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. They can be used to represent almost any kind of information, and they are particurlarly capable of representing the structure of complex system, thus to express the relations between its elements.

In the past ten years, a lot of effort has been put into trying to leverage the power of graphs to represent human knowledge and to build search tools capable of query and understand the semantic relations inside such graphs. 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 representaion of the knowledge of such domain.

Modern machine learning techniques can be used to mine latent informations 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 the 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.

1.2 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 content 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, by scoring search results based on trivial statistics, such as the number of times a document has been linked.

The Web rapidly became one of the most disruptive technology ever built, but it's power was limited to the fact that it was exploitable only by human beings. To build a more comprehensive system, where informations can be not only machine-readable, but machine-understandable, thus to allow new usage of such a giant source of informations, the WWW 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. It's primary goal has been to define a framework to describe and query semantic informations contained in the documents available on the web, so as to allow machines to understand the semantic informations contained in web pages. In the vision of Tim Berners-Lee, the father of WWW, this will bring to the transition from a World Wide Web to a Giant Global Graph, the GGG, 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 standards 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 allows to describe 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 informations 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 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, for describing social networks, or the one maintained by the Dublin Core Metadata Initiative, that defines common terms used in the definition of metadata for digital resources.

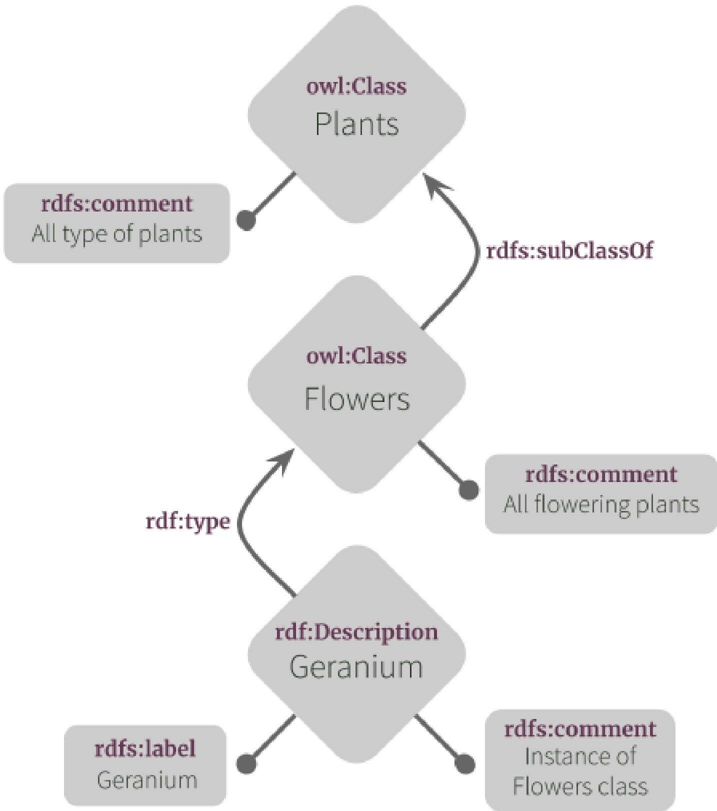

Figure 2.1. An example of ontology defined using OWL and RDF Schema.

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 select 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 stunting in its growth due to the complexity of it's vision, many new project empowered by it's enabling technologies have arise. 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 project, which developed a structured knowledge base from the unstructured data available on Wikipedia. Another example is the so called "Knowledge Graph" made by Google, which is used to enhance it's search engine and virtual assistant capabilities, allowing to retrieve punctual informations about everything that has been classified in it's ontology and described in it's knowledge base.

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 can then be stored in a triplestore that can be queryed using SPARQL. The biggest effort when building knowledge bases is to have a correctly understanding and prior knowledge of the domain of interest, to avoid the risk of mischaracterizing and misrepresenting concepts.

