Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Master in Data Science

Real use cases of property graphs and their exploitation via data analysis

Course project on Semantic Data Management

Project report

Arnau Arqué and Daniel Esquina

Group 11

Professor: Oscar Romero

June 23, 2022

Contents

List of Figures

List of Tables

1 Theoretical backbone

In this section of the deliverable we will summarize our research on the topic and the findings we have found. Precisely we will take a look at the concept of embedding and particularly graph embeddings. We will then focus on the different kinds of graphs that can be used for this task and the components to be embedded. Lastly, we will share a gathering of the different techniques used and current applications in literature.

1.1 Embeddings

Embedding is a technique used in many areas of study that represents the same concept. The simple idea behind this notion is to represent a complex structure in a low dimensionality space. The aim of an embedding is to capture some of the semantics of the data structure in order to place similar inputs together in the embedding space. Conversely, inputs semantically far should be represented remotely in this new representation. This method could be encapsulated in the family of dimensionality reduction procedures.

1.2 Embeddings in Machine Learning

One of the main areas in where this technique is extended is Natural Language Processing. In particular, this area implements word embeddings, which are fixed length vector representations for words [1]. Multiple approaches could be taken to make this transformation, for instance, methods which leverage the local information of a word versus methods that take into account general statistics from a corpus. Neural Networks for Natural Language Processing is one of the areas which mostly benefits from this concept [2]. In Machine Learning it is very common to have high dimensional data which can be represented fairly well with in a lower dimensional manner. Evidently, this transformation does not come for free, there is a trade-off and minimal information is lost in favour of reducing the scale of the problem. Using embedding techniques improves usability and efficiency of these computations the inputs of which tend to be massive.

1.3 Graph embedding

In our case we will focus this study on graph embeddings, also known in literature as network embeddings [3]. These particular techniques aim to achieve a low dimensional representation of some specific component of a graph. A graph can be defined as an structure that contains a set of nodes and a set of edges. These embeddings, would transform one of these sets to a low dimensional space. However, abstractions could also be applied in order to transform a substructure of the graph (subset of nodes and edges) or even the whole graph into a low dimensional vector. These abstractions will be further discussed in the section 1.3.2.

These representations could later be used to solve a classification problem, a prediction or node clustering for instance. In section **??** we will take a look at the wide range of possibilities for this models.

1.3.1 Embedding input

We could view this process of creating an embedding as a function which turns our graph data structure into a space of low dimensions. Using this analogy, an input of an embedding method can be defined as the to be transformed original structure. There are different typologies of graphs and each of them cause different challenges for this techniques.

If we take a close look at the morphology of the graph Cai, Zheng, and Chang define four different graph skeletons. On the one hand, **homogeneous graphs** are the ones whose nodes and edges only belong to a single type. The challenge associated to this type of graphs is preserving the connectivity patterns only with the given structural information. On the other hand, **heterogeneous graph** nodes and edges have different types although they are embedded into the same space, creating a consistency problem which can be further increased when there is type imbalance.

The aforementioned author also makes a difference on **graphs with auxiliary information** in either of its structures. The challenge of embedding these graphs is properly combining the topological and auxiliary information to define node similarity. The last morphology that can be defined are **graphs from non-relational data**, the challenge of which is precisely how to compute these relationships in order to subsequently make the embedding.

Other considerations particular to the edges of a graph are the existence of weights and directions. Some embedding algorithms take into account these information, for instance, the weight of and edge, to quantify proximity between the two connected nodes.

1.3.2 Embedding output

Following the aforementioned function analogy, the output of an embedding method can be defined as the low dimensional space solution found for a given input. In this case, the author Cai, Zheng, and Chang reflects on the granularity of these outputs.

As we have previously commented, any compositional structure of a graph, even itself, is a candidate to be embedded. If we assume a dimension of 1, **node embedding** will represent each node of the graph as a vector. The same will happen for **edge embedding** and **whole-graph embedding**. There is an additional kind of embedding which contemplates the combination of different components named **hybrid embedding**, an example of it with graphlets is introduced by Nikolentzos, Meladianos, Tixier, *et al.* Regardless of the embedding used, the vector representation of a component will aim to be similar between instances of the same component with high resemblance.

