also a deep understanding of the domain of interest. This has long been one of the main obstacles when dealing with representation learning tasks, since who has knowledge of the domain and who has to engineer the features were unlikely the same individual.

2.2.2 Deep Learning on graphs

In the latest years a big shift towards deep architectures has been made in machine learning, mainly thanks to the development of highly parallelized architectures that are able to efficiently compute at the hardware level vector and matrix multiplications, operations that are at the basis of any machine learning task. Deep Learning (DL) algorithms are able to extract relevant features from raw data by applying simple mathematical operations, such as convolution, to the input data. An example of one of the most successful applications of DL is in image recognition, where matrix representations of images are convolved with self-trained filters that are able to extract the relevant features needed to recognize patterns present in the input images.

Deep learning techniques have proven to function well also in the field of representation learning for graph data, this should not give rise to surprise, given that as can be seen in figure 2.4, a digital image is composed by pixels which can be thought of as nodes in a graph, where each pixel is connected by an edge to its immediate neighbors. This suggests that the techniques used when dealing with images can be adapted, with some major changes, to the field of representation learning on graphs, but also in other fields of research, such as learning on manifolds.

The problems when working with graph data is that commonly graphs are built to describe complex systems, such as the knowledge of a domain or field for knowledge graphs, and thus are composed of a fairly high amount of nodes and edges. The matrices used to store the graph structure can thus explode in dimensionality, so as to became impractical as input data. Moreover, graphs aren't regular structures with given shape and size, like a matrix of pixels for images, but they live in an irregular domain which led to highly irregular structures. The first problem can be solved by randomly sampling the graph at each training epoch,

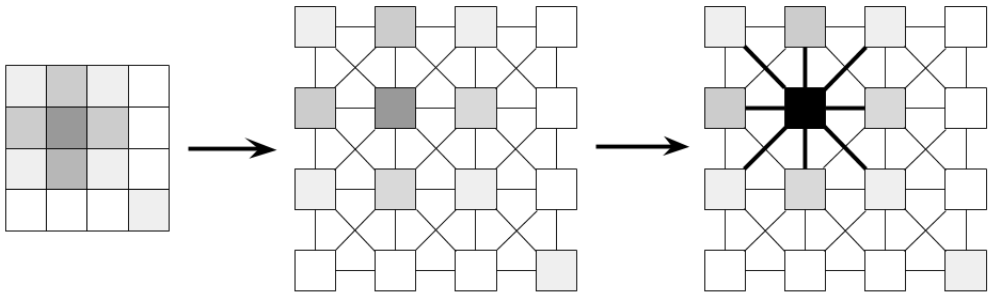

Figure 2.4. A digital image can be thought of as a graph.

the immediate drawback being that more than one epoch is required to train over all nodes. The second problem can instead be solved by adapting known algorithms to work on highly irregular domains. One of the possible approaches, which has proven to work well, is the one based on convolutions.

Graph Convolutional Networks (GCNs) [4] are a class of semi-supervised deep learning algorithms for graphs which are based on the same convolution and backpropagation operations as the so famous Convolutional Neural Networks (CNNs) used for feature learning on images. The main difference between CNNs and GCNs is in the architecture of the neural network, and so in how the convolution is performed, instead the backpropagation phase is the same as the one used to update the parameters of CNNs, with the loss function being task-specific. In a CNN the input matrix of each network layer, which is the pixel matrix of the input image for the first layer, is convolved with a convolutional filter, whose parameters are then updated during the backpropagation phase.

GCNs works differently, but similarly, by convolving at the l-th layer of the network the feature vector of each node with the feature vectors of its l-nearest neighbors, this is done by applying the following transformation:

$$H^{l+1} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{l}W^{l}) \tag{2.1}$$

Where H^l is the output of the previous layer or, for the input layer, the nodes feature matrix where each row is commonly initialized as a one-hot encoded feature vector. $\tilde{A}$ is the adjacency matrix of the graph summed with the identity matrix to add self loops of nodes, $\tilde{D}$ is the node degree matrix of $\tilde{A}$ and is used to normalize it, W^l is the weight matrix

of the layer, that is shared among all nodes for each layer, just like the convolutional filter in a CNN, and $\sigma()$ is a non linear activation function, like $ReLU$.

