



UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Rio Claro

Filipe Alves de Fernando

A Semantic Rank-Based Graph Embedding

Rio Claro - SP

2020

Filipe Alves de Fernando

A Semantic Rank-Based Graph Embedding

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Geociências e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de Rio Claro.

Financiadora: Petrobrás/Fundunesp

Orientador: Prof. Dr. Daniel Carlos Guimarães Pedronette

Co-orientador: Prof. Dr. Ivan Rizzo Guilherme

Rio Claro - SP
2020

F363s Fernando, Filipe Alves de
A Semantic Rank-Based Graph Embedding / Filipe
Alves de Fernando. -- Rio Claro, 2020
72 p. : il., tabs.

Dissertação (mestrado) - Universidade Estadual
Paulista (Unesp), Instituto de Geociências e Ciências
Exatas, Rio Claro
Orientador: Daniel Carlos Guimarães Pedronette
Coorientador: Ivan Rizzo Guilherme

1. Network Representation Learning. 2. Machine
Learning. 3. Graphs. 4. Artificial Intelligence. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp.
Biblioteca do Instituto de Geociências e Ciências Exatas, Rio Claro.
Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

ATESTADO DE APROVAÇÃO - DEFESA

Atestamos que **FILIPPE ALVES DE FERNANDO**, RA nº: CCO180165, RG nº 40.980.108-2, expedido pela SSP/SP, defendeu, no dia 29/09/2020, a dissertação intitulada ***Semantic Rank-Based Graph Embedding***, junto ao Programa de Pós Graduação em Ciência da Computação, Curso de Mestrado Acadêmico, tendo sido 'APROVADO'.

Atestamos ainda que a obtenção do título dependerá de homologação pelo Órgão Colegiado competente.

São José do Rio Preto, 29 de setembro de 2020

A handwritten signature in black ink, appearing to read "Silvia Emiko Kazama".

Silvia Emiko Kazama
Supervisor Técnico de Seção
Seção Técnica de Pós-Graduação

Filipe Alves de Fernando

A Semantic Rank-Based Graph Embedding

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação - Área de Concentração em Computação Aplicada, como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Financiadora: Petrobrás/Fundunesp

Banca Examinadora

Prof. Dr. Daniel Guimarães Pedronette (Orientador)
Instituto de Geociências e Ciências Exatas (IGCE)
Universidade Estadual Paulista - UNESP

Prof. Dr. Marco Antonio Garcia de Carvalho
Faculdade de Tecnologia (FT)
Universidade Estadual de Campinas - UNICAMP

Prof. Dr. Fabricio Aparecido Breve
Instituto de Geociências e Ciências Exatas (IGCE)
Universidade Estadual Paulista - UNESP

UNESP

29 de setembro de 2020

Acknowledgements

The author would like to thank Professor Ivan Rizzo Guilherme for encouraging him to join the MSc. program and advising him during this period; The author is also very grateful to Professor Daniel Carlos Guimarães Pedronette for advising him and directly collaborate with the proposal and ideas presented in this work. The author is grateful to LIAAP (Artificial Intelligence Laboratory Applied to Oil) members and its personal friends, specially Gustavo José de Sousa and Lucas Pascotti Valem, who collaborate with some results presented in this work and with the writing of the resultant papers of this work. Finally, the author would like to express his gratitude to Petrobrás and Fundunesp (Fundação para o Desenvolvimento da UNESP) for financially supporting him (process nº Fundunesp 2014/00545-0) during his master's research.

Resumo

Grafos são ferramentas poderosas, capazes de capturar relações complexas existentes entre objetos. Em decorrência de sua robustez e flexibilidade, diversas aplicações podem beneficiar-se de modelagens baseadas em grafos. No entanto, o processamento de grafos, principalmente em cenários de larga escala, é computacionalmente custoso e muitas vezes inviável. Métodos de *Graph Embedding* são usados para encontrar representações vetoriais de baixa dimensionalidade para grafos, preservando suas características principais como topologia, afinidade e vizinhança compartilhada entre nós, possibilitando que tarefas analíticas como classificação, agrupamento, recuperação de informações, predição de conexões e outras possam ser executadas. Com isso em mente, este trabalho, propõe RaDE (Rank Diffusion Embedding), um método eficiente e eficaz para aprender representações vetoriais de baixa dimensionalidade de nós de grafos. O método proposto identifica representantes altamente eficazes em grafos baseados em modelagens de ranqueamento, além de ser totalmente não supervisionado e independente do domínio de dados. Uma vantagem do RaDE sobre os métodos concorrentes é que cada dimensão das representações vetoriais geradas possui um significado semântico. Além do RaDE, este trabalho também propõe o RaDE+, que é uma variante do RaDE. O RaDE+ considera múltiplos representantes ao invés de um único por dimensão. A abordagem proposta foi avaliada em 8 conjuntos de dados tais como redes sociais, redes de co-referências, documentos textuais e conjuntos de imagem. Destacando que um dos conjuntos de dados avaliados é um conjunto de imagens de raio-x de pulmões de pessoas com Covid-19, pneumonia e saudáveis. O RaDE/RaDE+ foram aplicados neste conjunto de dados para ambas as tarefas: visualização e recuperação de informação e as representações vetoriais geradas por ambos os métodos alcançaram resultados de alta eficácia, superando os resultados dos métodos concorrentes na maioria dos conjuntos de dados.

Palavras-chave: Aprendizado de representação de grafos, Aprendizado de máquinas, Grafos, Inteligência Artificial.

Abstract

Graphs are powerful tools, capable of capture complex relationships existing among objects. Due to its robustness and flexibility, many application can benefit from gaph-based modeling. However, performance issues can be observed specially on large scale networks, making it computationally unfeasible to process information in various scenarios. Graph Embedding methods are usually used for finding low-dimensional vector representations for graphs, preserving its main properties such as topological characteristics, affinity and shared neighborhood between nodes, making it possible that analytical tasks such as classification, clustering, information retrieval and link prediction to be performed. To address theses problems, this work, proposes RaDE (Rank Diffusion Embedding), an efficient and effective approach to learn low-dimensional vector representations for nodes in graphs. The proposed method identifies high-effective representatives nodes in rank-based modeled graphs besides being completely unsupervised and independent of the data domain. A benefit of RaDE over the competitor methods is that each dimension of the vector representations generated by RaDE has a semantic meaning. In addition, this work also proposes RaDE+, which is a variant of RaDE, that considers multiple high-effective representative nodes for each dimension instead of single representatives as RaDE. The proposed approach was evaluated on 8 datasets such as a social networks, co-reference networks, textual documents and image datasets. Emphasizing that one of the evaluated datasets is an image dataset of lung x-ray from people with Covid-19, pneumonia or healthy. RaDE and RaDE+ were evaluated for both: visualization and retrieval tasks in this dataset and vector representations generated by both methods achieved better results than the competitor methods in most datasets.

Keywords: Network Representation Learning, Machine Learning, Graphs, Artificial Intelligence.

List of Figures

Figure 1 – General application scenario of NRL approaches.	16
Figure 2 – Semantics related to the embedding dimensions of RaDE+.	20
Figure 3 – Facebook’s user friendship network.	22
Figure 4 – First and second order proximities	25
Figure 5 – Traditional and embedding-based analysis.	29
Figure 6 – Two-dimensional vector representations example	29
Figure 7 – NRL methods classified by the embedding granularity.	30
Figure 8 – Distribution of terms occurrence in generic English texts.	35
Figure 9 – BFS and DFS Random walks comparison.	36
Figure 10 – Overview of the deep learning model used by SDNE.	38
Figure 11 – Main steps of the RaDE algorithm.	46
Figure 12 – Multi-Representative Node Embedding from expansion sets ϕ_{ri}	50
Figure 13 – UMAP visualization on 20NewsGroup dataset.	63
Figure 14 – UMAP visualization on COVIDx dataset.	64
Figure 15 – UMAP visualization on Iris dataset.	65

List of Tables

Table 1 – Popular NRL methods classified by embedding technique	34
Table 2 – Parameter settings used on RADE and RADE+.	57
Table 3 – Specific parameter settings of each NRL baseline method.	57
Table 4 – 20NewsGroup dataset evaluation.	58
Table 5 – Wiki dataset evaluation.	58
Table 6 – Iris dataset evaluation.	59
Table 7 – MPEG-7 dataset evaluation.	59
Table 8 – Flowers dataset evaluation.	59
Table 9 – Corel5k dataset evaluation.	60
Table 10 – COVIDx dataset evaluation.	60
Table 11 – ALOI dataset evaluation.	61
Table 12 – Time evaluation (in seconds) on Iris dataset.	66
Table 13 – Time evaluation (in seconds) on 20NewsGroup dataset.	66

List of abbreviations and acronyms

AAAI	<i>Association for the Advancement of Artificial Intelligence</i>
CNN	<i>Convolutional Neural Network</i>
CSC	<i>Compressed Sparse Columns</i>
CSR	<i>Compressed Sparse Rows</i>
DL	<i>Deep Learning</i>
GB	<i>Gigabyte</i>
HOPE	<i>High-Order Proximity preserved Embedding</i>
ICRL	<i>International Conference on Learning Representations</i>
IPM	<i>Information Processing Management</i>
KDD	<i>Knowledge Discovery and Data mining</i>
LE	<i>Laplacian Eigenmaps</i>
LINE	<i>Large-scale Information Network Embedding</i>
LLE	<i>Locally-Linear Embedding</i>
MAP	<i>Mean Average Precision</i>
ML	<i>Machine Learning</i>
NIPS	<i>Neural Information Processing Systems</i>
NRL	<i>Network Representation Learning</i>
OS	<i>Operating System</i>
PCA	<i>Principal Component Analysis</i>
RAM	<i>Random-Access Memory</i>
RaDE	<i>Rank Diffusion Embedding</i>
SDNE	<i>Structural Deep Network Embedding</i>
VISAPP	<i>International Conference on Computer Vision Theory and Applications</i>
TF-IDF	<i>Term Frequency-Inverse Document Frequency</i>
UMAP	<i>Uniform Manifold Approximation and Projection</i>
t-SNE	<i>t-Distributed Stochastic Neighbor Embedding</i>

Table of Contents

1	Introduction	14
2	Background of Graphs	21
2.1	General Definitions and Notations	21
2.2	Basic Graph Concepts	22
2.2.1	Unweighted graphs	22
2.2.2	Weighted Graphs	23
2.2.3	Undirected Graphs	23
2.2.4	Directed Graphs (Digraph)	23
2.2.5	Complete Graphs	24
2.2.6	Paths in a graph	24
2.2.7	Random Walks	24
2.2.8	First Order Proximity	25
2.2.9	Second Order Proximity	25
2.2.10	Higher Order Proximity	26
3	Network Representation Learning	27
3.1	Overview of Network Representation Learning	27
3.2	Embedding Granularity	30
3.2.1	Node Embedding	30
3.2.2	Edge Embedding	31
3.2.3	Subgraph Embedding	31
3.2.4	Whole-Graph Embedding	31
3.3	Embedding Techniques	32
3.3.1	Matrix Factorization	32
3.3.2	Random Walks	32
3.3.3	Deep Learning	33
3.3.4	Edge Reconstruction	33
3.3.5	Summary of Embedding Techniques	34
3.4	Methods	34

3.4.1	Deep Walk	34
3.4.2	Node2vec	35
3.4.3	LINE: (Large-scale Information Network Embedding)	36
3.4.4	SDNE - <i>Structural Deep Network Embedding</i>	37
3.4.5	HOPE - (<i>Higher Order Proximity preserved Embedding</i>)	39
3.5	Applications	40
3.5.1	Node related Applications	40
3.5.2	Edge Related Applications	40
3.5.3	Graph Related Applications	41
4	A Semantic Rank-Based Graph Embedding Algorithm	42
4.1	Overview	42
4.2	Formal Problem Definition	43
4.2.1	Graph Embedding	43
4.2.2	Rank Model	44
4.3	RaDE (Rank Diffusion Embedding)	45
4.3.1	Rank-based Similarity Graph	45
4.3.2	High-Effective Representative Nodes	47
4.3.3	Single Representative Node Embedding (RaDE)	48
4.3.4	Multi-Representative Node Embedding (RaDE+)	49
4.4	Efficiency, Scalability and Implementaion Aspects	51
5	Experimental Evaluation	52
5.1	Overview of experiments	52
5.2	Datasets	53
5.3	Baseline Algorithms	55
5.4	Evaluation Measures	55
5.4.1	Precision	55
5.4.2	Mean Average Precision (MAP)	56
5.5	Parameter Settings	56
5.6	Retrieval Results	57
5.7	Visualization Results	61
5.8	Efficiency Evaluation	65

6	Conclusions and Future Work	67
	BIBLIOGRAPHY	69

1 Introduction

The representation of connections among elements is a task of central relevance, in many real-world scenarios of diversified domains. In fact, it can be said that every entity in the universe is connected with another in some aspect [1]. In addition, the fascination with the complex *connectedness* of modern society experiments a significant increase over the past decades [1]. Therefore, with the prevalence of network data collected nowadays, from social media to communication or biological networks, learning and effectively representing such connections has become an essential task in many applications [2].

There are many techniques and methodologies currently available [1] for representing entities and their connections. In such scenario, graph-based approaches assumed a central role as an effective and powerful data representation tool. Additionally, effective graph analysis allows a deeper understanding of useful information hidden behind the data. As a result, many important applications can benefit from such analyses and have their effectiveness improved, such as node classification/retrieval, node recommendation, link prediction, among others [3]. Such wide range of applications justifies the significant attention received by graph-based approaches in the last decades [2, 3, 4, 5].

In general, many machine learning applications aim at making predictions or discovering new patterns by exploiting graph-structured data as feature information. Therefore, how to encode the structure of the graph into machine learning models has become a central challenge [5]. In this scenario, although graph-based learning and analysis have emerged as essential tasks, most of traversal-based methods require high computational costs to process graph-structured data [3]. A promising solution consists in *graph embedding* approaches, which have been increasingly exploited due to its capacity of creating vector representations for nodes, edges or even an entire network. Once graph structures can be well represented into a vector space, many mathematical and machine learning techniques available in the literature can be utilized for tasks such as classification, information retrieval, clustering and so forth [6]. Besides that, vector representations generated by graph embedding methods are able to compress huge networks into a significant smaller amount of data while preserving most of original information.

Graph embedding methods usually take into account some of the structural aspects of

the original network for creating the embedding representation. According to [3], the main goal of graph embedding approaches is to represent a graph as low dimensional vectors, while its structures are preserved. As a result, high-effective graph embedding methods are capable of keeping or even improving the accuracy of retrieval and machine learning tasks, even when original network had been significantly compressed. Actually, in some situations, the generated embedding can even improve the accuracy in comparison to the original graph representation.

In such scenario of growing interest, various graph embedding methods have been proposed in the last years [7, 8, 9, 10]. Such techniques are usually categorized in the literature as Network Embedding [6], Network Representation Learning (NRL) [11] or Graph Embedding [4].

The most common strategy used by NRL methods to represent each node in a graph [3] in a low-dimensional vector space, which is also the central idea of this work. The strategy of finding low-dimensional vector representations for nodes in a graph is defined as *node embedding* [3]. There is also other strategies for representing information in graphs defined in literature [4, 3] such as *edge embedding*, *subgraph embedding* and *whole graph embedding*. Each of them will be discussed in more detail in Chapter 3.

For now, let's clarify the central idea of the general usage of NRL methods for analytical tasks. Figure 1 shows an example of the application of various node embedding methods, including *RaDE* and *RaDE+*, which are the proposed methods in this work.

1. *Graph Representation*: The first row of Figure 1 illustrates the task of representing a collection of elements as a similarity-dissimilarity graph, where nodes represent elements and edges represent relations between them. Despite Figure 1 shows a collection of images, other different types of data could be considered (e.g. textual documents, data from social networks);
2. *NRL Methods*: Once the collection is represented as a graph, it is fed as input to a NRL method in order to generate vector representations for that collection. So, in the second row of Figure 1, it is illustrated the application of node embedding methods;
3. *Analytical Tasks*: The third row of the Figure 1, shows each element of the input collection being represented as a vector and being used as input for analytical tasks, which include classification, clustering, information retrieval and many others. One of the great advantages of NRL methods is that the generated representations can be easily applied to methods that

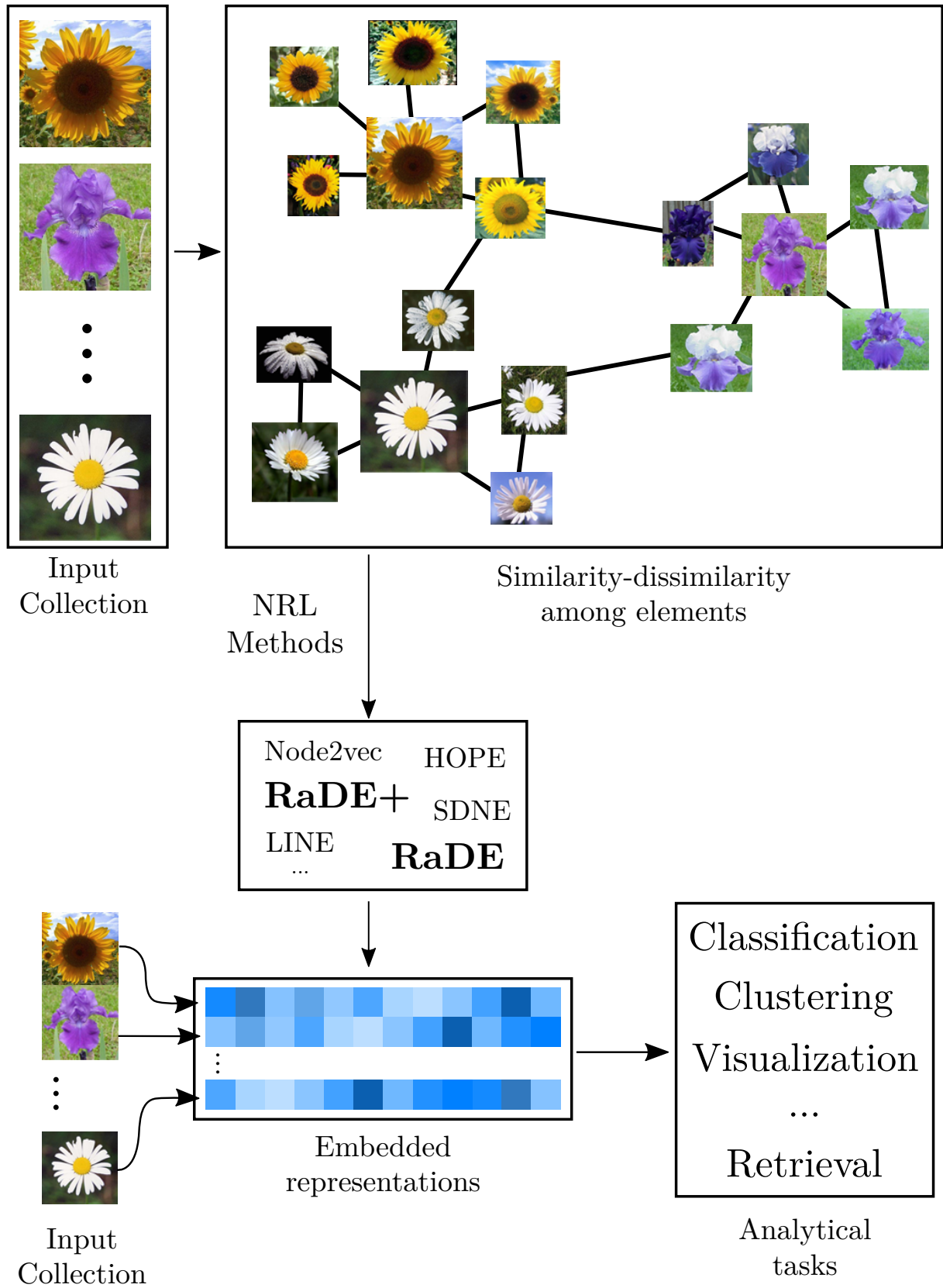


Figure 1 – General application scenario of NRL approaches.

work with vector spaces, so a plethora of methods already available in the literature can be used to perform the task of interest [6].