1.4 Graph Embedding techniques

In order to find good solutions to this problem, many different approaches have been considered. Each of them is based on a concept that researches found could be applied to this issue. There are multiple classifications for embedding methods. Some authors like Goyal and Ferrara, propose a classifications of three kinds of procedures, the ones **based on random walks**, the ones **based on factorization** and the ones **based on deep learning**. In contrast, Cai, Zheng, and Chang proposes five different taxonomies that will be presented below. Each of these are further divided into categories but we will not specify that lower granularity.

1.4.1 Matrix Factorization

This type of approaches obtain the embedding from factorizing a matrix that represents a graph property. The input is commonly a matrix computed from a no-relational data features as we commented on section 1.3.1. An example of this is presented by the authors Belkin and Niyogi using the Laplacian Eigenmaps.

1.4.2 Deep Learning

This classification of algorithms entails a composition of the adopted models from other fields and specific neural networks. The input they take is the graph or paths sampled from it. An example from the first case is explained by the authors Wang, Cui, and Zhu. In contrary to this previous example, Perozzi, Al-Rfou, and Skiena uses an approach which uses random walks to sample paths from the graph. Random walks is a process that, starting from a vertex, describes a path from a succession of random steps.

1.4.3 Edge Reconstruction

In this cluster, algorithms that aim to optimize objective functions based on edge reconstruction. They can either maximize edge reconstruction probability or minimize edge reconstruction loss. In other words, these methods try to reconstruct some edges that may not be explicit in a graph, that may have been removed or that may be inferred based on other information [10] [11].

1.4.4 Graph Kernel

Graph kernels represent each graph as a vector by interpreting its substructures and comparing them. The input of these methods are whole-graphs and there are multiple components that can be tested like the random walks we mentioned earlier, subtree patterns or graphlets. Nikolentzos, Meladianos, Tixier, *et al.* used this technique as we have commented in a previous section with graphlets (connected and non-isomorphic subgraphs), but other examples can be found in literature like the one proposed by Yanardag and Vishwanathan.

1.4.5 Generative Model

The input of generative graph models is either a heterogeneous or an auxiliary information graph and the output can either be node or edge embedding. There are many possibilities as the idea behind this concept is to combine a set of extracted features for each instance with their distribution and its associated label. The authors Xiao, Huang, Meng, *et al.* propose an embedding strategy over knowledge graphs that uses textual descriptions of entities and the definition triplets to project the graph.

1.4.6 Hybrid techniques and others

Just like in any other taxonomy attempt, there are many other approaches not contemplated in this classification which follow a very specific course of action. There are also techniques which implement a couple of the above mentioned methods in hopes of joining the best of their effects.

1.5 Graph embedding applications

Graphs are used widely to model data, they are schemaless by nature and their semantics are fixed by edge and node labels. Thanks to this concept and the fact that they follow an Open-World assumption, these data structures are able to represent a wide variety of real world concepts. As they are used in many different fields, each with its own purpose, techniques are developed to fulfill particular needs.

The output of the embedding can be used to group several applications as the ones related to the whole-graph, for example, can be seen as a family with different goals but the same parting point.

Just like in any other learning task, the goal of an study can be selected between two tasks: doing a prediction or a classification. In this case, embeddings can be used for classification by, for instance, detecting the label of some nodes based on the learnt information from a training set. In a similar fashion, judging from the already inferred knowledge we could predict if an edge exists between two nodes. A part from this two mentioned classes, other considered applications are network compression, visualization and clustering [6].

Network compression is the concept of simplifying a graph with two main purposes. The first objective is to store less data, and the second, to consequently run the desired algorithms faster. Both goals are related to efficiency and the trade-off to take into account is how much to reduce graph data versus losing its intrinsic information.

2 Practical backbone

In this section of the project we will put into practice what we have learnt. We will use a graph to compute a machine learning algorithm with a specific purpose. To run this algorithm we will need to propose and implement an embedding strategy into our graph. This will be a first contact with the topic, with the main objective of getting familiar with this technology so we can correctly apply it in further studies.

2.1 Graph data

The graph data that we will use is the one we presented on the first project (see figure 2.3). It is a syntetic dataset which we generated to represent the concepts of the paper publication domain. The following concepts were generated (in parentheses we mention its attributes): keywords (id, name), authors (id, name, country), conferences (id, name), conference editions (name of conference, num. of edition, city, year), journals (id, name, available online), journal volumes (name of journal, num. of volume, month of publ., year of publ., num. of pages, issn num.), papers (id, title, month of publ., num. of pages, issn num., abstract, published), companies (name) and universities (name, country). Initially, a total of 200 (out of 2000) unpublished papers are created. To approach the project with enough data, we created a total of 57 keywords, 100 authors associated with 9 companies and 10 universities, 20 conferences and journals with 5 editions/volumes each and 2000 papers.