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

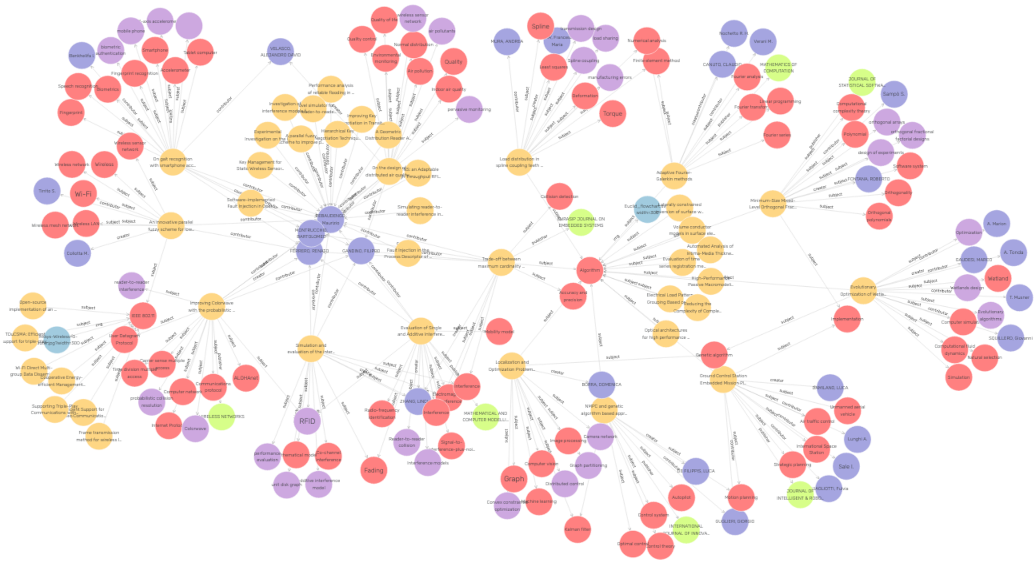

Figure 2.2. An extract of the Polito Knowledge Graph.

In a Big Data era, knowledge bases can't be any less. The vastity of human knowledge is reflected onto the complexity of the graphs builded from it. 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 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 informations that can be extracted from it. In fact, a knowledge base is a structured representation of the human knowledge in a specific field, thus it's completess 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 requires 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 knows as representation learning (RL).

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 succesful algorithms when dealing with representation learning of words is Word2Vec, 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 recognize as similar if they're used in similar contextes, thus in the vector space of the learnt representations words that have similar meaning have higher cosine similarity with respect to dissimilar ones. For intance, the cosine similarity between the word vectors of "man" and "woman" is roughly the same as the one between the words "King" and "Queen", since such words are used in similar contexes. This can led to new scenario for language recognition and processing, since it allows to perform vector operations on such words which can led to interesting results, as can be seen in figure 2.3.

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 refered to as "embeddings".

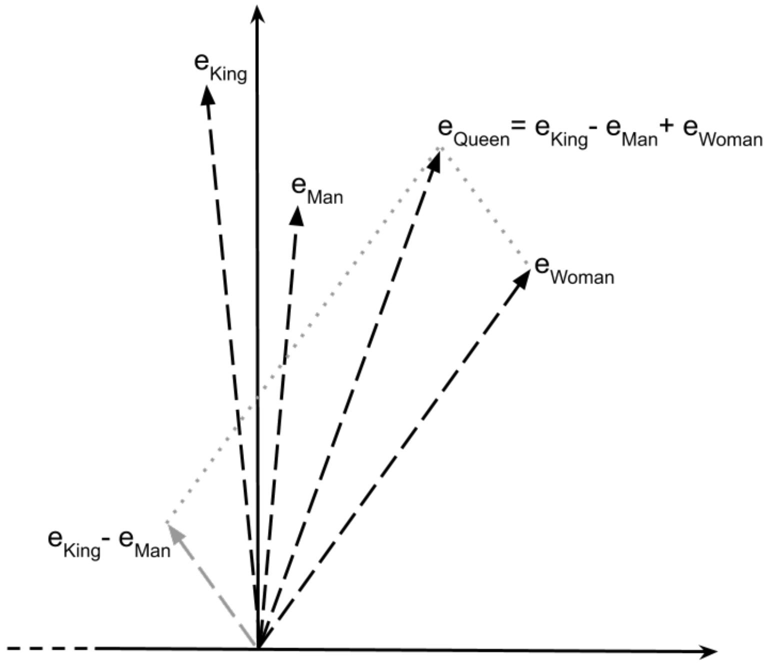

Figure 2.3. Word vectors allows to perform vector operations, the results obtained reflect the fact that Word2Vec is capable of embed the meaning of such words.