The necessity of representing information in low-dimensional vector spaces, as illustrated in Figure 1 and described above, emerged in the early 2000s. The main concern at that time was to efficiently represent information in order to spare computational resources to storing it. Therefore, first Network Representation Learning methods evolved [3]. The first developed NRL methods use the *Matrix Factorization* idea [3] as will be discussed shortly next and in more detail in Chapter 3.

Later, in 2010, with the proliferation of graphs in various fields [3], NRL methods started exploiting auxiliary information present in graphs to improve the quality of the embeddings and the concern became not only to store but also to process these graphs. Some of the recently developed NRL techniques are based on *Random Walks* on graphs, some are based on *Deep Learning* models and some others are based on *Edge Reconstruction Optimization* [3]. Next, it is presented a brief overview of each of the main NRL techniques developed along the past years.

1. *Matrix Factorization* methods are the earliest approaches of embedding graph information into low-dimensional vector space. These strategies aims to represent adjacency or weight matrices in terms of vectors that can reconstruct the original matrix. In a recommender system, for example, The adjacency matrix could be encoding the relationship between users and items. The goal of factorization matrix approaches in a scenario like this is to find a vector representation for users and items in a manner that the original matrix can be reconstruct. For doing that, the error between the multiplication of these vectors and the values in the original matrix must be minimized [4]. Some examples of Matrix factorization methods are Laplacian Eigenmaps (LE) [12] and Local Linear Embedding (LLE) [13]. Although these methods result in very effective, embeddings most are inefficient, since they consume a lot of time and space to be processed [3].
2. *Random Walk* methods comes with the goal of finding a compromise between being efficient and effective. They relies on the idea of generating collections of truncated paths in a graph in order to capture the structure and relationships existing between nodes [4]. DeepWalk [14] and Node2vec [8] are popular Random Walk-based approaches that aims to maximize the probability of occurrence of subsequent nodes in these fixed length random walks [4]. For doing that, they take advantage of Skipgram [15], a popular word representation learning technique [3]. This category of method stands out specially in large scale scenarios, where exploring the entire network is unfeasible [6]. The downsides of

random walk based methods are that they only consider a local context with a path and it is hard to find an optimal sampling strategy [3].

3. *Deep Learning* algorithms, such as SDNE [9] use an autoencoder architecture to generate the embeddings. The central idea of these approaches is to preserve the relationships (first and second order proximities) between nodes using an encoder that maps the input data to a low-dimensional vector space. Then, to minimize the error of reconstructing the original vector representations from the low-dimensional representations using a decoder [9, 6]. There is also other deep learning based algorithms such as GCNs (Graph Convolutional Networks) [16] that uses deep convolutional neural networks to learn vector representations for graphs. In general, deep learning algorithms are very effective and robust but there is a high computational cost required to process them [3].
4. *Edge Reconstruction Optimization* methods are based on defining and optimizing objective functions that find vector representations with the goal of preserving some characteristics observed in the original graph [3]. An example of this category of algorithm is LINE [7], which has as main goal to preserve first and second order proximities of nodes. The training of these methods generally is efficient but the downside is that in general, optimization only considers local information.

There is some optimization step required in all of the four categories discussed above [4]. Optimization algorithms are computationally expensive, specially when a large amount of data is considered, even for Random Walk based methods ¹. In addition to the high computational cost required by optimization methods, since the embeddings are generated with a numerical approach, the dimensions of the embeddings do not have a semantic meaning. That is, the dimensions are only numerical values that fits an objective function. This lack of meaning and interpretability of the resulting embeddings by most of the existing methods is a disadvantage, since useful information that could be benefit analysis could be being left aside.

Motivated by the presented problems, this work proposes *RaDE* (*Rank Diffusion Embedding*) [17], a novel graph embedding algorithm, completely unsupervised, for generating low-dimensional representations for nodes in networks. *RaDE* is highly effective and more scalable compared to other NRL methods because it is defined through a rank-based model and no optimization techniques are involved. Besides that, the nature of the rank-based model assigns

¹ Optimization of the Skipgram [15] model.

an intrinsic semantic meaning for the each dimension of the vector representations, making them more informative than that of the competitor methods.

The central idea of *RaDE* consists in identifying high-effective representative nodes of a class or a subset of elements in a collection. In this way, each network node is represented according to its similarity to such representatives. Therefore, nodes with high-similarity to each other are expected to be also similar to same representatives. The representatives are selected based on the density of reciprocal similarity to its neighbors, which operates as an effectiveness estimation of representative rankings.

Besides *RaDE*, this work also proposes *RaDE+* [18], which is a variant of the original approach. The central idea of *RaDE+* consists in expanding each high-effective representative node selected to a set of representatives, denominated expansion sets. In this way, a more robust and diversified representation can be constructed.

A wide and comprehensive experimental evaluation was conducted to assess the effectiveness of generated embedding in retrieval and visualization tasks (see Chapter 5). The evaluation was conducted on 8 diverse datasets considering different data modalities: images, text, social networks, and a traditional pattern recognition dataset. The experimental results are compared with 4 recently proposed node embedding approaches. In various scenarios, the proposed method achieved the best results in most of datasets, demonstrating the ability in generating high-effective representations and being more efficient than other evaluated methods.

The main contributions and novelties of the proposed methods are the following: (i) the ranked-based model used provides an effective manner for representing similarity information among nodes, which is innovative in graph embedding approaches; (ii) no labeled data is required since it is based only on unsupervised estimations; (iii) low computational costs for processing, since the method is based only on ranking information and dispense costly optimization steps often involved in training steps; (iv) there is a degree of semantics associated with each dimension of the generated representations, and this information can be useful for analysis, as shown in Figure 2.

Representatives

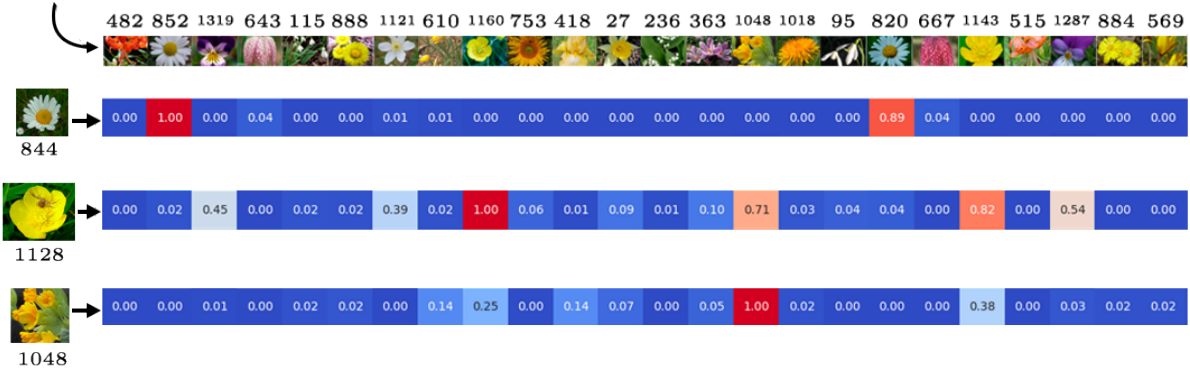


Figure 2 – Degree of semantics associated with each dimension of the resulting embedding representation of RaDE+.

The Figure 2 illustrates three vector representations generated by RaDE on Oxford17Flowers [19] dataset. In this example, it was generated 24-dimensional vector representations and each dimension of the vector representation encodes the similarity of a sample to a high-effective representative element. The set of high-effective representative elements found by the method is presented in the first line of the Figure 2. The other three lines bellow shows the embeddings of samples 844, 1128 and 1048, respectively. It is interesting to notice that the highest values in the dimensions of the vector representations are associated to the most similar representative to the sample analysed. This aspect can assign a degree of semantics to the dimensions of the vector representations and can be useful for inferring the class of a sample, based on the class of the representative, for example.

The remaining of this work is organized as follows: Chapter 2 presents a background of graphs and the notations used along the text; Chapter 3 gives an overview of Representation Learning; the proposed methods are presented on Chapter 4; Chapter 5 discusses the experimental evaluation conducted; finally, Chapter 6 discusses the conclusions, cites the publications and submissions made until the moment and possible future work.

2 Background of Graphs

Graphs are a powerful tool for representing data and provides an elegant and rich manner of encoding relationships existing between data. As the centerpiece of study of this work, this chapter presents a basic background, main definitions and notations of graphs used along this work. Section 2.1 presents general definitions and notations and Section 2.2, gives a background on graphs besides presenting important concepts for the comprehension of further chapters.

2.1 General Definitions and Notations

1. *Graphs, nodes and edges:* A graph or network $G(V, E)$ is defined as a set of elements, called nodes or vertices V , interconnected through connections or edges belonging to the set E [3, 9].
2. *Enumerating nodes:* An arbitrary node belonging to the set V , is written as v_i , i being a generic index. Eventually this notation will be considered with other figures to designate the index of a node v [3];
3. *Enumerating edges:* If a pair of nodes v_i and v_j are connected, $\exists e_{i,j} \in E$ [3];
4. *Quantifiers:* The quantity of elements in the V and E sets are expressed respectively as $|V|$ and $|E|$ [3];
5. *Direct neighborhood:* The set $N(v_i)$ refers to nodes in the direct neighborhood of v_i . Therefore, if $v_j \in N(v_i)$, then $\exists e_{i,j} \in E$ [3];
6. *Node degree:* The degree d_{v_i} of v_i is the number of connections that v_i makes with the other nodes in the network. Therefore, $d_{v_i} = |E \cap \{e_{i,1}, \dots, e_{i,|V|}\}|$ [14];

Figure 3 shows an example of a graph of Facebook user friendships [20], made available by the network repository *Konect* [21]. The set V of nodes in the graph of the Figure 3 is illustrated by the dark blue dots and they represent users. The set E of edges is defined by the lines that connect nodes. If there is an edge connecting a pair of nodes, then these users are friends on Facebook.

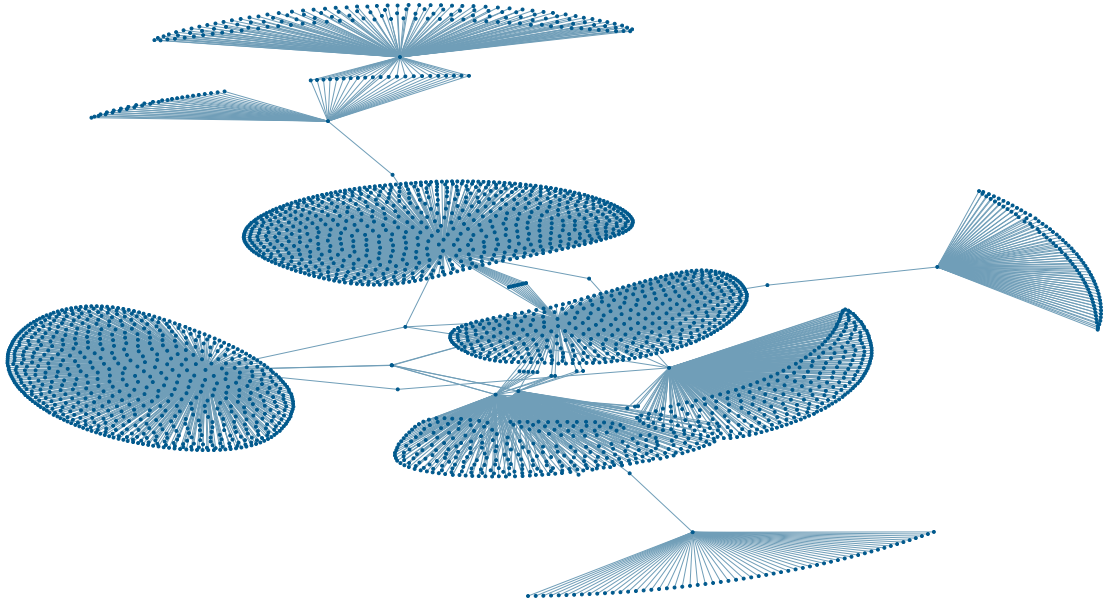


Figure 3 – Facebook's user friendships network [20, 21]. Each node represents a user and each edge indicates the existence of friendship between users.

2.2 Basic Graph Concepts

This section presents a basic graph background and important concepts for the comprehension of further chapters.

2.2.1 Unweighted graphs

Unweighted graphs are those that all connections $e_{i,j} \in E$ have the same importance value $W(e_{i,j})$. In other words, all the connections have the same weight. This kind of graph can be represented as an adjacency matrix [22]. An adjacency matrix A of $G(V, E)$ encodes the existence or not of a connection $e_{i,j}$ between v_i and v_j . Therefore, if and only if $\exists e_{i,j} \in E$, $A_{i,j}$ is a value that represents the existence of a connection between v_i and v_j . A possible way of representing A would be using 0 if there is no edge between v_i and v_j , and 1 otherwise, as illustrated on Equation 2.1.

$$A_{ij} = \begin{cases} 1, & \text{if } \exists e_{ij} \in E \\ 0 & \text{otherwise.} \end{cases} \quad (2.1)$$

2.2.2 Weighted Graphs

A weighted graph $G(V, E)$ has at least one connection e_{ij} which have a different importance value $W(e_{ij})$ than other connections in E [23].

This is formally defined by Equation 2.2.

$$\exists e_{ij} \in E | W(e_{ij}) \neq W(e_{kl}) | e_{ij} \neq e_{kl} \quad (2.2)$$

These weights, depending on the problem, can express the distance between v_i and v_j or the degree of similarity between them. A weighted graph can be represented by weight matrix W that encodes the intensities $W(e_{i,j})$ of the relationships between v_i and v_j . A possible way of representing W would be using 0 if there is no edge between v_i and v_j and real numbers greater than 0, proportionally to the intensity of the connections between v_i and v_j , as shown in the Equation 2.3.

$$W_{ij} = \begin{cases} x \in \mathbb{R} | x > 0, & \text{if } \exists e_{ij} \in E \\ 0 & \text{otherwise.} \end{cases} \quad (2.3)$$

2.2.3 Undirected Graphs

In undirected graphs, all the edges are bidirectional. In other words, if an edge e_{ij} connects v_i and v_j , it also connects v_j and v_i . The adjacency (or weight) matrix of undirected graphs is symmetric, $A_{ij} = A_{ji}$.

2.2.4 Directed Graphs (Digraph)

Edges of directed graphs (also called digraphs) are not bidirectional. That is, the existence of e_{ij} does not implies the existence of e_{ji} . And even if e_{ij} and e_{ji} exist in a directed graph, their semantics are not necessarily the same (e.g.: $W(e_{ij})$ could be different from $W(e_{ji})$). As a consequence, the adjacency (or weight) matrix of directed graphs is not symmetric, $\exists A_{ij} \neq A_{ji} \in A$.

The indegree $d_{v_{in}}$ for a node v_i in a directed graph is the number of edges that depart

from other nodes and have v_i as their destination. This is formally defined in Equation 2.4.

$$d_{viin} = |E \cap e_{1i} \dots e_{|V|i}| \quad (2.4)$$

Outdegree d_{viout} of v_i is defined as the number of edges that start from v_i and have other nodes as their destination, which is formally defined in Equation 2.5.

$$d_{viout} = |E \cap e_{i1} \dots e_{i|V|}| \quad (2.5)$$

2.2.5 Complete Graphs

A graph $G(V, E)$ is called complete when every node v_i is adjacent to all other nodes $v_j \in V$. This is defined by Equation 2.6.

$$\forall i, j \in 1, \dots, |V|, \exists e_{ij} \in E \quad (2.6)$$

As a consequence of this definition, the degree d_{v_i} of all nodes v_i in a complete graph is equal to the total number of nodes $|V|$ in the graph.

2.2.6 Paths in a graph

A path of size k between v_i and v_j in $G(V, E)$ is defined as a sequence $P^k(v_i, v_j)$, that must satisfy the properties below:

- $P^k(v_i, v_j)$ starts on v_i and ends on v_j ;
- $|P^k(v_i, v_k)| = k$;
- $\forall v_q \in P^k(v_i, v_j), v_q \in N(v_{j-1})$

2.2.7 Random Walks

A Random walk T in a graph is defined as a Markovian process [8], which has as principle the selection of a sequence of k nodes, that define a path of size k , starting from an arbitrary node v_i and randomly transitioning to $v_j \in N(v_i)$.

The notation used to define a random walk starting from v_i is T_{v_i} and the sequence of nodes that compose a random walk is the succession of states, based on the random variables $T_{v_i}^1, T_{v_i}^2, \dots, T_{v_i}^k$, such that $T_{v_i}^{k+1} \in N(v_k)$. Random walks can capture information from the network's local community structure in sub-linear time to its total size, in addition of being easily parallelizable [14].

2.2.8 First Order Proximity

First order proximity $P_1(v_i, v_j)$ in $G(V, E)$ is defined as the strength of the connection between pairs of nodes v_i and v_j [7], as formally defined in Equation 2.7.

$$P_1(v_i, v_j) = W_{ij} \text{ if } \exists e_{ij} \in E \quad (2.7)$$

In the Figure 4, the blue line that connects v_6 and v_7 represents the first-order proximity between these two nodes. Since the value in $W_{6,7}$ is high, the first-order proximity between v_6 and v_7 is also high.