Analyzing the forward rule for a single node make it more clear how the embedding of a node is updated through the convolution:

$$h_i^{(l+1)} = \sigma(\sum_{j \in \eta_i} \frac{1}{c_{i,j}} h_j^{(l)} W^{(l)}) \tag{2.2}$$

Setting aside the normalization costant c_{ij} which is obtained from the multiplication between the adjancency and degree matrices, at each layer the updated feature vector of the node i is obtained by summing over all the nodes in its neighborhood the result of the multiplication between the neighbors feature vectors and the weight matrix of the layer. A consequences of applying this rule to all nodes is that at the l-th layer the feature vectors of nodes that are at a l hop distance from the node i will be embedded in its feature vector, because the feature vectors of such nodes were embedded, at the previous layer, by the immediate neighbors of node i, which are now passing the information of their surrounding to it. So the amount of layers of the network is a parameter that controls how much information from furthest nodes has to be collected by each node embedding.

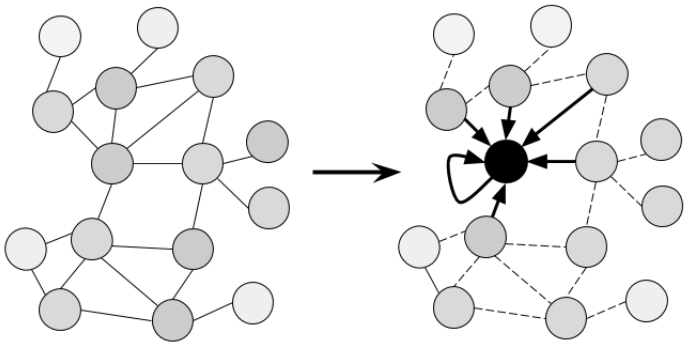

Figure 2.5. First layer of a GCN updating a node feature vector by embedding the features of adjacent nodes.

The result is that each node embedding will be characterized by its context, just like it happens in Word2Vec, but in a non-Euclidean domain. So for example, in a social graph where each person is characterized by

its friends, interests, places visited and so on, two people will have similar embeddings if they are connected to similar nodes.

The node embeddings obtained by applying a GCN or one of its variants can then be used to perform some learning task on the graph, like classification of unseen nodes or link prediction of non-existent edges, the latter being one of the most interesting task because it empowers most of the recommendation systems available in the industry.

2.2.3 Link prediction on Knowledge Graphs

Predicting new facts is one of the most common task in the field of knowledge graph completion. The goal of such task is to evaluate unseen triples in order to evaluate wether or not there are good candidates to be included in the knowledge base. Link prediction can be done in two steps, by firstly train an encoder model that is able to embed the node features and produce meaningful embeddings, and then by applying a factorization model that is able to score the unseen triples under evaluation.

Deep learning techniques such as GCN can be exploited to obtain meaningful embeddings of nodes, but fall short when dealing with knowledge graphs due to the inability of taking into account the diversity of the relations between the nodes. In example, using a GCN, two nodes that have the same neighborhood but are connected to the surrounding nodes with different relations will have the same node embedding, even if it's clear that their characteristics are not the same. To overcome this limitation, changes to the basic GCN architecture have been proposed, so to obtain models that works well when applied to multi-relational graphs.

Relational Graph Convolutional Networks [5] (RGCNs) are an extension of GCNs that are focused on modeling multi-relational graphs composed by directed edges, and thus are particularly capable of embedding both nodes and relations of a knowledge graph. RGCNs can be used for both link prediction and node classification tasks.

The idea behind this new architecture is to have different set of parameters for different relations. At each step inside the network, the feature vector of a node is updated by convolving its first neighbors features with a convolutional filter that is different based on the kind of relation that connects the nodes. The forward rule to update the embedding of a node at the l layer is the following:

$$h_i^{(l+1)} = \sigma\left(W_0^{(l)} h_i^{(l)} + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)}\right) \tag{2.3}$$

Where h_i is the embedding (or, for the first layer, the input feature vector) of the node i, W_0 is the learnt filter for the relation between the node and itself (called self loop), $\mathcal{N}_i^r$ is the set of indices of the neighbors of node i under the relation $r \in \mathcal{R}$, with $\mathcal{R}$ being the set of all the relations present in the graph. W_r is the learnt filter for the relation r. As for the GCN architecture, σ is a non linear activation function and $c_{i,j}$ is a normalization constant, commonly initialized to $|\mathcal{N}_i^r|$.

As can be seen the update rule looks similar to the one for GCNs (2.2), with the major difference that in the case of a RGCN the parameters used to convolve the feature vectors of neighboring nodes are relation specific, this means that the kind of relation that connect the node to its neighbors has an important role in determining the node embedding.