Early approaches required these representations to be learned from feature vectors that where handcrafted from the ground up, 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 the most succesful applications of DL is in image recognition, where the matrix representations of the input 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 it's first neighbours. 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.

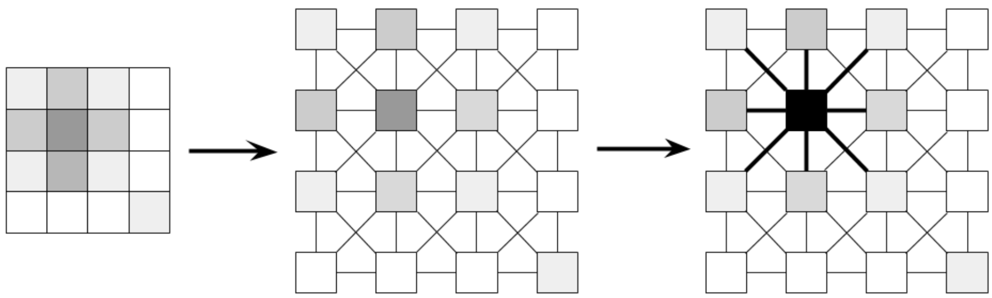

Figure 2.4. A digital image can be thought of as a graph.

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, 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 know algorithms to work on highly irregular domains. One of the possible approaches, which has prooven to work well, is the one based on convolutions.

Graph Convolutional Networks (GCNs) are a class of semi-supervised deep learning algorthms for graphs which are based on the 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, 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 it's 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:

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

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 informations for farest nodes has to be collected from each node embedding.

This means that each node is characterized by it's surroundings, 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 it's friends, interests, places visited and so on, two people will have similar embeddings if they are connected to similar nodes.

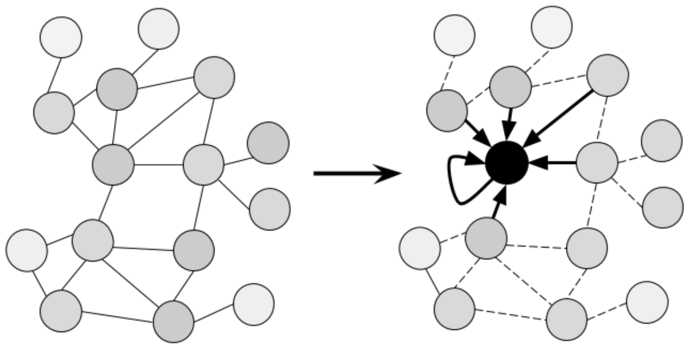

Figure 2.5. First layer of a GCN updating a node feature vector by embedding the features of adjacent nodes.

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.

Chapter 3

State of the art

Chapter 4

Approach and Methodology

Chapter 5

Development and Implementation

Chapter 6

Evaluation

Chapter 7

Conclusions

Bibliography

[1] G. Galilei, *Nuovi studii sugli astri medicei*, Manuzio, Venetia, 1612.

[2] E. Torricelli, in “La pressione barometrica”, *Strumenti Moderni*, Il Porcellino, Firenze, 1606.

[3] E. Torricelli e A. Vasari, in “Delle misure”, *Atti Nuovo Cimento*, vol. III, n. 2 (feb. 1607), p. 27–31.

[4] Duane J.T., *Learning Curve Approach To Reliability Monitoring*, IEEE Transactions on Aerospace, Vol. 2, pp. 563-566, 1964