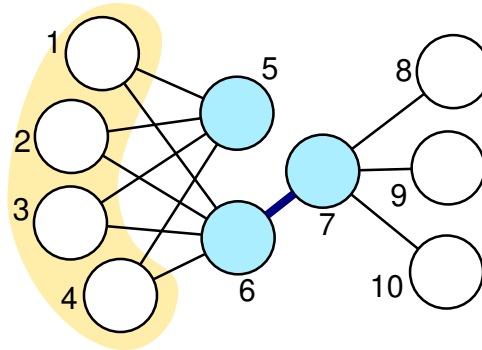


Figure 4 – Generic graph illustrating an example of the difference between first and second order proximity. Adapted from [7].

2.2.9 Second Order Proximity

Second-order proximity $P_2(v_i, v_j)$ refers to the direct neighborhood shared between nodes. Therefore, a node can be treated either as just a node or as a context for other nodes in the graph [7]. Second-order proximity assumes that the degree of similarity between nodes that share the same neighbors, or in other words, that have similar contexts, is high. Equation 2.8

formally defines the second-order proximity between v_i and v_j .

$$P_2 = |N(v_i) \cap N(v_j)| \quad (2.8)$$

In the example of Figure 4, even v_5 and v_6 have a low first-order proximity value (there is no edge connecting these two nodes directly), their second-order proximity is high because they share the same neighborhood (v_1, v_2, v_3 and v_4).

2.2.10 Higher Order Proximity

Similarly to the second-order proximity, higher-order proximity also considers the common neighbors shared by a pair of nodes but instead of considering a 2-step transition (*i.e.* direct neighborhood), higher-order proximity considers a k -step transition ($k \geq 3$). As a consequence, it captures a more global structure of the network [11].

3 Network Representation Learning

Graphs are present in problems in many areas of knowledge. In biology with interactions between proteins, linguistics in problems of co-occurrence of words between documents, in social sciences with friendships network, among other problems of this nature [4]. In several cases, the use of graphs allows a better understanding of the problems as they facilitate the identification of relationships between nodes and analysis of distribution of data through the sub-structures present in the network, assisting the identification of implicit information in the original network [3].

This chapter introduces the main concepts of Network Representation Learning (NRL) area. The remaining sections are organized as following: Section 3.1 presents the motivation and an overview of Network Representation Learning; Section 3.2 presents a categorization of NRL methods according to the granularity of the outputted embedding; Section 3.3 presents a taxonomy of NRL methods according to the technique used for generating the embeddings; Section 3.5 presents the main applications that benefits from NRL; and Section 3.4 discusses about some of popular NRL in literature.

3.1 Overview of Network Representation Learning

Using traditional network representations, specially in large-scale scenarios, leads to several of challenges to process and analyse networks [6]. With traditional network representation approaches, since nodes are related and dependent to each other through a set E of edges, processing and analysing algorithms require a high computational cost. An example is finding the shortest path between two nodes. For doing that, it is necessary to enumerate many possible paths between them. Notice that the complexity involved in this problem is combinatorial [6]. In addition to the difficulty to process data, another bottleneck that the dependency between nodes casts is the difficulty to distribute and parallelize the processing of these algorithms, which is essential in large-scale scenarios [6].

Traditional network representations use adjacency or weight matrices to represent the connections. With that in mind, each node can be represented as a row of this matrix [6]. But representing graphs with this approach causes some limitations, specially in large-scale

scenarios [6]. The bigger the network is, the bigger its adjacency matrix is. Consequently, the number of dimensions of vector representations of nodes will be the same as the total number of nodes in the network. This approach to represent nodes is not efficient because processing and storing these large vector representations require a lot of computational resources and depending on the size of the network, this is impracticable.

Besides that, traditional network representation faces the difficulty to use machine learning (ML) techniques in analytical tasks. In recent years machine learning and deep learning (DL) techniques evolved and have shown effective results in many analytical tasks such as classification, clustering, visualization and others [6]. However, the traditional manner of representing networks (i.e. adjacency matrices) imposes difficulties for applying ML methods in such matrices [6, 4] because adjacency matrices encode nodes in a dependent manner, besides having a high dimensionality.

Therefore, Network Representation Learning (NRL) [6, 3, 4] area has emerged with the purpose of efficiently and effectively representing graphs in low-dimensional vector space, preserving its original structure and characteristics, in order to take advantage of the existing mathematical and machine learning methods developed in recent years. Generally these low-dimensional vectors are generated in a way that they are independent among themselves making it possible to distribute and parallelize the processing of analytical tasks [6]. Besides that, NRL possibilitates a better usage of computational resources to store the vector representations, since they effectively compress huge networks in low-dimensional vector spaces. Formally, NRL approaches can be expressed as a function $f : v_i^d \mapsto v_j^{d'}$ such that $d' \ll d$ [4].

Figure 5 shows a comparison between the analysis of networks based on traditional and embedding representations and illustrates that same analytical task can be applied either using bloated representations (i.e. adjacency matrices) or compact structures (i.e. low-dimensional vectors) that represent the same network in a more efficient manner.

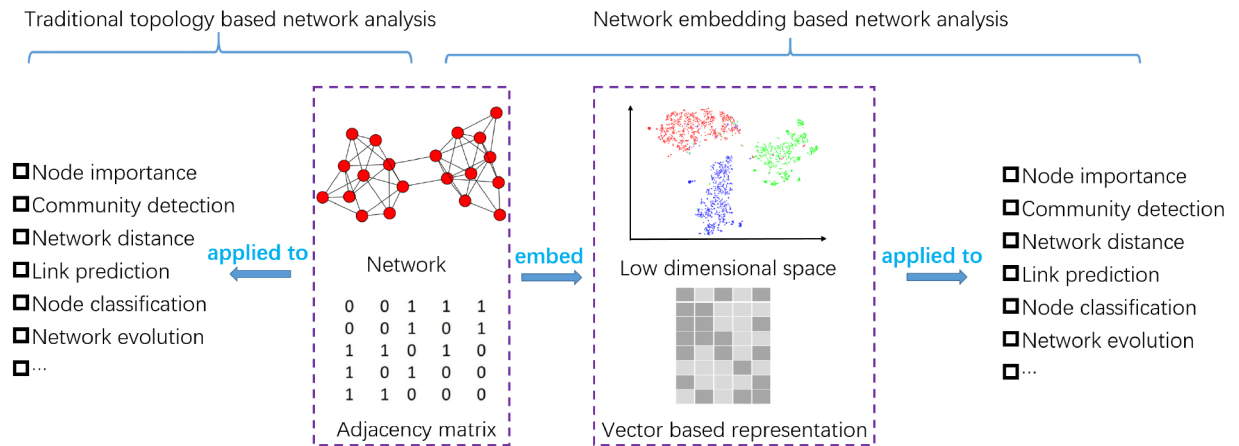


Figure 5 – Comparison between traditional and embedding based analysis of Networks. Extracted from [6].

It is common to associate NRL methods with dimensionality reduction methods, such as PCA [24], t-SNE [25] and UMAP [26]. Although NRL and dimensionality reduction approaches are similar in many aspects, it can be said that while dimensionality reduction methods are better suited for generic data, the application of NRL methods fits better in scenarios that data can be represented as a network.

Figure 6 illustrates an example of the application of Deep Walk [14] to generate a bi-dimensional vector representation for the nodes in the network. Such vector representations are generated in a manner that characteristics in the original network are preserved in the vector representations. In the example of Figure 6 it is possible to notice that nodes that are close in the network are also close in the vector space, that is, the distances between nodes are being preserved in the embedding representation.

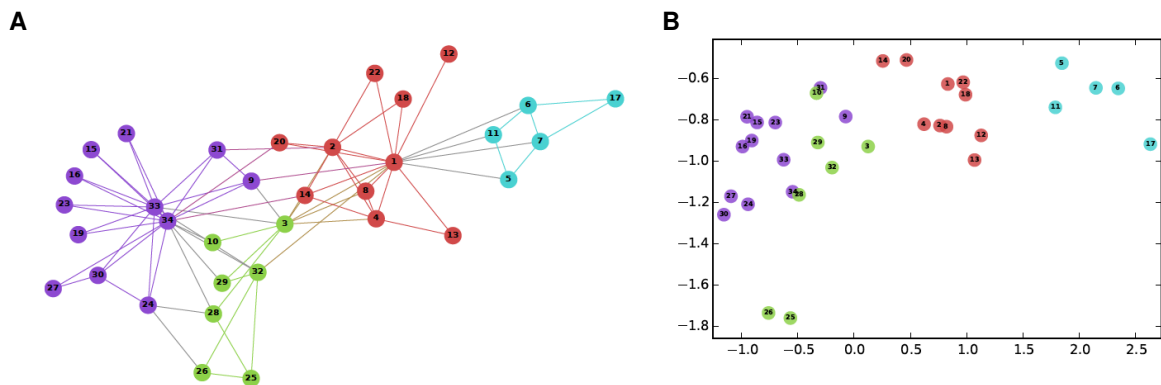


Figure 6 – Example of a two-dimensional vector representation generated from the application of *Deep Walk* for the graph Zachary Karate Club. Extracted from [27].

3.2 Embedding Granularity

Some authors [28, 27] classify NRL methods according to the granularity of the embedding output. The most common categories are the following: (i) node embedding; (ii) edge embedding; (iii) subgraph embedding; and (iv) whole-graph embedding as illustrated in Figure 7.

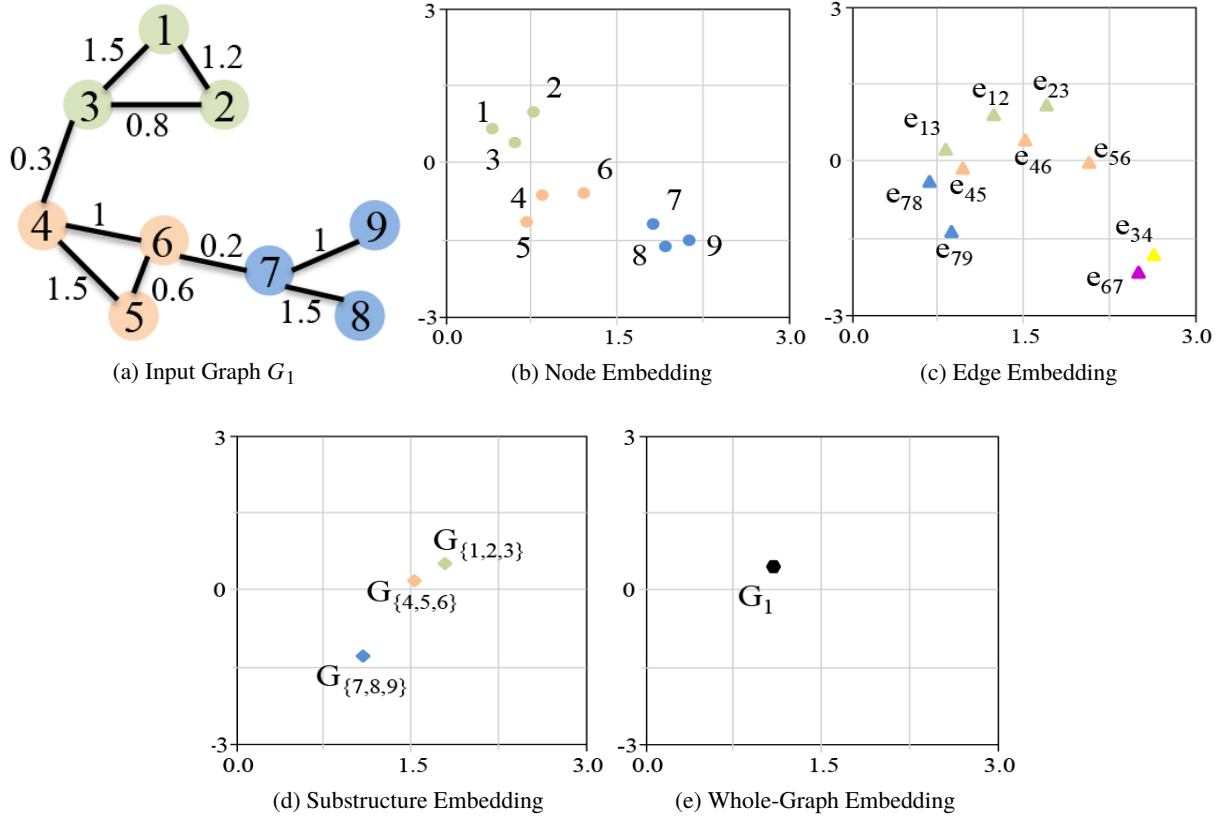


Figure 7 – Categorization of NRL methods according to the embedding granularity. Extracted from [3].

Each granularity level of the output embedding suits better depending on the nature of the application of interest. On one hand, node embedding can benefit node related tasks such as node classification and node clustering. On the other hand, edge embedding, can benefit edge related tasks such as edge recommendation, for example and this same logic applies to other granularity levels of the output embedding [3]. Next subsections discuss about each of the above mentioned granularity levels of embedding.

3.2.1 Node Embedding

Node embedding is the most common embedding strategy used in NRL. Its central idea is to find vector representations for each node in the network in a way that similar nodes in the

graph are positioned close in the embedding space, i.e. have similar vector representations [3]. The differences between node embedding methods relies on how they define closeness of nodes. For example, Tang et al.[7] define closeness using First Order Proximity (See Section 2.2.8) and Second Order Proximity (See Section 2.2.9) metrics. Other authors [9, 10] use Higher Order Proximity metric. In Section 3.5.1 it is discussed about the main scenarios where node embeddings strategies stands out.

3.2.2 Edge Embedding

The goal of edge embedding methods is to represent edges as low-dimensional vectors. These approaches faces different challenges of node embedding approaches do, since the proximity of edges is different than the proximity of nodes, because they encode pairwise node relations. Another difference existing between edges and nodes is that edges may be directed or undirected and this properties should be encoded the vector representations [3]. See applications of edge embedding methods in Section 3.5.2.

3.2.3 Subgraph Embedding

This granularity level of embedding output aims to find low-dimensional vector representations for different graph components, such as substructures (i.e. connected compoents) and node + community. As mentioned in [3], subgraph embedding has been studied in various works. Liu et al.[29] embeds the community existing between two distant nodes to support semantic proximity, [30] learns subgraphs to help classification task and in [31], to enrich information in question answering applications, it was embedded the paths and subgraphs from question entity to the answer entity, in a knowledge base [3], Wang et al.[32] proposes community-aware proximity for node embedding, in a way that the community embedding can reinforce the node embeddings [3]. However, this category of embedding output faces diverse challenges such as, how to define the substructures that will be embedded and how to embed different types of substructues in the same vector space, which leads to a heterogeneity problem [3].

3.2.4 Whole-Graph Embedding

This type of embedding aims to represent a whole graph, usually small graphs, such as proteins and molecules in a vector space. Whole-graph embedding provides an efficient and straightforward way to classify and compute similarities among graphs [3]. The first challenge of

whole-graph embedding is to capture the all properties of the graph and the second one is to establish a trade-off between efficiency and the expressiveness of the embedding. Mousavi et al.[33] proposes a framework to compute the whole-graph embedding hierarchically. The idea, is to mount a pyramid where each level is a subgraph. Then, the embeddings for each subgraph is computed and concatenated into one single vector [3]. Li et al.[34] learns the embeddings for the whole cascade graph and trains a multi-layer perceptron to predict the increment of the size of the cascade graph in the future [3].

3.3 Embedding Techniques

In addition to the categorization according to the granularity of the embedding, as discussed in Section 3.2, some authors [3, 4] define a taxonomy of NRL methods, according to the technique used to generate the embeddings. There is not a definitive convention for this taxonomy in literature but most important categories are: matrix factorization, random walks, deep learning and edge reconstruction [6, 4, 3]. The following subsections discuss about each of these embedding techniques.

3.3.1 Matrix Factorization

Matrix factorization methods represent a graph as a matrix, and factorize this matrix to obtain the embedding [4]. These matrices can be of different types such as Laplacian matrices, node transition probabilities matrices, Katz similarity matrices, among others and the factorization strategy varies based on the matrix properties. The idea in factorization matrix methods is to compute vectors that encode the information present in the original matrix and are able to reconstruct it when multiplied. Cai, Zheng e Chang[3] highlights two main types of matrix factorization techniques: *Graph Laplacian Eigenmaps* and *Node Proximity Matrix*.

3.3.2 Random Walks

Random walks methods are useful specially in large scale graphs, which exploring the whole graph is impracticable [4]. Random walk methods [8, 14] have shown good results on efficiently e effectively approximate node centrality and node similarity.

The central idea of this strategy is to represent a graph as a set of truncated random walks paths sampled from it. Then, to use these random walks to generate the embeddings.

DeepWalk [14] and Node2vec [8], for example, treat these truncated random walks similarly to sentences, where each node is analogous to a term. Then, they borrow the idea of natural language processing models, such as SkipGram [15], to learn the vector representations.

3.3.3 Deep Learning

Deep learning models have evolved in recent years and due to good results in a wide range of researching areas, it naturally started to be applied in NRL.

A popular application of Deep learning models in NRL is the usage of autoencoder models. Autoencoders have the purpose of converting the input data to a low-dimensional vector space by minimizing the error of reconstructing the input data from encoded vector space (See Equation 3.3). The central idea of using autoencoder models is to preserve the neighborhood of nodes. Thus, for nodes with similar neighborhoods, the reconstruction process will result in similar vectors [3].

SDNE [9] and DNGR [35] proved that autoencoder models are robust solution and can capture highly non-linear structures in networks. Since the training process of autoencoder models can be computationally expensive [4], specially in large-scale scenarios, recent works [36, 37, 38] propose the usage of convolutional neural networks to learn graph low-dimensional vector representations. GCNs (Graph Convolutional Networks) [16] proposes a model that iteratively learns the embedding of a node and its neighbors [4].

3.3.4 Edge Reconstruction

Edge Reconstruction-based approaches aim to generate embeddings such that the edges derived from the embedded representation should be as similar as possible to those observed in the input graph [3]. For doing that, these techniques directly optimize objective functions that either maximize the edge reconstruction probability, minimize the edge reconstruction distance-based loss or minimize edge reconstruction margin-based ranking loss [3]. Some of popular methods that are based on edge reconstruction are LINE [7], TransE [39], TransR [40] and TransD [41].