Once the basic concepts were created, we proceeded to generate the relationships between them. Each paper contains between 1 and 10 citations, a single main author, between 1 and 5 co-authors (among whom there won't be the main author), between 3 and 5 reviewers (among whom there won't neither be the main author nor none of the co-authors) and between 3 and 5 keywords. In addition, each author has a relationship with their company or university.

For this study, in addition to the already specified graph, we wanted to try the behaviour of a more manageable graph in order to better visualize the task at hand and solution provided. We have decided to use a subgraph of the original one and apply the same embedding technique on both of them and compare the results obtained. The second graph we will use only contains the papers and the authors, with their roles (main author, coauthor and reviewer) represented as relationships.

2.2 Machine learning task

Now that we have chosen the two graphs for this project, we will propose a machine learning task to solve. We have opted for a node classification task, specifically, we will try to determine the type of node from instances of a test set (20% of our data). To do so, we will use a training set (80% of our data) to train a Logistic Regression classifier [14].

Logistic Regression is a classification technique that uses a logistic function to model the dependent variable. It has multiple assumptions like the absence of multicollinearity. As this is not the focus of the study, we will not further investigate on this technique since we have already done so in other subjects.

2.3 Embedding proposal

This graph can be classified as a graph with auxiliary information, however, for this study we will only consider an homogeneous representation of it. This is the case because the embedding solution we will propose is node2vec [15].

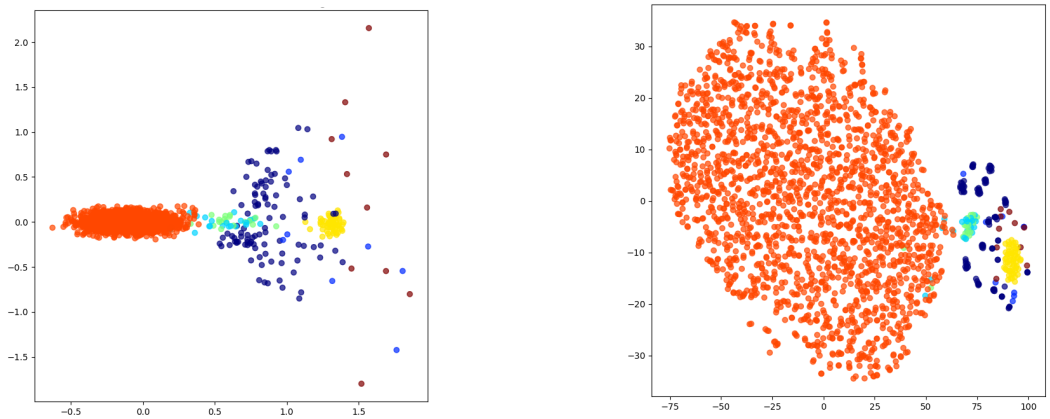
Node2vec is an embedding method which preserves proximity between nodes by maximizing the probability of occurrence of subsequent nodes in fixed length random walks [6]. In fact, it uses biased-random walks that provide a trade-off between breadth-first (BFS) and depth-first (DFS) graph searches. Node2vec is an embedding of the Deep Learning class we defined on section 1.4.

Deep learning node embeddings with random walks, like node2vec, can automatically exploit the neighbourhood structure through sampled paths on the graph, however, fail to recognize the global structure information. In the next section we will take a look at the results obtained with our two experiments.

2.4 Practical results

In this section we will share the obtained results when using a Logistical regression for node classification with node embeddings. As we have previously commented, we have tried two different graphs from the same data in order to check how different structures can affect the learning process.

The first experiment takes into account all nodes and relationships of the defined graph and we can see a visualization of the embeddings using TSNE in the figure 2.1 and PCA in the figure 2.1a. There is a clear separations between concepts. Paper, in orange, is clearly the most represented class and the embedding perfectly reflects the difference between the difference between this node type and the others.



(a) Using PCA.

(b) Using TSNE.