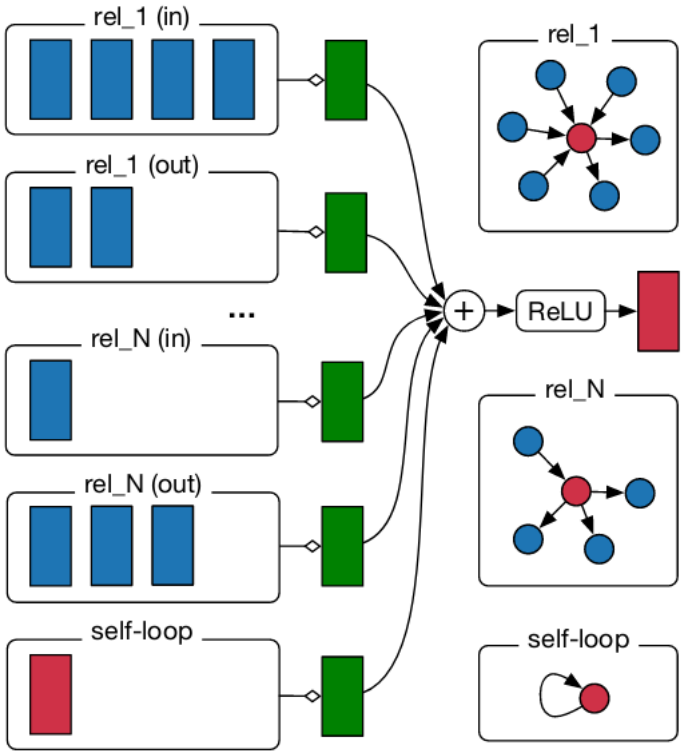

Figure 2.6. RGCN encoder model. Image taken from [5].

Some form of regularization is required in order to avoid overfitting on rare relations, thus to obtain a more generalized model, and also to avoid the rapid growth in the number of parameters of the network for highly multi-relational graphs. One of the solutions proposed by the original paper [5] is to decompose each relation filter W_r using basis decomposition:

$$W_r^{(l)} = \sum_{b=1}^{B} a_{r,b}^{(l)} V_b^{(l)} \tag{2.4}$$

This allows to store only the relation specific coefficients and the basis, which will be shared by all the relations.

The model obtained by training an RGCN can then be used to build an encoder that given as input a graph, gives as output the embeddings of all nodes and the relations parameters. Then a factorization method can be used to evaluate unseen facts inside the graph, exploiting the embeddings obtained. Such methods are used as scoring functions in order to obtain, starting from the entities embeddings, a real value that can be used to score the unseen triples under evaluation.

DistMult [6] is one of the most common and simple factorization methods used to score unseen triples. Given as input the embeddings of the source node and destination node, and the relation parameter vector transformed into a diagonal matrix $R_r \in \mathbb{R}^{d \times d}$, it computes an associated real valued score:

$$f(s, r, o) = e_s^T R_r e_o \tag{2.5}$$

The score obtained by applying such function can then be used to evaluate wether the triple (s, r, o) is a good candidate to be added to the graph, an high score is to be interpreted as a high confidence of the model in the fact that the triple should belong to the knowledge graph.

Chapter 3

Approach and methodology

3.0.1 Polito Knowledge Graph empowering tools

IRIS dump, topic extraction with TMF, RDFlib.

3.0.2 Machine Learning tools

* rgcn complessa, usa un sacco di parametri, creare rete da zero molto complesso senza strumenti adeguati
* Deep Graph Library in soccorso, breve descrizione sul progetto, citazione del paper
* Spiegare architettura di dgl, spiegando il message passing framerwork
* hardware used.

Chapter 4

Related work

Open Academic Graph as example of existing Scientific KG.

CSO Classifier as example of existing topic extraction tool for scientific publications.

Closing with: to the best of my knowledge I'm the first to use RGCN for link prediction on a scientific KG.

Chapter 5

Development and implementation

5.0.1 Creating Polito Knowledge Graph

5.0.2 Training model on PKG

5.0.3 Link prediction on PKG

Chapter 6

Evaluation

evaluating the result obtained from link prediction

Chapter 7

Conclusions and future work

Bibliography

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Technical Report 1999-66, November 1999, previous number = SIDL-WP-1999-0120. [Online]. Available: http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

[2] M. Graves, A. Constabaris, and D. Brickley, "Foaf: Connecting people on the semantic web," *Cataloging & Classification Quarterly*, vol. 43, no. 3-4, pp. 191–202, 2007. [Online]. Available: https://doi.org/10.1300/J104v43n03_10

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv e-prints*, p. arXiv:1301.3781, Jan 2013, provided by the SAO/NASA Astrophysics Data System. [Online]. Available: https://arxiv.org/pdf/1301.3781.pdf

[4] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv e-prints*, p. arXiv:1609.02907, Sep 2016, provided by the SAO/NASA Astrophysics Data System. [Online]. Available: https://arxiv.org/pdf/1609.02907.pdf

[5] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F978-3-319-93417-4_38.pdf

[6] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014. [Online]. Available: https://arxiv.org/pdf/1412.6575.pdf