3.3.5 Summary of Embedding Techniques

This subsection presents a summary of main NRL methods classified according to the technique used to generate the embeddings and the properties of the original network that is aimed to be preserved. This summary is presented in Table 1.

Technique	Year	Published	Method	Property preserved
Matrix Factorization	2000	Science [13]	LLE	1 st order proximity
	2001	NIPS [12]	LE	
	2016	KDD [10]	HOPE	1 – k^{th} order proximity
Random Walks	2014	KDD [14]	Deep Walk	
	2016	KDD [8]	node2vec	1 – k^{th} order proximities, structural equivalence
Deep Learning	2016	KDD [9]	SDNE	1 st and 2 nd order proximities
	2016	AAAI [35]	DNGR	1 – k^{th} order proximity
	2017	ICLR [16]	GCN	
Edge Reconstruction	2015	WWW [7]	LINE	1 st and 2 nd order proximities

Table 1 – Popular NRL methods classified according to the technique used for embedding information on the original network. Adapted from [4]

3.4 Methods

Some of the most popular NRL methods in the literature are presented in the following subsections. The motivation to present only these methods in this work is because they were used as baselines ¹ in Experimental Section. See Chapter 5.

3.4.1 Deep Walk

Deep Walk [14] is a unsupervised node embedding and random walk based method that implicitly preserves higher order proximity of nodes [4]. It is based on the premise that the distribution of nodes degree follows the power law [42]. When this property is observed in a graph, it is called a free network scale [43]. On free of scale networks, few nodes have a high degree, while many nodes have a low degree. As a consequence, the frequency of occurrence of nodes, in short random-walks, in this type of network, also follows a law of power.

¹ DeepWalk was presented for a better understanding of node2vec, since node2vec is a generalization of DeepWalk. Experimentally, only node2vec was evaluated.

In 1994, Zipf [44] showed that the frequency of occurrence of terms in texts follows the power law. Therefore, the idea of Deep Walk is to exploit the similarity between the words occurrence in texts and the frequency of occurrence of nodes in short random walks, treating these random walks as sentences.

Figure 8 shows that the distribution of the frequency of occurrence of terms in texts written in English on a log-log scale graph follows the power law.

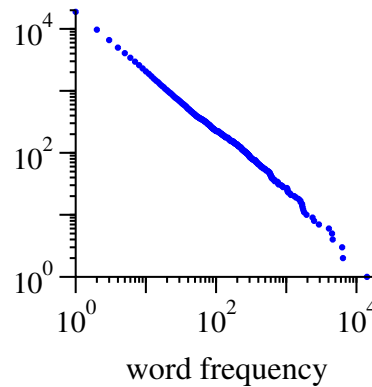


Figure 8 – Distribution of the frequency of occurrence of terms in generic texts written in English. Extracted from [42].

After performing random walks for each node in the graph, a further step of Deep Walk is to use SkipGram [15], which is a natural processing language model that can learn vector representations for terms in sentences. Since each random walk is analogous to a sentence and each node in these random walks is analogous to a term in a sentence, SkipGram and other NLP models can be used to learn vector representations for nodes.

Since Deep Walk is based on random walks, it is easily parallelizable. That is, different parts of the graph can be explored simultaneously. In addition of being parallelizable, the property of local exploration that random walks have, makes it possible to accommodate small changes in the graph without the need for global recalculation. This is, the model learned can be updated iteratively from new random walks starting from the region of the graph that has been changed [14].

3.4.2 Node2vec

Node2vec [8] is a generalization of DeepWalk [14]. Therefore it also is a node embedding method random walk-based. Its crucial difference from DeepWalk is that Node2vec introduces parameters that generates biased-random walks. These biased-random walks considers breadth-

first and depth-first graph search. While *breadth first* random walks visit the largest number of neighbors in one node, emphasizing local characteristics of the graph, *depth first* random walks visit a greater number of nodes in the graph as a whole, and the vector representations preserve more global aspects of the graph. As a consequence, more informative and higher-quality embeddings than DeepWalk are generated [4]. Figure 9 illustrates the comparison of random walks in depth and width, highlighted in blue and red, respectively.

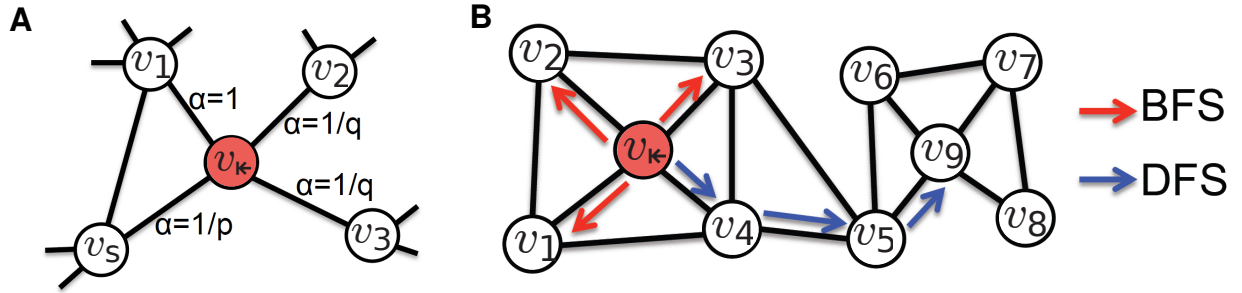


Figure 9 – Comparison of Random Walks in depth highlighted in blue or width highlighted in red, depending on the parameters p and q . Extracted from [27].

3.4.3 LINE: (Large-scale Information Network Embedding)

LINE [7] is an efficient and scalable NRL method based on edge reconstruction. Its central idea is to optimize a mathematical model that aims to preserve first and second order proximities of nodes in a network. This mathematical model is based on the combination of two objective functions: one for first-order proximity and other for the second-order proximity. Equation 3.1 presents the objective function for the first-order proximity [7].

$$\begin{aligned}
 p_1(v_i, v_j) &= \frac{1}{1 + e^{(-\vec{u}_i \cdot \vec{u}_j)}} \\
 O_1 &= d(\hat{p}_1(., .), p_1(., .)) \\
 O_1 &= - \sum_{(i,j) \in E} w_{ij} \log p_1(v_i, v_j)
 \end{aligned} \tag{3.1}$$

The first row of Equation 3.1 defines a distribution $p(., .)$ over the space $V \times V$, $\vec{u}_i \in \mathbb{R}^d$ is the low-dimensional vector representation of v_i and its empirical probability $\hat{p}_1(i, j) = \frac{w_{ij}}{W}$, where $W = \sum_{(i,j) \in E} w_{ij}$. The second row of Equation 3.1 defines the distance between the two probability distributions and the third defines an objective function to preserve first-order proximity, replacing $d(., .)$ with KL-divergence. By finding $\vec{u}_{i=1, \dots, |V|}$ that minimizes the third row in the Equation 3.1, each node can be represented in a low dimensional vector space [7].

Tang et al.[7] applies the same idea presented in Equation 3.1 to define an objective function that generate vector representations for nodes that preserves the second order proximity. Then concatenates the results for generating the final vector representations.

The optimization of these functions is computationally expensive, so besides the model for finding representing nodes as low-dimensional vectors, Tang et al.[7] also proposes an optimization strategy based on *edge sampling*, which makes the time complexity of LINE $O(|E|d)$, that is, linear to the number of edges $|E|$ times the number of dimensions d [4].

3.4.4 SDNE - *Structural Deep Network Embedding*

As addressed in [45], the underlying structure of networks is highly non-linear. Since deep learning models show success in capturing highly non-linear informations in data. In order capture these highly non-linear structures present in networks, Wang, Cui e Zhu[9] proposes SDNE, which is a semi-supervised node embedding method based on an autoencoder [46] deep learning model.

SDNE tackles the three main problems: *(i)* capturing highly non-linearity structures in networks; *(ii)* preserving the original network structures; *(iii)* solving the sparsity problem [14] present in real world networks.

Autoencoders are unsupervised models composed by two main parts, i.e. the encoder and decoder. The role of encoder is to map the input data to a representation space, while the decoder map the vectors in representation space to reconstruction space. Both encoder and decoder use multiple non-linear functions in order to achieve the objective of mapping data to representation and reconstruction spaces [9]. Figure 10 gives an overview of the deep model used by SDNE.

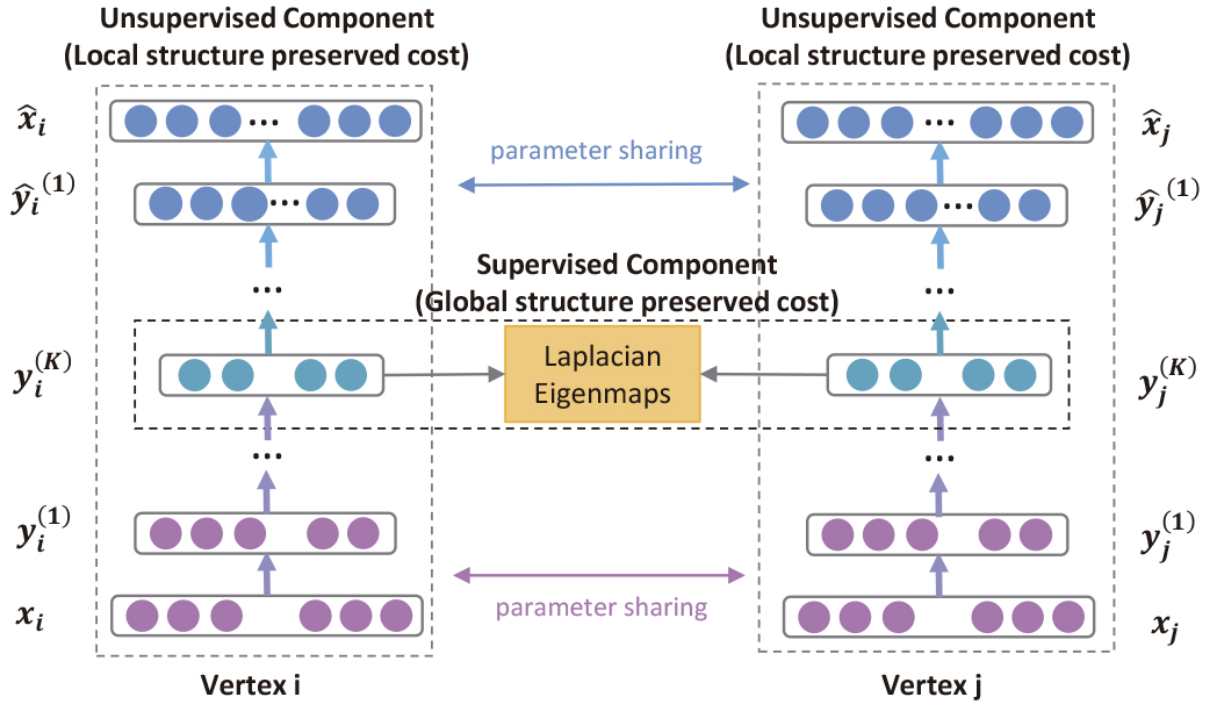


Figure 10 – Overview of the deep learning model used by SDNE. Extracted from [9].

So, for an input x_i , the hidden representations for each layer is shown in Equation 3.2.

$$\begin{aligned} y_i^{(1)} &= \sigma(W^{(1)}x_i + b^{(1)}) \\ y_i^{(k)} &= \sigma(W^{(k)}y_i^{(k-1)} + b^{(k)}), k = 2, \dots, K \end{aligned} \quad (3.2)$$

In Equation 3.2, $\sigma(x)$ is a non-linear function. Wang, Cui e Zhu[9] used a sigmoid function; $W^{(k)}$ is the k - th layer weight matrix, $b^{(k)}$ is the k - th layer biases, x_i is an input vector [9].

Since $y_i^{(k)}$ is calculated, it is also possible to reverse the calculation process of the encoder to obtain $\hat{x}_i$, which is an approximation of the input x_i . So, the main goal of the autoencoder is to minimize the reconstruction error of the input and output. The loss function $\mathcal{L}$ for the reconstruction is illustrated on Equation 3.3 [9].

$$\mathcal{L} = \sum_{i=1}^n \|\hat{x}_i - x_i\|_2^2 \quad (3.3)$$

Considering that the input for SDNE is the adjacency matrix S , each row s_i of S characterizes the neighborhood structure of a node v_i . With that in mind, using the reconstruction process described in Equation 3.3 directly from S faces two main problems: (i) there are many legit edges not observed in networks. The existence of edges indicates similarity between nodes

but its nonexistence not necessarily indicates dissimilarity [3]; (ii) real world networks are generally very sparse. So it is more prone to reconstruct the zero elements.

In order to address these problems, Wang, Cui e Zhu[9] proposes two objective functions to preserve first and second order proximities jointly in learning process, as shown respectively in Equations 3.4 and Equation 3.5.

$$\begin{aligned}\mathcal{L}_{1st} &= \sum_{i,j=1}^n s_{i,j} \|y_i^{(K)} - y_j^K\|_2^2 \\ &= \sum_{i,j=1}^n s_{i,j} \|y_i - y_j\|_2^2\end{aligned}\tag{3.4}$$

In Equation 3.4, $s_{i,j}$ is an entry of the adjacency matrix S . Its central idea is to use the principle of Laplacian Eigenmaps [12] that penalizes vertices mapped far away in the embedding space.

$$\begin{aligned}\mathcal{L}_{2nd} &= \sum_{i=1}^n \|(\hat{x} - x_i) \odot b_i\|_2^2 \\ &= \|(\hat{X} - X) \odot B\|_F^2\end{aligned}\tag{3.5}$$

In the first row of Equation 3.5, $\odot$ means the Hadamard product, $\mathbf{b}_i = \{b_{i,j}\}_{j=1}^n$. If $s_{i,j} = 0$, $b_{i,j} = 1$, else $b_{i,j} = \beta > 1$; $\hat{X}$ is the reconstructed matrix of the input data, while X is the matrix of the input data [9]. Equation 3.5 guarantee that the vertices with similar neighborhood will be mapped near in embedding space.

3.4.5 HOPE - (*Higher Order Proximity preserved Embedding*)

HOPE [10] is a factorization matrix based embedding method and follows the strategy of the Laplacian Eigenmaps method [12] to preserve the relative distance between nodes, based on the scalar product (angle between vectors) in the new vector space [10].

This method aims to preserve asymmetric transivities in directed graphs. In non-directed graphs, transivities are symmetrical. That is, given that there is an edge e_{ij} and e_{jk} , then nodes v_i and v_k are likely to be related in some way. For directed graphs, transivities are asymmetric. That is, if there is e_{ij} and e_{jk} , v_i and v_k can be related but nothing can be stated about the relationship between v_k and v_i .

Considering the aforementioned restrictions, HOPE aims to find a representative vector for each node of a graph, so that the relative distances of the asymmetric transivities of the vector

representations learned are minimum in relation to the distances of the asymmetric transivities of the vectors real.

The equation 3.6 illustrates the general form of the function that should be optimized to preserve asymmetric transivities denoted K , which refers to the Katz proximity [10]; S is equal to the real relative asymmetric transivities matrix and Z is the matrix of the representations learned for each node [10, 4].

$$K \approx \|Z^T Z - S\|^2 \quad (3.6)$$

3.5 Applications

A wide variety of applications use graphs as base foundation. Depending on the nature of the application, different analytical tasks are required. Some of them are more related to nodes, while some others are more related to edges or the whole graph.

This section presents a categorization of NRL applications divided into node related, edge related and graph related applications.

3.5.1 Node related Applications

- *Node Classification*: Based on vector representations learned for nodes in a graph and their respective original labels, the goal of this application is to predict the class of a node, whose label is not known. Some applications of this task are: classifying textual documents, videos, pages of the web or individuals in different categories/communities [27].
- *Node Clustering*: As with the visualization task, the vector representations for the nodes of a graph enable techniques generic clustering like *k-means* [47] and DB-scan [48] to be applied to the node vectors [27].

3.5.2 Edge Related Applications

- *Link Prediction*: Predicting links from vector representations learned consists in identifying relationships that are missing or are likely to occur in the future [4] in the graph. Examples of link prediction applications are suggestions for friendships on social networks and recommending system [4].

3.5.3 Graph Related Applications

- *Visualization:* Visualizing graphs in a bi-dimensional vector space is a challenging problem because graphs do not have any prior information on the spatial arrangement of nodes. The allocation of a vector representation for nodes using NRL approaches allows techniques used for visualizing data sets with many dimensions, such as PCA [24] and t-SNE [25] can be used to generate views of graphs and assist in the discovery of communities or implicit structures [27].
- *Network compression:* Given $G(V, E)$, this task consists of generating a new graph $G * (V', E')$, such that $|E'| < |E|$, that is, the quantity of edges in the new graph is smaller than the quantity in the original one and still being as descriptive as the original graph. The main goal of this task is to store information performing graph analysis algorithms more efficiently than in the original graph [27].

4 A Semantic Rank-Based Graph Embedding Algorithm

Most Graph Embedding methods require some computationally expensive optimization step [3]. Besides the high computational cost required in optimization methods, the dimensions of the resulting embeddings do not have interpretability, since these are computed with the only goal of satisfying an objective function that aims to preserve properties observed in the original graph [7, 9, 8, 3], thus, generally these representations do not encode a semantic meaning, which is a downside of most embedding methods, since useful information for analysis is being left aside.

In order to address such problems, this chapter presents RaDE (Rank Diffusion Embedding), which is an efficient graph embedding approach, since it is not based on optimization strategies, besides generating effective and semantically reinforced embeddings.

This chapter is divided into four main sections: Section 4.1 gives an overview and the key ideas of the proposed approaches; Section 4.2 presents the notations used and a formal problem definition; Section 4.3 shows the proposed methods in more depth as well as its formal definition; and in Section 4.4, are discussed the aspects of efficiency, scalability and implementation aspects of the proposed approaches.

4.1 Overview

Rank-based approaches have been successfully used in diverse retrieval and machine learning tasks recently [49, 50], mainly due to its capacity of encoding relevant similarity information defined in relationships among dataset elements. Such capacity is exploited in this work in order to embed the nodes from a similarity graph into a vector space, while maintaining similarity and neighborhood relationships. The used rank-based model allows an effective similarity representation, since the most relevant information are located at top rank positions.

Gradient based optimizations are generally computationally expensive. Since most of graph embedding competitor approaches are optimization-based, it can be difficult to apply them in large-scale scenarios [4, 3]. One of main benefits of the proposed approach is that it

does not relies on optimization techniques to generate vector representations. As consequence, it makes RaDE easily to scale and more efficient in relation to optimization-based ones. RaDE generate richer embeddings than its competitors because it encodes similarity information among each element and the most representative nodes in the graph in each dimension of the resulting representation. In addition, the proposed method is fully unsupervised and independent of the domain of data.