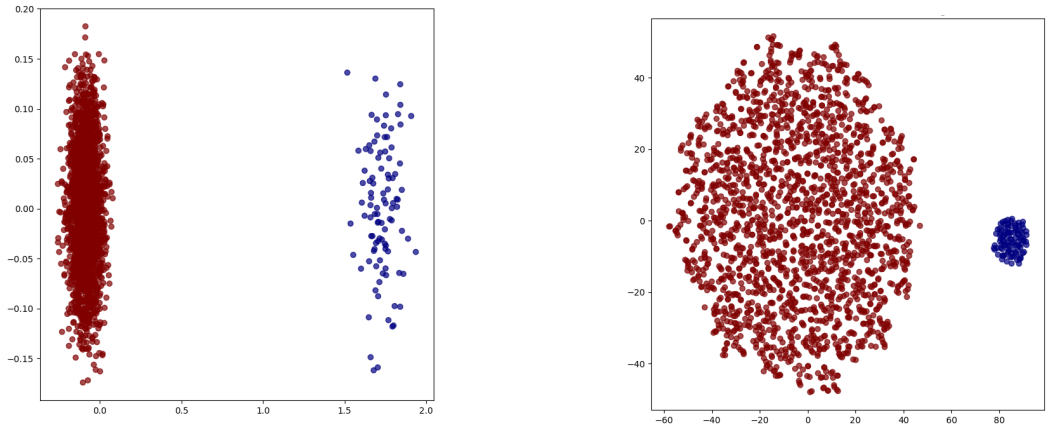
Figure 2.1: Visualization of node embedding.

Table 2.1: Confusion matrix of the prediction phase.

	Author	Company	Conference	Journal	Keyword	Paper	University
Author	**20**	0	0	0	4	0	0
Company	0	**0**	0	0	3	0	0
Conference	3	0	**0**	0	0	3	0
Journal	2	0	0	**0**	0	1	0
Keyword	0	0	0	0	**9**	0	0
Paper	0	0	0	0	0	**397**	0
University	0	0	0	0	1	0	**1**

In table 2.1 we can see a confusion matrix that represents the obtained classification results of the first experiment. This experiment considers all nodes and relationships in the graph and yields an accuracy result of 96.8%. This accuracy is higher than expected but it makes sense taking into account that the data is synthetic and there is not much random variation between nodes of the same type. With such a high accuracy result we expect the classifier to perform equally well with a simpler task.

In figures 2.2a and 2.2 we can see that this time there is not a single doubt whether a node belongs to a type. In this case, we are very surprised to see how the classification perfectly matches reality even with the unbalanced data that it presents.



(a) Using PCA.

(b) Using TSNE.

Figure 2.2: Visualization of node embedding.

In the table 2.2 we can see the classification results obtained for the first subgraph. Only the author and paper nodes have been considered, in addition to their relationships. The yielded accuracy for this small sample is 100,0%. The already presented hypothesis is confirmed, comparing these results does not make much sense anymore because this data, although having a randomized number of associations, is too clean for such a powerful classification technique.

Table 2.2: Confusion matrix of the prediction phase using the Author-Paper's subgraph.