The central idea of RaDE is to select high-effective representative nodes in a pairwise similarity-dissimilarity collection of elements. The selection of the high-effective representatives is done by propagating the affinities among elements encoded in the dataset through a diffusion process [51, 52]. The hypothesis of RaDE is that elements with highest reciprocal affinities are likely to be good representatives. Then, in order to capture diversity of good representatives in the dataset, RaDE calculates iteratively a set $\mathcal{R}$ with the d highest-effective representative nodes in a manner that they have a high reciprocal affinity score and at the same time a low similarity value to the already selected elements in $\mathcal{R}$. The final hypothesis of RaDE is that each element in the collection can be well represented in terms of its similarity to good representatives ones in $\mathcal{R}$.

4.2 Formal Problem Definition

This section provides the formal definitions and notations for the concepts used by the method proposed in this work. Section 4.2.1 defines the task of *graph embedding*; and Section 4.2.2 gives the definition of the *rank model* used.

4.2.1 Graph Embedding

Let $C = \{e_1, e_2, \dots, e_n\}$ be a collection of data elements, where $n = |C|$ denotes the size of C . Along this work, the notation is relaxed in such a way that an element $e \in C$ can be used either as the element itself or its index, depending on context.

The collection C can be represented by a graph $G(V, E)$, with $V = C$ as the set of vertices and $E \subseteq V^2$ as the set of edges. If $(e_i, e_j) \in E$, we say that vertices which represents elements e_i and e_j are connected in the graph. Weights can be assigned to edges, commonly represented by an adjacency matrix $\mathbf{S}$ such that the value assigned to the edge (e_i, e_j) is given by $s_{i,j}$.

We define a *graph embedding* task (more specifically, *node embedding*) similarly to [4]. Given a graph $G(V, E)$, graph embedding can be seen as a mapping function $f : V \rightarrow \mathbb{R}^d$ with

the following characteristics:

- the number of dimensions is much smaller than the size of the collection, i.e., $d \ll |V|$;
- f preserves some structural information of G .

More specifically, in regards to the latter, it is expected that the similarity information encoded in the graph G is preserved, such that similar nodes in the graph are projected close to each other in the $\mathbb{R}^d$ space.

4.2.2 Rank Model

As the proposed method is defined in terms of ranking information, this section presents a formal definition of the ranking model considered along this work.

Ranking tasks are often defined through pairwise dissimilarity measures. Such that a measure between two elements e_i and e_j is denoted by $\rho(i, j)$.

Let e_q be a query element. A ranked list τ_q can be computed in response to e_q , in which the top positions of τ_q are expected to contain the elements most similar to e_q . As such, τ_q is sorted by the values of ρ in ascending order, which means that a full ranked list might be expensive to compute, specially when n is high. Therefore, given that the most similar (and relevant) elements are in the initial portion of the rank for e_q , a more computationally efficient approach for generating a ranked list is to compute only the first L elements of the full rank. We now follow with a definition for τ_q with that restriction.

Let τ_q be a ranked list that contains only the L elements most similar to e_q , where $L \ll n$. Formally, let C_L be a sub-set of C , such that $|C_L| = L$ and $\forall e \in C_L, \forall e' \in C \setminus C_L, \rho(e_q, e) \leq \rho(e_q, e')$. The ranked list τ_q can then be defined as a bijection from the set C_L onto the set $[L] = \{1, 2, \dots, L\}$, such that $\forall e_i, e_j \in C, \tau_q(e_i) < \tau_q(e_j) \iff \rho(e_q, e_i) < \rho(e_q, e_j)$.

Every element $e_i \in C$ can be taken as a query element e_q . As a result, a set of ranked lists $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_n\}$ can be obtained, with a ranked list for each element in the collection C . The set $\mathcal{T}$ constitutes a rich source of similarity information about the dataset. Such information is exploited to compute the embedding.

4.3 RaDE (Rank Diffusion Embedding)

Given a graph with edge weights assigned by similarity/dissimilarity measures¹, an intermediary graph representation based only on ranking information is derived. Next, RaDE exploits this graph to learn a novel vector representation based on two conjectures: (i) high-effective representative nodes can be identified by analysing the rank-based graph; (ii) each node can be represented according to its similarity to a set of representative nodes. In this way, the method can be computed through three main steps:

- A. Rank-based Similarity Graph: the input graph is represented through a sparse rank-based graph constrained to top similar nodes;
- B. High-Effective Representative Nodes: reciprocal affinities in the graph are analysed to select high-representative nodes;
- C. Node Embedding: the node embeddings are computed based on the similarity to the high-representative nodes.

An overview of these steps is illustrated in Figure 11 and each of them is described formally and in more detail in next sub-sections. Section 4.3.1 presents the Rank-based Similarity Graph, while Section 4.3.2 discusses the strategy for selecting High-Effective Representative Nodes. Two embedding approaches are presented on Sections 4.3.3 and 4.3.4: considering Single (*RaDE*) and Multi-Representative Node Embeddings (*RaDE+*).

4.3.1 Rank-based Similarity Graph

Various retrieval and machine learning approaches define a similarity matrix $\mathbf{W}$ that represents a graph based on a dissimilarity measure ρ . A Gaussian kernel is often considered, such that $w_{ij} = \exp(\frac{-\rho^2(i,j)}{2\sigma^2})$, where σ is a parameter to be defined.

Inspired by [50, 53], we define a rank similarity matrix $\mathbf{W}$ based only on rank information. The similarity score w_{ij} varies according to the position of e_j in the ranked list τ_i (i.e., $\tau_i(e_j)$).

¹ Note that a dataset that is not originally represented as a graph might be viewed as a complete graph applicable to the method as long as there is a similarity/dissimilarity measure between its elements.

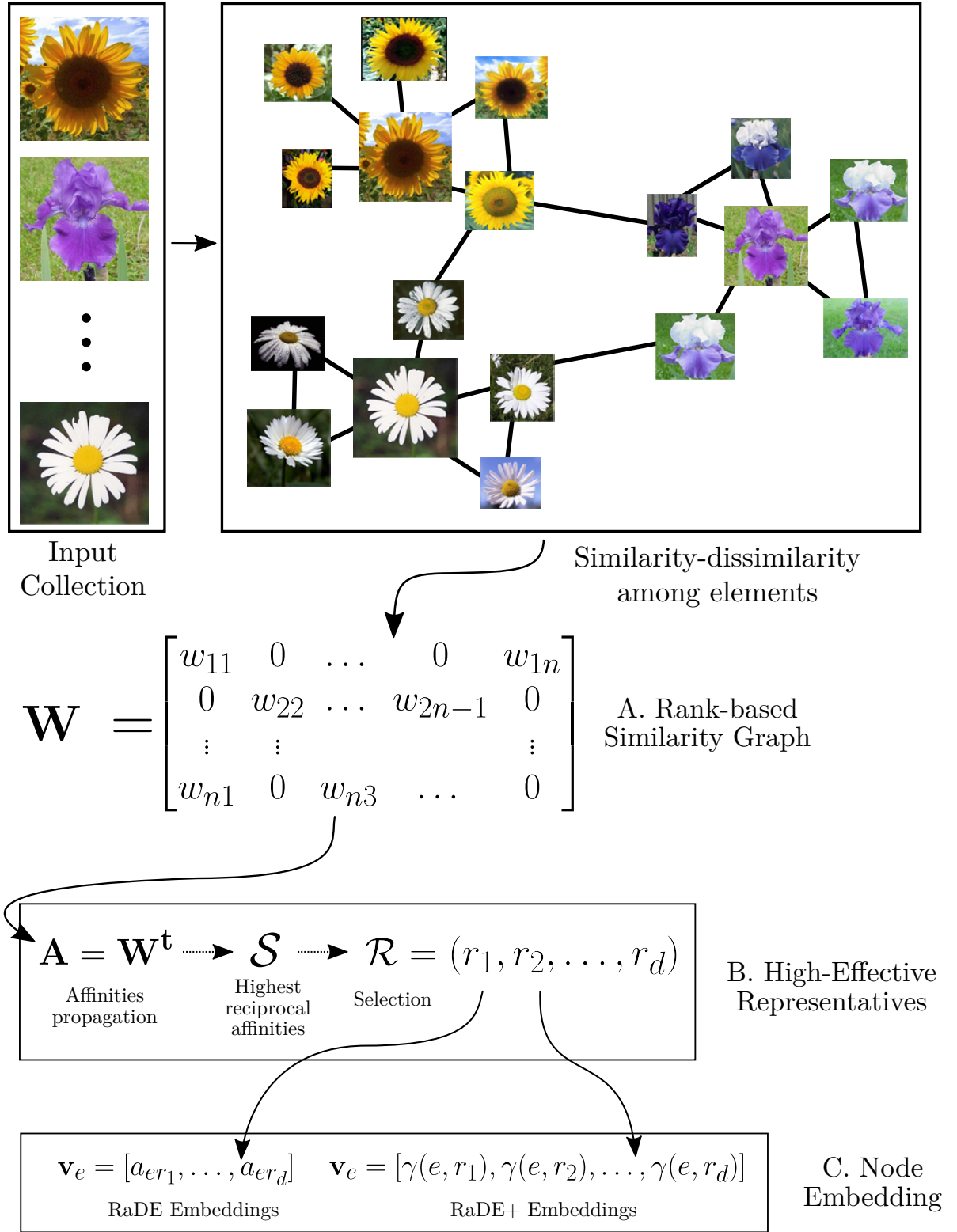


Figure 11 – Main steps of the RaDE algorithm.

Additionally, the score considers only a neighborhood set, which is limited by the size L of the

ranked lists. Thus, the element w_{ij} of $\mathbf{W}$ is defined as:

$$w_{ij} = \begin{cases} 1 - \log_L \tau_i(e_j), & \text{if } \tau_i(e_j) \text{ is defined} \\ 0 & \text{otherwise.} \end{cases} \quad (4.1)$$

L can assume different values depending on the desired analysis. In the proposed method, the matrix $\mathbf{W}$ is defined assuming $L \ll n$ and, since it has dimension of $n \times n$, $\mathbf{W}$ is very sparse.

4.3.2 High-Effective Representative Nodes

The proposed approach relies on determining the most representative nodes in a graph for generating an embedding representation. The guiding hypothesis of RaDE is that a good representative node has high affinity with nearest nodes and low affinity with distant nodes.

First, we compute contextual affinities, which take into account the structure of dataset manifold. The key idea is based on diffusion process methods [51, 52], which propagate affinities encoded in $\mathbf{W}$. The more global similarity measures can be obtained by powers of $\mathbf{W}$, as shown in Equation 4.2:

$$\mathbf{A} = \mathbf{W}^t, \quad (4.2)$$

where t is a constant that defines the number of iterations. A small value of $t = 2$ was used in all experiments.

Note that $\mathbf{A}$'s diagonal values represent reciprocal affinities. For example, for $t = 2$, $a_{ii} = \sum_{j=1}^n w_{ij}w_{ji}$, which means that d_{ii} aggregates direct reciprocal affinities between e_i and all the other elements. For $t > 2$, the diagonal still encodes reciprocal affinity, but indirectly.

We use the diagonal to find a set of k high-effective representative candidates, namely $\mathcal{S}$. The items in this set are supposed to satisfy the first requirement of the guiding hypothesis, thus this set is composed by the k elements with the highest reciprocal affinities. Elements with high reciprocal similarities are expected to have high-effective ranked lists, and therefore be good candidates.

Formally, the candidates set $\mathcal{S}$ must hold the following properties:

$$\mathcal{S} \subseteq \mathcal{C} \quad (4.3)$$

$$|\mathcal{S}| = k \quad (4.4)$$

$$\forall e \in \mathcal{S}, e' \in \mathcal{C} \setminus \mathcal{S}, a_{ee} \geq a_{e'e'} \quad (4.5)$$

Finally, a resulting ordered list of d representative nodes is obtained from the set of candidates $\mathcal{S}$. The objective is to select high-effective candidates with the maximum of diversity among them. In this way, both the diagonal affinity scores and the affinities to already selected nodes are considered. Let $\mathcal{R} = (r_1, r_2, \dots, r_d)$ be an ordered list with the d most effective representative nodes, each element r_i is defined as:

$$r_i = \underset{e \in \mathcal{S} \setminus \mathcal{R}_{i-1}}{\operatorname{argmax}} \frac{a_{ee}}{1 + \sum_{j=1}^{i-1} a_{er_j}}, \quad (4.6)$$

where $\mathcal{R}_{i-1} = \{r_1, \dots, r_{i-1}\}$ is the set of the elements selected for previous indexes. Equation 4.6 can be understood as follows: select the next candidate from $\mathcal{S}$ that has high reciprocal affinity value (the numerator) and low affinity with the candidates already selected (the denominator).

The main objective of Equation 4.6 is to complete the guiding hypothesis, by favoring elements with high reciprocal affinity, but, at the same time, penalizing them for being similar to the ones already selected.

We present in the next subsections two approaches that use the selected representative nodes in order to generate the embeddings for elements of $\mathcal{C}$. An interesting point to note about the generated embeddings is that the value of each dimension expresses a degree of similarity with one or more representative nodes. This takes this method apart from other popular methods with respect to the interpretability of the generated embeddings: there are clear semantics associated with each dimension and such information might be useful for understanding results of tasks applied on the new representations. For example, in a classification task, specific labels might manifest strong relationship with specific dimensions of the embeddings.

4.3.3 Single Representative Node Embedding (RaDE)

Once the representative nodes have been chosen, the embedding can be generated for any desired data element. The embedding is computed based on the conjecture that nodes similar

to each other are also similar to the same set of representative nodes. Formally, the embedded vector $\mathbf{v}_e$ for an element $e \in C$ is defined as follows:

$$\mathbf{v}_e = [a_{er_1}, \dots, a_{er_d}], \quad (4.7)$$

where a_{er_i} denotes the affinity between the element e and the representative node r_i .

4.3.4 Multi-Representative Node Embedding (RaDE+)

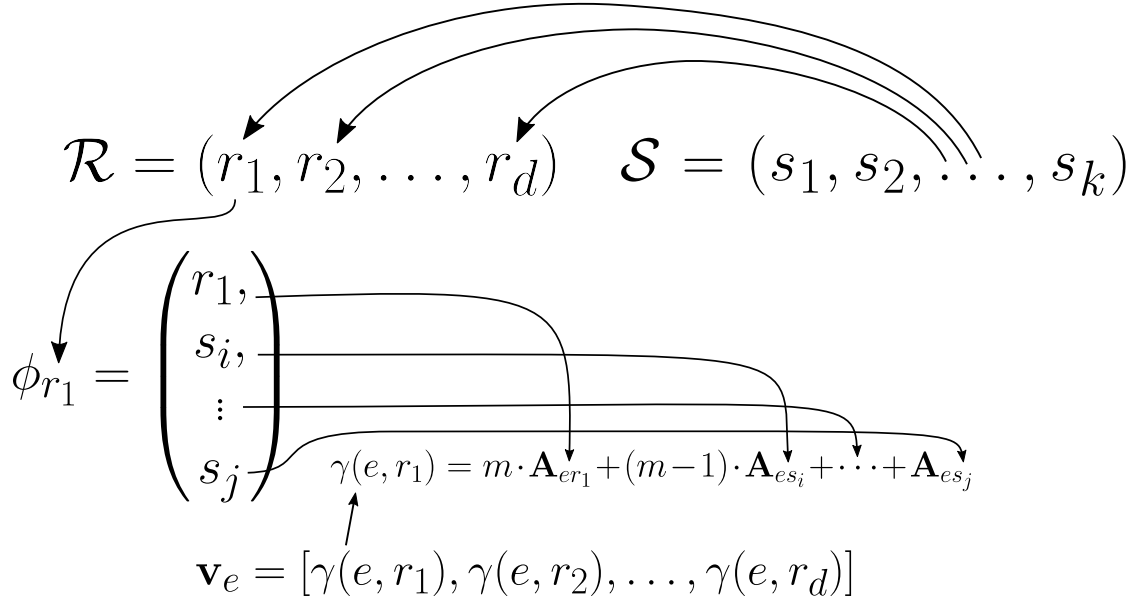
The selected high-representative nodes provide a general characterization of similarity structure of the dataset. Each element represent a set of similar objects, often portrayed by a class in typical retrieval and machine learning problems. However, creating a robust representation for a whole class is a challenging task, mainly due to the high diversity inter-class prevalent in many scenarios.

In order to address the inter-class diversity question and provide a more robust representation, a Multi-Representative Node Embedding is proposed. The central idea consists in expanding each selected high-effective representative node to a set of representatives, we call them *expansion sets*. In this way, a more robust and diversified representation can be constructed for each class.²

The procedure for the proposed approach is as follows. For each selected high-effective representative node r , we construct the expansion set ϕ_r , which contains r itself and other similar representatives among the candidates defined by $\mathcal{S}$. Once all expansion sets are generated, one for each dimension of the future vector space, the embedding for each node is generated by assigning to each dimension a value that aggregates the similarities between the node and each element of the respective expansion set. Figure 12 illustrates the proposed approach, which is formally defined in the following.

For each high-effective representative node $r \in \mathcal{R}_d$, let ϕ_r denote the expansion set of r . Each set ϕ_r is initialized to contain r and is expanded to contain elements from $\mathcal{S}$ that are similar to r . The expansion process is executed iteratively until all expansion sets reach a size defined by a parameter m . At each iteration, an expansion set ϕ_r is selected and an element s is removed

² Note that the single representative approach can be viewed as a specialization of the multi-representative case where the sizes of the expansion sets are always one.

Figure 12 – Multi-Representative Node Embedding from expansion sets ϕ_{ri}

from $\mathcal{S}$ and included into ϕ_r . The selection of s is based on the affinity matrix $\mathbf{A}$: s is the element from $\mathcal{S}$ with the highest affinity value for r . This expansion procedure is depicted in Algorithm 1.

Algoritmo 1 Multi-Representative Nodes Set Expansion.

Require: Set of high-effective representatives $\mathcal{R}_d$, Set of candidates $\mathcal{S}$, Expansion parameter m such that $m \cdot d \leq L$

Ensure: Expanded sets of representatives $\{\phi_{r_1}, \phi_{r_2}, \dots, \phi_{r_d}\}$

```

1: for all  $r \in \mathcal{R}_d$  do
2:    $\phi_r = \{r\}$ 
3:    $\mathcal{S} = \mathcal{S} \setminus \{r\}$ 
4: end for
5:  $j = 1$ 
6: while  $j \leq m$  do
7:   for all  $r \in \mathcal{R}_d$  do
8:      $s =_{x \in \mathcal{S}} a_{rx}$ 
9:      $\phi_r = \phi_r \cup \{s\}$ 
10:     $\mathcal{S} = \mathcal{S} \setminus \{s\}$ 
11:   end for
12:    $j = j + 1$ 
13: end while

```

The expansion sets are then used to compute the embeddings. For each element $e \in C$, a vector $\mathbf{v}_e$ of dimension d is generated, with each component assigned a value given by a function γ . The vector $\mathbf{v}_e$ and the function γ are defined as follows:

$$\mathbf{v}_e = [\gamma(e, r_1), \gamma(e, r_2), \dots, \gamma(e, r_d)], \quad (4.8)$$

$$\gamma(e, r) = \sum_{s \in \phi_r} (m - (\tau_{\phi_r}(s) - 1)) \cdot a_{es}, \quad (4.9)$$

where τ_{ϕ_r} represents the ranked list of ϕ_r with respect to affinities to r , that is, $\tau_{\phi_r}(s)$ gives the position of s in the ranked list. In practice, that position reflects the order in which the elements of ϕ_r were selected in Algorithm 1.

The function γ aggregates affinity values between e and elements of r 's expansion set in such a way that representatives closer to r are given higher weights than those that are further.

4.4 Efficiency, Scalability and Implementaion Aspects

For a large scale scenario, it would be very expensive or even unfeasible to compute the entire ranked lists for each element e_i in C . The hypothesis is that the relevant similarity information of an element $e_i \in C$ is encoded on the top- L positions and because of that, it would not be necessary to calculate those ranked lists entirely. The computation of all top- L ranked lists can be efficiently done in $O(n(\log L))$ using Ball-trees [54]. These computations can also be easily parallelized since they are independent of each other.

In our approach, a rank-based matrix $\mathbf{W}$ is calculated from the ranked lists $\tau_q, \forall q \in C$, such that for each row in $\mathbf{W}$, only L columns are used. Considering that often $L \ll n$, $\mathbf{W}$ is generally a very sparse matrix, which can be efficiently represented using formats such as *CSR* (*Compressed Sparse Rows*) or *CSC* (*Compressed Sparse Columns*) [55], saving a lot of memory space.

An expensive step for most of the approaches is calculating $\mathbf{A}$ from $\mathbf{W}^t$, especially when the $\mathbf{W}$ is a full $n \times n$ matrix. However, when $\mathbf{W}$ is represented in a *CSR* or *CSC* format, it allows fast matrix-vector multiplications and also drastically reduces the time to compute $\mathbf{A}$, which is the most expensive and time consuming step of RaDE.

These aspects provide an efficient way to store and process large scale datasets, making RaDE and RaDE+ scalable and more efficient in terms of time and memory usage in comparison to the evaluated baselines.

5 Experimental Evaluation

A broad experimental analysis was conducted to evaluate the proposed method. An overview about experiments is discussed in Section 5.1. Section 5.2 presents the datasets, while Section 5.3 describes the baselines. Section 5.4 and 5.5 discusses the evaluation measures and parameters settings, respectively. The results on diverse information retrieval tasks are discussed in Section 5.6, a visual analysis is presented in Section 5.7 and an efficiency evaluation is presented in Section 5.8.

5.1 Overview of experiments

Vector representations generated by RaDE/RaDE+ were evaluated on information retrieval and visualization tasks on 8 datasets of multiple domains (e.g. images, texts and social networks) and each of these datasets have different characteristics (e.g. dense or sparse graphs, weighted or not weighted graphs, large or low scale graphs). The efficiency of the proposed approaches was also evaluated and compared to the baselines.

The datasets of images and texts are not networks by itself. In order to be able to apply Network Representation Learning methods on these datasets, it is necessary to generate a graph from the original dataset. For this task, two main steps are necessary: *i)* extracting the feature vectors of samples in dataset; *ii)* calculating distances among the extracted feature vectors in order to generate a graph weighted by these distances. In this work the graphs generated for this kind of dataset were complete¹ and the Euclidean distance was used to calculate the weights.

Other category of datasets used to evaluate the proposed approach was datasets that are networks by itself. The selected network datasets are not originally weighted and for being able to apply RaDE and RaDE+, the graphs must be weighted. In order to assign weights, it was used an approach based on shared neighborhood between nodes.

The vector representations generated by RaDE were compared with the vector representations generated by 4 other NRL methods, which the implementations were provided by OpenNE library ². The experiments were executed on a machine with a Intel Xeon E5-2660 @ 2.0Ghz

¹ Note that the proposed method does not require the graph to be complete.

² <https://github.com/thunlp/OpenNE>

processor, 128GB of RAM and Arch Linux x86_64, kernel version 5.0.7 OS.

5.2 Datasets

We evaluated the effectiveness of RaDE on 8 datasets of multiple domains. MPEG-7 [56], Oxford17Flowers [19], Corel5k [57], COVIDx ³, and ALOI [58] are image datasets where each sample is described in function of its extracted features. The features for Oxford17Flowers [19], Corel5k [57] and MPEG-7 were extracted using the descriptors that presented the highest MAP according to experimental results presented on [59] and the features of the ALOI dataset were extracted considering the MAP results presented on [60]. For MPEG-7 [56], the features were extracted using a contour descriptor [61]. For both COVIDx and ALOI [58], CNNs were employed for feature extraction [62, 63].

The weighted graphs, for MPEG-7 [56], Oxford17Flowers [19], Corel5k [57], COVIDx, ALOI [58], 20NewsGroup [64], and Iris [65], were generated using the Euclidean distances between feature vectors of each sample. For datasets that originally are networks, such as Wiki ⁴, it was necessary a strategy for assigning the weights, since this dataset is not originally weighted. It was assumed that the more neighbors are shared between a pair of nodes, the more similar they are to each other. The approach used for assigning the weights is described in Equation 5.1,

$$w_{i,j} = \frac{1}{1 + |\mathcal{N}(i, m) \cap \mathcal{N}(j, m)|}, \quad (5.1)$$

where $\mathcal{N}(i, m)$ is the m nearest neighbors of a node N_i .

The details about each dataset evaluated on this work are exposed bellow:

- MPEG-7 [56]: 1,400 images, divided into 70 balanced classes, each one containing 20 samples. The features of images were extracted by CFD [61] which is a contour based descriptor.
- Oxford17Flowers [19]: 1,360 images of 17 different species of flowers, each one containing 80 different images. Each image is described in function of 2,048 features, which were

³ <https://github.com/lindawangg/COVID-Net>

⁴ <https://github.com/thunlp/OpenNE>

extracted using ResNet152 ⁵ [62] which is a residual neural network trained on ImageNet dataset [66]. This descriptor was chosen based on results present by [67].

- Corel5k [57]: 5,000 miscellaneous images (e.g. fireworks, trees, boats, tiles, etc). This dataset is divided into 50 categories, with 100 images each. Each image is described in function of 1,000 features, which were extracted using a DualPathNetwork92 ⁶. The chosen of the DPN92 is because it presents the best MAP result, in relation to other descriptor as shown in [67].
- ALOI [58]: 72,000 images of 1,000 varied objects captured in distinct viewpoint, occlusion, and illumination settings. The ResNet152 ⁷ [62] trained on ImageNet dataset [66] was used to extract 2,048 features for each image.
- COVIDx ⁸: 15,471 chest x-ray images divided into three classes (healthy patients, patients with pneumonia, and patients with COVID-19). The dataset is composed of different public sources. For feature extraction, it was used a InceptionResNetV2 [63] network pretrained on ImageNet [66] and fine-tuned on 5,216 chest x-ray images ⁹ of a 2008 Kaggle competition, which presents healthy and pneumonia patients. Each image is described by 1,024 features, extracted from the last fully convolutional layer.
- Iris [65]: A dataset widely used in pattern recognition task. It contains 150 samples of flowers, divided into 3 balanced classes. Each sample is described in function of the petal and sepal width and petal and sepal length.
- Wiki ¹⁰: A reference network between Wikipedia documents. It contains 2,405 documents divided into 19 classes. This network is composed by 17,981 directed and not weighted edges.
- 20NewsGroup [64]: Originally, it contains 18,846 news texts, sorted by date and separated into 20 categories. In this work, it was selected 3 categories (*comp.graphics*, *rec.sport.baseball* and *talk.politics.guns*) that compose a subset with 1,729 texts. It was done with the intention of reproduce the subset presented in [9]. The features of each

⁵ <https://github.com/Cadene/pretrained-models.pytorch>

⁶ <https://github.com/Cadene/pretrained-models.pytorch>

⁷ <https://github.com/Cadene/pretrained-models.pytorch>

⁸ <https://github.com/lindawangg/COVID-Net>

⁹ <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>

¹⁰ <https://github.com/thunlp/OpenNE>

document were extracted based on its respective TF-IDF [68] vector, which originally have 30.664 dimensions.

5.3 Baseline Algorithms

The embeddings generated by the proposed methods were compared with the following algorithms: Node2Vec (see Section 3.4.2); HOPE (see Section 3.4.5); LINE (see Section 3.4.3) and SDNE (see Section 3.4.4).

5.4 Evaluation Measures

The results were reported considering two different effectiveness measures commonly used for information retrieval tasks: Precision and Mean Average Precision (MAP). Given a ranked list τ_q as input, the measures report a score in the interval $[0, 1]$, where higher values refer to better results. The results correspond to the mean of these measures computed for each of the ranked lists in the dataset.

5.4.1 Precision

The Precision measure corresponds to the number of retrieved samples that belong to the class of the query element in the top- k positions. This is formally defined by Equation 5.2.

$$P(q, k) = \frac{1}{k} \times \sum_{i=1}^k f_c(\tau_q^{-1}(i), q), \quad (5.2)$$

where q is the index of the query element, k is the size of the ranked list, $\tau_q^{-1}(i)$ is the i -th element in the ranked list and f_c is a function that returns 1 if two elements belong to the same class and 0 otherwise. In this case, it is equivalent to the number of true positives against the sum of true positives and false positives. For readability purposes, to report the mean of the Precision of all elements, we use $P@k$.

5.4.2 Mean Average Precision (MAP)

The Average Precision (A_p) computes the sum of the Precisions for different depths of a ranked list, which is formally defined in Equation 5.3.

$$A_p(q, k) = \frac{1}{f_s(q)} \times \sum_{i=1}^k P(q, i) \times f_c(\tau_q^{-1}(i), q), \quad (5.3)$$

where q is the index of the query element, k is the depth of the ranked list, f_s is a function that returns the class size of an element, $\tau_q^{-1}(i)$ is the i -th element in the ranked list and f_c is a function that returns 1 if two elements belong to the same class and 0 otherwise.

The MAP (Mean Average Precision) is defined as the mean of A_p for all the Q queries. The formulation is given by Equation 5.4.

$$MAP = \frac{\sum_{q=1}^Q A_p(q)}{Q} \quad (5.4)$$

5.5 Parameter Settings

For information retrieval tasks, the generated vector representations were composed by 128 dimensions. This value have been chosen in order to match the results presented on [4]. For visualization tasks, vector representations were generated with 50 dimensions, in order to suppress noise and speed up the computation of pairwise distances between samples, as recommended in the t-SNE documentation provided by scikit-learn library ¹¹. Except for the Iris dataset, where 4 dimensional embeddings were used for retrieval and 20 dimensional embeddings for visualization. Other exception of the dimensionality was for the ALOI dataset. The embeddings were generated with 2048 dimensions, due to the bigger size of the input graph and number of classes in relation to other datasets evaluated. It is important to notice that the dimensionality of the embeddings generated with RaDE/RaDE+ must be at least the number of classes in the dataset. Small dimensionality values tend to degenerate the results.

Table 2 reports the parameters considered for the executions of the proposed approach on each dataset. Note that both RaDE and RaDE+ has the same set of parameters, except m , which is specific to RaDE+. The only exception for m on RaDE+ was on Iris dataset, since it has a smaller number of elements in comparison to other datasets. This value was found empirically. The

¹¹ <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html>

parameter L , which is the size of ranked lists is associated with the number of elements per class of the dataset. So, L started from the number of elements per class and it was empirically varied.

	Dataset							
	MPEG-7	Flowers	20NewsGroup	Corel5k	COVIDx	Wiki	ALOI	Iris
L	25	70	900	100	800	76	90	80
d	128						2048	4
k	1000						10000	150
m	3						30	

Table 2 – Parameter settings used on RADE and RADE+.

Table 3 shows the parameter settings for each of the baseline method. Most of them were executed with the default parameter configuration, provided by the OpenNE library, such as performed in [4].

Method	Parameters
Node2vec	Number of paths: 10
	Path length: 80
	Window size: 10
	p: 0,25 q: 0,25
LINE	Negative ratio: 5
	First and second order
SDNE	Autoencoder List: [1000, 128]
	Learning rate: 0,01
	First order loss: 10^{-6}
	l1 loss : 10^{-5}
	l2 loss: 10^{-6}
HOPE	Batch size: 200
	OpenNE defaults

Table 3 – Specific parameter settings of each NRL baseline method.

5.6 Retrieval Results

In this section it is presented the results for RaDE and RaDE+ on information retrieval tasks, considering various domains (e.g. images, texts and social networks). Since 20NewsGroup is a textual dataset, it was used TF-IDF [68] algorithm to extract its features in relation to the terms of a vocabulary.

Table 4 – 20NewsGroup dataset evaluation.

	Original	RaDE+	RaDE	HOPE	LINE	SDNE	Node2vec
P@2	0.9690	0.9806	0.9786	0.9253	0.6621	0.7292	0.6693
P@4	0.9337	0.9645	0.9608	0.8824	0.4994	0.5791	0.4998
P@8	0.8841	0.9489	0.9451	0.8523	0.4108	0.4932	0.4162
P@16	0.8318	0.9380	0.9333	0.8256	0.3691	0.4551	0.3753
P@32	0.7713	0.9284	0.9223	0.7937	0.3488	0.4352	0.3543
P@64	0.6965	0.9170	0.9066	0.7479	0.3418	0.4242	0.3445
P@128	0.6076	0.8977	0.8828	0.6765	0.3381	0.4166	0.3399
MAP	0.4924	0.7856	0.7538	0.5132	0.3396	0.4051	0.3404

Table 4 presents the experimental results on 20NewsGroup dataset. RaDE and RaDE+ presented the best results compared to the baselines and even compressing the original representations, from 30.664 dimensions to 128, RaDE and RaDE+ presented gains in Precision and MAP.

Table 5 – Wiki dataset evaluation.

	RaDE+	RaDE	HOPE	LINE	SDNE	Node2vec
P@2	0.7559	0.7601	0.7848	0.8299	0.8008	0.8168
P@4	0.6244	0.6307	0.6511	0.7059	0.6740	0.6936
P@8	0.5359	0.5416	0.5634	0.6061	0.5696	0.6001
P@16	0.4746	0.4774	0.4836	0.5108	0.4666	0.5175
P@32	0.4221	0.4242	0.4027	0.4198	0.3649	0.4484
P@64	0.3669	0.3707	0.3103	0.3317	0.2720	0.3756
P@128	0.2864	0.2886	0.2233	0.2489	0.2021	0.2933
MAP	0.2558	0.2593	0.1971	0.2317	0.1833	0.2554

Table 5 present results on Wiki dataset. The best precision results were presented by LINE and Node2vec. Although, the results presented by RaDE and RaDE+, specially for higher precisions, were close to the presented by Node2vec and besides that, the best MAP result was presented by RaDE.

RaDE+ also presented the best Precision and MAP scores on Iris, as shwon on Table 6. Since Iris have only 50 elements per class, P@64 and P@128 were not evaluated.

Table 6 – Iris dataset evaluation.

	Original	RaDE+	RaDE	HOPE	LINE	SDNE	Node2vec
P@2	0.9800	0.9700	0.9533	0.9833	0.9366	0.99	0.9233
P@4	0.9633	0.9450	0.9233	0.9650	0.8550	0.9616	0.8850
P@8	0.9500	0.9242	0.9108	0.9533	0.8008	0.9233	0.8616
P@16	0.9337	0.9171	0.8983	0.9362	0.7591	0.8791	0.8370
P@32	0.8947	0.9067	0.8854	0.8937	0.7068	0.8410	0.8029
MAP	0.8858	0.9141	0.8969	0.8905	0.6988	0.8400	0.8082

Table 7 – MPEG-7 dataset evaluation.

	Original	RaDE+	RaDE	HOPE	LINE	SDNE
P@2	0.9921	0.9850	0.9857	0.9871	0.8892	0.6146
P@4	0.9786	0.9661	0.9638	0.9560	0.7889	0.4019
P@8	0.9320	0.9391	0.9348	0.9023	0.6800	0.2716
P@16	0.8212	0.8939	0.8893	0.8143	0.5735	0.1894
P@32	0.5125	0.5593	0.5571	0.5144	0.3810	0.1369
P@64	0.2760	0.2884	0.2883	0.2751	0.2226	0.0990
P@128	0.1448	0.1482	0.1475	0.1442	0.1266	0.0720
MAP	0.8071	0.8842	0.8787	0.8008	0.5565	0.1860

Table 8 – Flowers dataset evaluation.

	Original	RaDE+	RaDE	HOPE	LINE	SDNE
P@2	0.9401	0.9324	0.9335	0.9047	0.7772	0.6117
P@4	0.8906	0.8871	0.8831	0.8292	0.6250	0.3641
P@8	0.8349	0.8529	0.8487	0.7638	0.5245	0.2382
P@16	0.7707	0.8219	0.8143	0.6926	0.4596	0.1652
P@32	0.6800	0.7793	0.7728	0.6072	0.3976	0.1280
P@64	0.5488	0.6992	0.6913	0.4856	0.3314	0.1060
P@128	0.3713	0.4712	0.4672	0.3373	0.2555	0.0963
MAP	0.5183	0.6885	0.6789	0.4466	0.2930	0.1115

MPEG-7, according to Table 7 and Flowers, as shown on Table 8. On Corel5k dataset, RaDE presented better results than RaDE+ some Precision and MAP measures. In all these

Table 9 – Corel5k dataset evaluation.

	Original	RaDE+	RaDE	HOPE	LINE	SDNE
P@2	0.9621	0.9494	0.9505	0.7867	0.6144	0.5160
P@4	0.9290	0.9161	0.9155	0.6622	0.4140	0.2742
P@8	0.8988	0.8918	0.8925	0.5894	0.3026	0.1518
P@16	0.8595	0.8706	0.8721	0.5377	0.2358	0.0894
P@32	0.8070	0.8505	0.8527	0.4931	0.1930	0.0581
P@64	0.7206	0.8182	0.8191	0.4372	0.1584	0.0422
P@128	0.5232	0.6300	0.6300	0.3211	0.1253	0.0339
MAP	0.6517	0.7823	0.7826	0.3100	0.1043	0.0375

datasets, the proposed approach was able to improve the effectiveness while compressing the the original vector representations.