	Author	Paper
Author	**14**	0
Paper	0	**406**

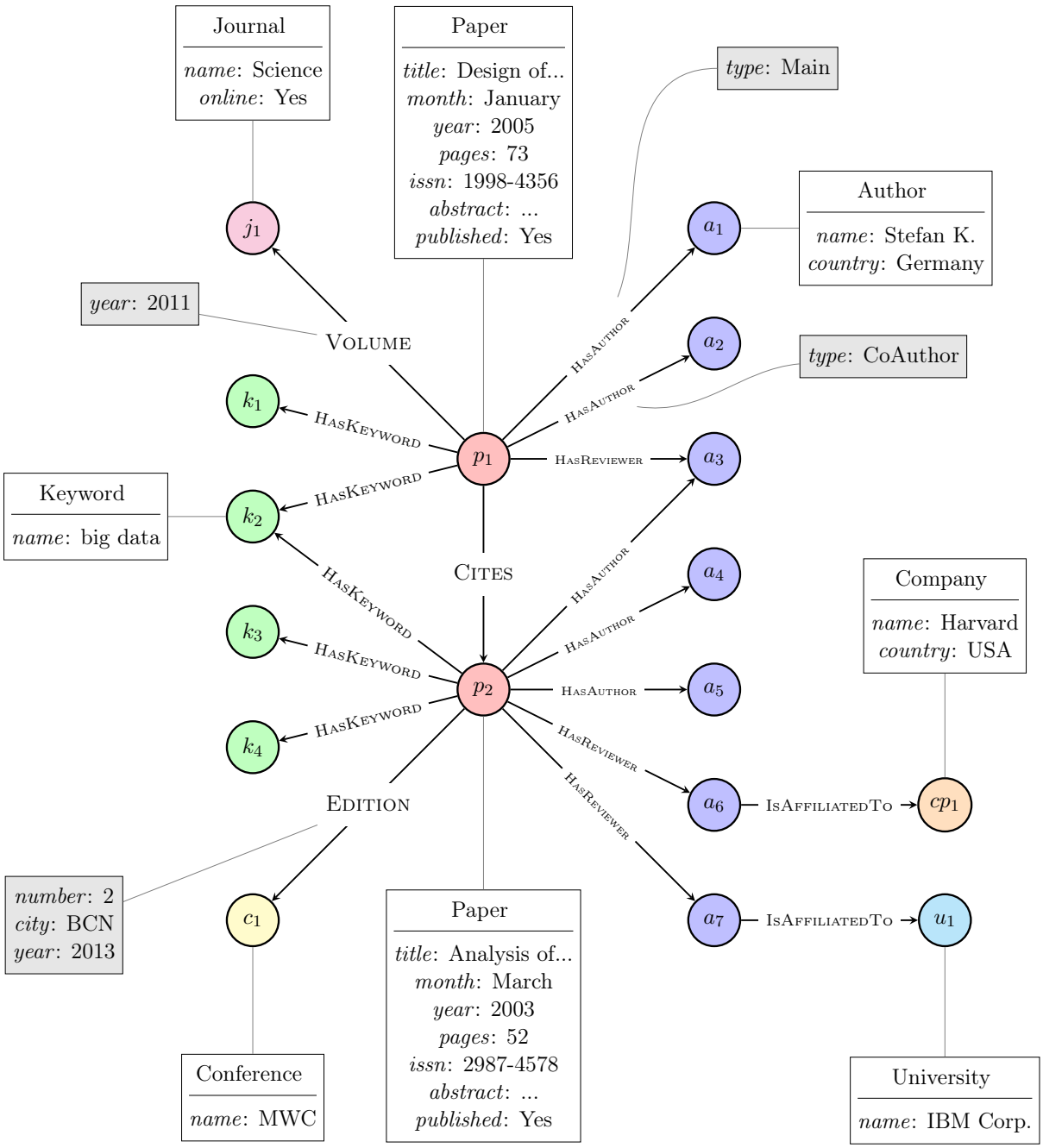


Figure 2.3: Visual representation of the graph used in the practical backbone.

Bibliography

[1] F. Almeida and G. Xexéo, "Word embeddings: A survey," *arXiv preprint arXiv:1901.09069*, 2019.

[2] S. Ghannay, B. Favre, Y. Esteve, and N. Camelin, "Word embedding evaluation and combination," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016, pp. 300–305.

[3] H. Chen, B. Perozzi, R. Al-Rfou, and S. Skiena, "A tutorial on network embeddings," *arXiv preprint arXiv:1808.02590*, 2018.

[4] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[5] G. Nikolentzos, P. Meladianos, A. J.-P. Tixier, K. Skianis, and M. Vazirgiannis, "Kernel graph convolutional neural networks," in *International conference on artificial neural networks*, Springer, 2018, pp. 22–32.

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

[7] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[8] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1225–1234.

[9] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[10] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng, "Predict anchor links across social networks via an embedding approach.," in *Ijcai*, vol. 16, 2016, pp. 1823–1829.

[11] X. Wei, L. Xu, B. Cao, and P. S. Yu, "Cross view link prediction by learning noise-resilient representation consensus," in *Proceedings of the 26th international conference on World Wide Web*, 2017, pp. 1611–1619.

[12] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.

[13] H. Xiao, M. Huang, L. Meng, and X. Zhu, "Ssp: Semantic space projection for knowledge graph embedding with text descriptions," in *Thirty-First AAAI conference on artificial intelligence*, 2017.

[14] M. Maalouf, "Logistic regression in data analysis: An overview," *International Journal of Data Analysis Techniques and Strategies*, vol. 3, no. 3, pp. 281–299, 2011.

[15] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.