Table 10 – COVIDx dataset evaluation.

	Original	RaDE+	RaDE	HOPE	LINE	SDNE
P@2	0.8962	0.8811	0.8823	0.8849	0.8905	0.8637
P@4	0.8381	0.8179	0.8180	0.8225	0.8312	0.7873
P@8	0.8040	0.7816	0.7819	0.7880	0.7968	0.7465
P@16	0.7827	0.7613	0.7620	0.7708	0.7780	0.7235
P@32	0.7688	0.7477	0.7488	0.7586	0.7657	0.7069
P@64	0.7572	0.7367	0.7374	0.7487	0.7563	0.6945
P@128	0.7467	0.7264	0.7267	0.7397	0.7477	0.6801
MAP	0.5394	0.5470	0.5455	0.4983	0.5221	0.3015

Results on COVIDx dataset are presented in Table 10. The best MAP score was achieved by RaDE+ and RADE. Both of the proposed methods were able to improve the effectiveness of the original representations even using less data for representing them. Although LINE presented the best precision results on COVIDx, the proposed approaches also performed well and its results were close to the LINE ones. RaDE and RaDE+ require a lot less memory and time to run in comparison to the baselines. It was not possible to generate the embeddings with Node2Vec on COVIDx dataset, as shown in Table 10, since this method requires a lot of memory usage, specially for large datasets and computational resources available for executing the experiments

could not handle the execution of Node2Vec on this dataset.

Table 11 – ALOI dataset evaluation.

	Original	RaDE+	RaDE	SDNE
P@2	0.9984	0.9923	0.9923	0.9906
P@4	0.9968	0.9824	0.9831	0.9776
P@8	0.9925	0.9651	0.9676	0.9553
P@16	0.9777	0.9367	0.9411	0.9185
P@32	0.9265	0.8889	0.8953	0.8580
P@64	0.7940	0.8141	0.8209	0.7487
P@128	0.4630	0.4690	0.4712	0.4466
MAP	0.7961	0.8014	0.8063	0.7356

Table 11 shows results on ALOI dataset. Both RaDE and RaDE+ were able to be applied in a large-scale scenario besides improving the original MAP result and outperforming SDNE. Since ALOI is a very large dataset, the only baseline that was able to be executed on the available machine to run the experiments was SDNE.

5.7 Visualization Results

Vector representations generated with RaDE were also evaluated on visualization tasks. On this experiment, it was evaluated the effectiveness of vector representations generated by the baselines presented on Subsection 5.3 on three datasets: 20NewsGroup, COVIDx, and Iris. Those datasets have been chosen for this task due to the fact that they are composed by only 3 classes and this facilitates a visual analysis.

For each of the selected datasets, it was generated embeddings with RaDE, RaDE+ and baseline algorithms described in Section 5.3. The embeddings were then projected to the bi-dimensional plane via UMAP [26], a fairly recent non-linear dimensionality reduction technique, and plotted using Matplotlib¹². It was also plotted the bi-dimensional visualization of the original features for comparison.

A 50-dimensional embedding representation was generated for 20NewsGroup and COVIDx. And it was generated a 20-dimensional embedding representation for Iris. The

visualizations for each of those datasets are presented in Figures 13, 14 and 15, respectively. Each class in the visualizations is represented with a different color.

For each dataset, the following observations are made:

- *20NewsGroup*: the visualizations for both RaDE and RaDE+ allow the viewer to notice well defined clusters, one for each class. Although most of the other methods also show the tendency of keeping elements of the same class close to each other, the viewer can see that the classes are not as well separated as done for RaDE and RaDE+.
- *COVIDx*: For visualization purposes, only a random subset of 1500 elements from the full COVIDx dataset were used for the projection to the bi-dimensional space¹³. In the plots healthy patients are red, patients with pneumonia are green and patients with COVID-19 are blue. Although neither RaDE or the baseline methods were able to separate COVID-19 diagnoses, we can see that projections for RaDE, RaDE+ and SDNE show separations of two clusters, one with a majority of healthy patients and another with a majority of pneumonia patients.
- *Iris*: For this dataset, all methods but RaDE and RaDE+ scattered the points making the visualizations worse than the plot for the original features. Further investigation is required to understand what is causing that, but it is suspect that the phenomena might be related to the size of the dataset, which is small (150). Nevertheless, we can see that RaDE and RaDE+ successfully improved separation between different classes and arguably detected further separation between elements of the same class.

¹² <https://matplotlib.org/>

¹³ Note, however, that the full dataset was used for the embedding tasks. Also, it is worth mentioning that the same random subset was used for the projections of each evaluated method.

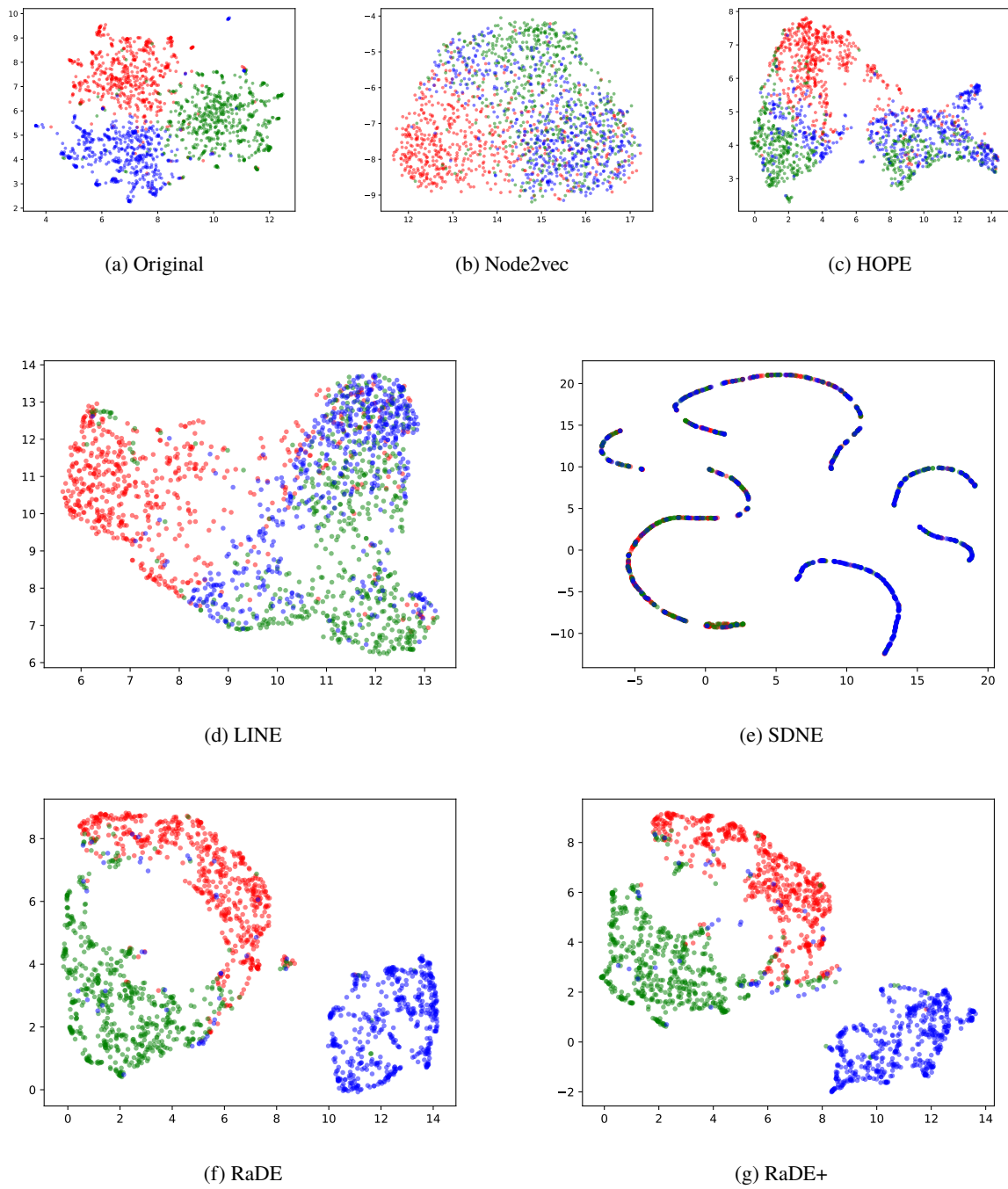


Figure 13 – UMAP visualization on 20NewsGroup dataset.

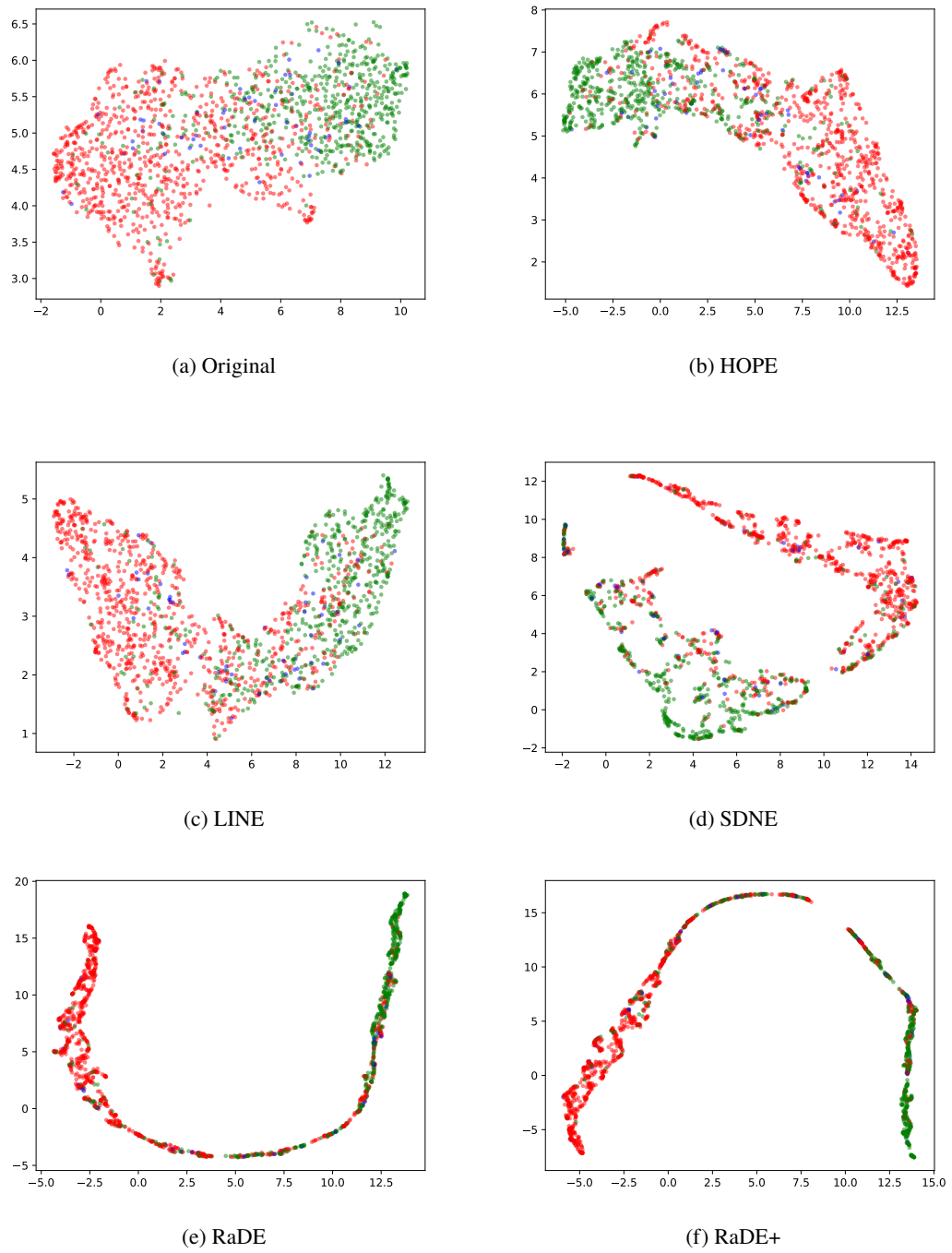


Figure 14 – UMAP visualization on COVIDx dataset.

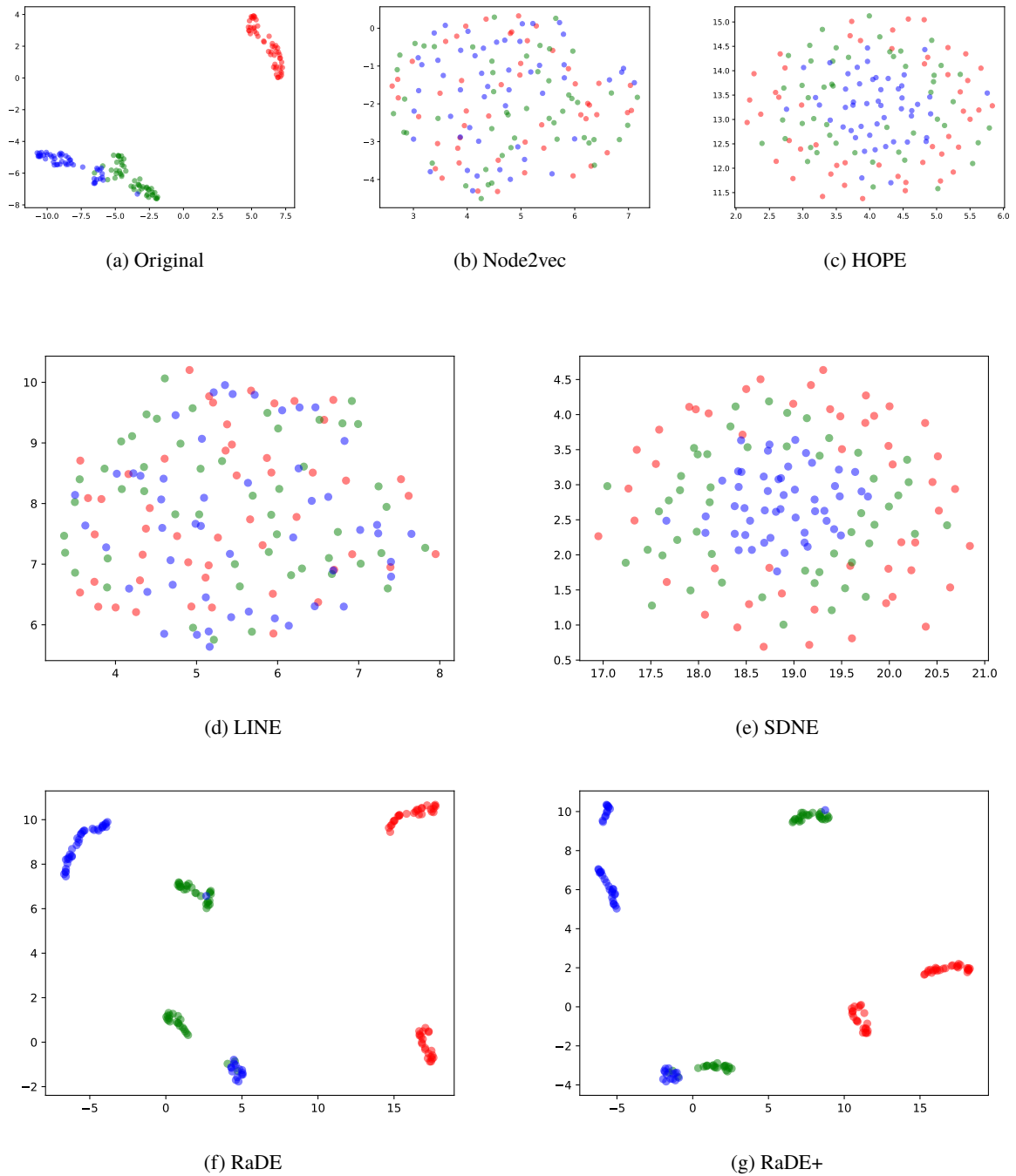


Figure 15 – UMAP visualization on Iris dataset.

5.8 Efficiency Evaluation

In order to evaluate the efficiency of RaDE and RaDE+, the execution time was measured and compared to the baselines. The method was evaluated on two scenarios: in a small and in a medium dataset. The execution time in large-scale datasets was not evaluated because most of the baselines struggled to scale, either because of time or memory consumption. It is important

to notice that all the methods were implemented in python and the same environment was used to run the experiments, in order to have a fair time comparison. As shown on Table 11, the only method besides RaDE/RaDE+ that was able to conclude the execution, with the available computational resources for running these experiments, was SDNE. Therefore, in order to have enough results for a comparison, large-scale datasets were not considered. Iris was used as the small dataset, while 20NewsGroup as the medium one.

For both Iris and 20NewsGroup, the time was measured in seconds. Each method was executed 5 times, then the arithmetic mean and standard deviation were calculated. Each execution is labeled as $t@i$ and its respective time taken is presented.

Table 12 presents the time evaluation on Iris dataset and Table 13 presents the evaluation on 20NewsGroup dataset. In both scenarios RaDE and RaDE+ outperformed all other methods by a significant margin. Table 6 shows that while RaDE takes about 1 second to generate the embeddings for the Iris dataset, Node2vec takes about 35 minutes and Table 13 shows that while RaDE takes about 15 seconds to generate the embeddings for 20NewsGroup, LINE takes about 32 minutes. Since Node2vec took much longer time to execute than the competitors on Iris, it was not evaluated on 20NewsGroup ¹⁴.

	RaDE	RaDE+	HOPE	SDNE	LINE	Node2vec
t@1	1.025	1.136	10.626	8.539	82.840	2121.374
t@2	1.308	1.048	11.345	8.984	88.096	2114.561
t@3	1.102	1.026	11.343	8.983	89.604	2132.115
t@4	1.465	1.074	9.818	8.574	82.881	2138.399
t@5	1.461	1.359	11.827	9.675	82.669	2143.682
Avg	1.272	1.128	10.991	8.950	85.218	2130.026
SD	0.181	0.120	0.701	0.409	3.004	10.723

Table 12 – Time evaluation (in seconds) on Iris dataset.

	RaDE	RaDE+	HOPE	SDNE	LINE
t@1	17.991	15.806	42.379	61.713	1944.996
t@2	17.807	16.248	31.397	58.437	1913.192
t@3	18.303	17.556	40.684	59.614	1945.078
t@4	15.477	14.488	40.560	60.587	1935.345
t@5	18.323	15.276	44.417	57.505	1910.417
Avg	17.580	15.874	39.887	59.571	1929.805
SD	1.069	1.024	4.469	1.495	15.143

Table 13 – Time evaluation (in seconds) on 20NewsGroup dataset.

¹⁴ The results of Node2vec presented on Table 4 took about some days to be generated and the execution time was not measured.

6 Conclusions and Future Works

Considering the importance of graphs in many scenarios and the existing issues of processing and representing these structures, specially in large-scale scenarios, this work proposed RaDE [17] (Rank Diffusion Embedding), an unsupervised method for generating low-dimensional vector representations based on similarity between common nodes and high-effective representative nodes in a network. With the goal of improving the quality of the embeddings generated by RaDE, this work also presented RaDE+ [18], which is a variant of RaDE that considers multiple representative nodes for each dimension of the embeddings. RaDE was published on VISAPP 2020 [17] and its variant, RaDE+ was recently submitted to *Information Processing & Management* journal [18].

A wide and comprehensive experimental evaluation was conducted to assess the effectiveness of generated embedding in retrieval tasks. The evaluation was conducted on 8 diverse datasets considering different data modalities: images, text, social networks, and a traditional pattern recognition dataset. The experimental results were compared with 4 recently proposed node embedding approaches. In various scenarios, the proposed methods achieved the best results in most of datasets, demonstrating the ability in generating high-effective representations. The visualization results show the capacity of the method of effectively separating the classes in clusters, always better than the original features for the evaluated datasets.

RaDE and RaDE+ presented best results on originally wheighted datasets and this can be seen as a limitation of the methods: in order to present a good effectiveness, elements of the input graph must have good quality ranked lists, that is, there must be a good metric of similarity-dissimilarity between the elements.

In general, RaDE and RaDE+ achieved remarkable results for both tasks: visualization and information retrieval in most of the evaluated datasets. The proposed approaches were capable of creating high-effective low-dimensional vector representations that can be useful in a wide variety of applications, not only for the ones evaluated (i.e information retrieval and visualization) but also for others such as node classification and link-prediction. In most cases, the proposed methods was not only capable of providing very dense and smaller representations in comparison to the original ones, but has also improved the general effectiveness by a significant margin. For

the ALOI dataset, for example, the provided output was about 35 times smaller than the original and even so, RaDE presented an absolute gain of more than 1% of MAP. Besides the promising results in terms of effectiveness, the proposed methods also demonstrated to be very efficient and present best execution times in relation to all the baselines.

Another interesting advantage that RaDE/RaDE+ offers in comparison to other graph embedding methods is the fact that each dimension of the vector representations have interpretability. Since most of the existing NRL methods relies on some optimization strategy, the resulting embeddings are created with the only goal of satisfying an objective function, that is, most NRL methods are not necessarily concerned about the interpretability and the semantic meaning of the dimensions of the embedded representation, which is a downside because it could have useful information for analysis being left aside. In contrast to that, RaDE/RaDE+ provides semantically enriched embeddings due to the nature of the rank-based model used to select high-effective representatives.

For future work, it is intend to perform a strict time and complexity analysis of the proposed methods. Besides that, it is also intend to evaluate RaDE/RaDE+ for other machine learning tasks such as classification, especially in unsupervised and weakly supervised scenarios. One possible idea is to evaluate RaDE and RaDE+ both as a pre-processing for different classifiers.

Bibliography

- 1 DAVID, E.; JON, K. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. USA: Cambridge University Press, 2010. ISBN 0521195330.
- 2 HUANG, X.; CUI, P.; DONG, Y.; LI, J.; LIU, H.; PEI, J.; SONG, L.; TANG, J.; WANG, F.; YANG, H.; ZHU, W. Learning from networks: Algorithms, theory, and applications. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2019. (KDD '19), p. 3221–3222. ISBN 978-1-4503-6201-6.
- 3 CAI, H.; ZHENG, V. W.; CHANG, K. C. A comprehensive survey of graph embedding: Problems, techniques and applications. *CoRR*, abs/1709.07604, 2017. Disponível em: <<http://arxiv.org/abs/1709.07604>>.
- 4 GOYAL, P.; FERRARA, E. Graph embedding techniques, applications, and performance: A survey. *CoRR*, abs/1705.02801, 2017. Disponível em: <<http://arxiv.org/abs/1705.02801>>.
- 5 HAMILTON, W. L.; YING, R.; LESKOVEC, J. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, v. 40, n. 3, p. 52–74, 2017.
- 6 CUI, P.; WANG, X.; PEI, J.; ZHU, W. A survey on network embedding. *CoRR*, abs/1711.08752, 2017. Disponível em: <<http://arxiv.org/abs/1711.08752>>.
- 7 TANG, J.; QU, M.; WANG, M.; ZHANG, M.; YAN, J.; MEI, Q. Line: Large-scale information network embedding. In: *Proceedings of the 24th International Conference on World Wide Web*. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015. (WWW '15), p. 1067–1077. ISBN 978-1-4503-3469-3. Disponível em: <<https://doi.org/10.1145/2736277.2741093>>.
- 8 GROVER, A.; LESKOVEC, J. Node2vec: Scalable feature learning for networks. In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2016. (KDD '16), p. 855–864. ISBN 978-1-4503-4232-2. Disponível em: <<http://doi.acm.org/10.1145/2939672.2939754>>.
- 9 WANG, D.; CUI, P.; ZHU, W. Structural deep network embedding. In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2016. (KDD '16), p. 1225–1234. ISBN 978-1-4503-4232-2. Disponível em: <<http://doi.acm.org/10.1145/2939672.2939753>>.
- 10 OU, M.; CUI, P.; PEI, J.; ZHANG, Z.; ZHU, W. Asymmetric transitivity preserving graph embedding. In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2016. (KDD '16), p. 1105–1114. ISBN 978-1-4503-4232-2. Disponível em: <<http://doi.acm.org/10.1145/2939672.2939751>>.
- 11 DAOKUN, Z.; YIN, J.; ZHU, X.; ZHANG, C. Network representation learning: A survey. *IEEE Transactions on Big Data*, PP, 12 2017.
- 12 BELKIN, M.; NIYOGI, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. Cambridge, MA, USA: MIT Press, 2001. (NIPS'01), p. 585–591.

- 13 ROWEIS, S. T.; SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, American Association for the Advancement of Science, v. 290, n. 5500, p. 2323–2326, 2000. ISSN 0036-8075. Disponível em: <<https://science.sciencemag.org/content/290/5500/2323>>.
- 14 PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: Online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2014. (KDD '14), p. 701–710. ISBN 978-1-4503-2956-9. Disponível em: <<http://doi.acm.org/10.1145/2623330.2623732>>.
- 15 MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. Disponível em: <<http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781>>.
- 16 KIPF, T. N.; WELING, M. Semi-supervised classification with graph convolutional networks. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2017.
- 17 FERNANDO, F. A. de; PEDRONETTE, D. C. G.; SOUSA, G. J. de; VALEM, L. P.; GUILHERME, I. R. Rade: A rank-based graph embedding approach. In: *International Conference on Computer Vision Theory and Applications, VISAPP*. [S.l.: s.n.], 2020. p. 142–152.
- 18 FERNANDO, F. A. de; PEDRONETTE, D. C. G.; SOUSA, G. J. de; VALEM, L. P.; GUILHERME, I. R. Rade+: A semantic rank-based graph embedding algorithm. In: *Information Processing Management, IPM*. [S.l.: s.n.], 2020. Submitted.
- 19 NILSBACK, M.-E.; ZISSERMAN, A. A visual vocabulary for flower classification. In: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*. Washington, DC, USA: IEEE Computer Society, 2006. (CVPR '06), p. 1447–1454. ISBN 0-7695-2597-0. Disponível em: <<http://dx.doi.org/10.1109/CVPR.2006.42>>.
- 20 LESKOVEC, J.; MCAULEY, J. J. Learning to discover social circles in ego networks. In: PEREIRA, F.; BURGESS, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 539–547. Disponível em: <<http://papers.nips.cc/paper/4532-learning-to-discover-social-circles-in-ego-networks.pdf>>.
- 21 KUNEGIS, J. Konect: The koblenz network collection. In: *Proceedings of the 22Nd International Conference on World Wide Web*. New York, NY, USA: ACM, 2013. (WWW '13 Companion), p. 1343–1350. ISBN 978-1-4503-2038-2. Disponível em: <<http://doi.acm.org/10.1145/2487788.2488173>>.
- 22 WEST, D. B. et al. *Introduction to graph theory*. [S.l.]: Prentice hall Upper Saddle River, NJ, 1996. v. 2.
- 23 WILLIAMSON, E. *Lists, Decisions and Graphs*. S. Gill Williamson. Disponível em: <https://books.google.com.br/books?id=vaXv_yhefG8C>.
- 24 JOLLIFFE, I.; SPRINGER-VERLAG. *Principal Component Analysis*. Springer, 2002. (Springer Series in Statistics). ISBN 9780387954424. Disponível em: <https://books.google.com.br/books?id=_olByCrhjwIC>.
- 25 MAATEN, L. van der; HINTON, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, v. 9, p. 2579–2605, 2008. Disponível em: <<http://www.jmlr.org/papers/v9/vandermaaten08a.html>>.

- 26 MCINNES, L.; HEALY, J.; MELVILLE, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
- 27 HAMILTON, W. L.; YING, R.; LESKOVEC, J. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, v. 40, p. 52–74, 2017.
- 28 CHEN, H.; PEROZZI, B.; AL-RFOU, R.; SKIENA, S. A tutorial on network embeddings. In: . [S.l.: s.n.], 2018.
- 29 LIU, Z.; ZHENG, V. W.; ZHAO, Z.; ZHU, F.; CHANG, K. C.-C.; WU, M.; YING, J. Semantic proximity search on heterogeneous graph by proximity embedding. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2017. (AAAI'17), p. 154–160.
- 30 YANARDAG, P.; VISHWANATHAN, S. Deep graph kernels. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2015. (KDD '15), p. 1365–1374. ISBN 9781450336642. Disponível em: <<https://doi.org/10.1145/2783258.2783417>>.
- 31 BORDES, A.; CHOPRA, S.; WESTON, J. Question answering with subgraph embeddings. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014. p. 615–620. Disponível em: <<https://www.aclweb.org/anthology/D14-1067>>.
- 32 WANG, X.; CUI, P.; WANG, J.; PEI, J.; ZHU, W.; YANG, S. Community preserving network embedding. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2017. (AAAI'17), p. 203–209.
- 33 MOUSAVI, S. F.; SAFAYANI, M.; MIRZAEI, A.; BAHONAR, H. Hierarchical graph embedding in vector space by graph pyramid. *Pattern Recogn.*, Elsevier Science Inc., USA, v. 61, n. C, p. 245–254, jan. 2017. ISSN 0031-3203. Disponível em: <<https://doi.org/10.1016/j.patcog.2016.07.043>>.
- 34 LI, C.; MA, J.; GUO, X.; MEI, Q. Deepcas: An end-to-end predictor of information cascades. In: *Proceedings of the 26th International Conference on World Wide Web*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017. (WWW '17), p. 577–586. ISBN 9781450349130. Disponível em: <<https://doi.org/10.1145/3038912.3052643>>.
- 35 CAO, S.; LU, W.; XU, Q. Deep neural networks for learning graph representations. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2016. (AAAI'16), p. 1145–1152.
- 36 HENAFF, M.; BRUNA, J.; LECUN, Y. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015. Disponível em: <<http://arxiv.org/abs/1506.05163>>.
- 37 DUVENAUD, D. K.; MACLAURIN, D.; IPARRAGUIRRE, J.; BOMBARELL, R.; HIRZEL, T.; ASPURU-GUZI, A.; ADAMS, R. P. Convolutional networks on graphs for learning molecular fingerprints. In: CORTES, C.; LAWRENCE, N. D.; LEE, D. D.; SUGIYAMA, M.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems* 28. Curran Associates, Inc., 2015. p. 2224–2232. Disponível em: <<http://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints.pdf>>.

- 38 DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016. Disponível em: <<http://arxiv.org/abs/1606.09375>>.
- 39 BORDES, A.; USUNIER, N.; GARCIA-DURÁN, A.; WESTON, J.; YAKHNENKO, O. Translating embeddings for modeling multi-relational data. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Red Hook, NY, USA: Curran Associates Inc., 2013. (NIPS'13), p. 2787–2795.
- 40 LIN, Y.; LIU, Z.; SUN, M.; LIU, Y.; ZHU, X. Learning entity and relation embeddings for knowledge graph completion. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2015. (AAAI'15), p. 2181–2187. ISBN 0262511290.
- 41 JI, G.; HE, S.; XU, L.; LIU, K.; ZHAO, J. Knowledge graph embedding via dynamic mapping matrix. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, 2015. p. 687–696. Disponível em: <<https://www.aclweb.org/anthology/P15-1067>>.
- 42 NEWMAN, M. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, Taylor Francis, v. 46, n. 5, p. 323–351, 2005. Disponível em: <<https://doi.org/10.1080/00107510500052444>>.
- 43 BARABASI, A.-L.; BONABEAU, E. Scale-free networks. *Scientific American*, 06 2003.
- 44 ZIPF, G. K. Human behavior and the principle of least effort. v. 6, 01 1949.
- 45 LUO, D.; DING, C.; NIE, F.; HUANG, H. Cauchy graph embedding. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Madison, WI, USA: Omnipress, 2011. (ICML'11), p. 553–560. ISBN 9781450306195.
- 46 HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *science*, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006.
- 47 HARTIGAN, J. A.; WONG, M. A. Algorithm AS 136: A K-Means clustering algorithm. *Applied Statistics*, Blackwell Publishing for the Royal Statistical Society, v. 28, n. 1, p. 100–108, 1979. ISSN 00359254. Disponível em: <<http://dx.doi.org/10.2307/2346830>>.
- 48 ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996. (KDD'96), p. 226–231. Disponível em: <<http://dl.acm.org/citation.cfm?id=3001460.3001507>>.
- 49 ZHONG, Z.; ZHENG, L.; CAO, D.; LI, S. Re-ranking person re-identification with k-reciprocal encoding. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. [S.l.: s.n.], 2017. p. 3652–3661.
- 50 Pedronette, D. C. G.; Valem, L. P.; Almeida, J.; da S. Torres, R. Multimedia retrieval through unsupervised hypergraph-based manifold ranking. *IEEE Transactions on Image Processing*, v. 28, n. 12, p. 5824–5838, Dec 2019.

- 51 DONOSER, M.; BISCHOF, H. Diffusion processes for retrieval revisited. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. [S.l.: s.n.], 2013. p. 1320–1327.
- 52 BAI, S.; BAI, X.; TIAN, Q.; LATECKI, L. J. Regularized diffusion process on bidirectional context for object retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 41, n. 5, p. 1213–1226, 2019.
- 53 PEDRONETTE, D. C. G.; TORRES, R. da S. Unsupervised rank diffusion for content-based image retrieval. *Neurocomputing*, v. 260, p. 478 – 489, 2017. ISSN 0925-2312. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0925231217308330>>.
- 54 OMOHUNDRO, S. M. *Five Balltree Construction Algorithms*. [S.l.], 1989. Disponível em: <<ftp://ftp.icsi.berkeley.edu/pub/techreports/1989/tr-89-063.pdf>>.
- 55 BULUC, A.; FINEMAN, J. T.; FRIGO, M.; GILBERT, J. R.; LEISERSON, C. E. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In: *Proceedings of the Twenty-First Annual Symposium on Parallelism in Algorithms and Architectures*. New York, NY, USA: Association for Computing Machinery, 2009. (SPAA '09), p. 233–244. ISBN 9781605586069. Disponível em: <<https://doi.org/10.1145/1583991.1584053>>.
- 56 LATECKI, L. J.; LAKÄMPER, R.; ECKHARDT, U. Shape descriptors for non-rigid shapes with a single closed contour. In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2000. p. 424–429.
- 57 LIU, G.-H.; ZHANG, L.; HOU, Y.-K.; LI, Z.-Y.; YANG, J.-Y. Image retrieval based on multi-texton histogram. *Pattern Recogn.*, Elsevier Science Inc., New York, NY, USA, v. 43, n. 7, p. 2380–2389, jul. 2010. ISSN 0031-3203. Disponível em: <<http://dx.doi.org/10.1016/j.patcog.2010.02.012>>.
- 58 GEUSEBROEK, J.-M.; BURGHOUTS, G. J.; SMEULDERS, A. W. M. The amsterdam library of object images. *Int. J. Comput. Vision*, Kluwer Academic Publishers, USA, v. 61, n. 1, p. 103–112, jan. 2005. ISSN 0920-5691. Disponível em: <<https://doi.org/10.1023/B:VISI.0000042993.50813.60>>.
- 59 VALEM, L. P.; PEDRONETTE, D. C. G. Unsupervised selective rank fusion for image retrieval tasks. *Neurocomputing*, v. 377, p. 182 – 199, 2020. ISSN 0925-2312. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0925231219313335>>.
- 60 PEDRONETTE, D. C. G.; VALEM, L. P.; TORRES, R. da S. A bfs-tree of ranking references for unsupervised manifold learning. *Pattern Recognition*, v. 111, p. 107666, 2021. ISSN 0031-3203. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0031320320304696>>.
- 61 PEDRONETTE, D. C. G.; TORRES, R. da S. Shape retrieval using contour features and distance optimization. In: *VISAPP (2)*. [S.l.: s.n.], 2010. p. 197–202.
- 62 He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 770–778.
- 63 SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V.; ALEMI, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2017. (AAAI'17), p. 4278–4284.

- 64 LANG, K. Newsweeder: Learning to filter netnews. In: *Proceedings of the Twelfth International Conference on Machine Learning*. [S.l.: s.n.], 1995. p. 331–339.
- 65 DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <<http://archive.ics.uci.edu/ml>>.
- 66 DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255.
- 67 VALEM, L. P.; PEDRONETTE, D. C. G. a. An unsupervised genetic algorithm framework for rank selection and fusion on image retrieval. In: *Proceedings of the 2019 on International Conference on Multimedia Retrieval*. New York, NY, USA: ACM, 2019. (ICMR '19), p. 58–62. ISBN 978-1-4503-6765-3. Disponível em: <<http://doi.acm.org/10.1145/3323873.3325022>>.
- 68 SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 24, n. 5, p. 513–523, ago. 1988. ISSN 0306-4573. Disponível em: <[http://dx.doi.org/10.1016/0306-4573\(88\)90021-0](http://dx.doi.org/10.1016/0306-4573(88)90021-0)>.