UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXIII

Neural network embedding: representation learning and latent knowledge extraction for data mining applications

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutor:
Chiar.mo Prof. Agostino Poggi

Co-advisor:
Chiar.mo Prof. Panos M. Pardalos

Dottorando: Gianfranco Lombardo

Anni 2017/2020

If we knew what it was
we were doing,
it would not be called research,
would it ?
- Albert Einstein -

Abstract

Big data represents what machine learning models need to learn concepts and tasks, providing enough generalization margins. Moreover, it can also be useful for data mining applications when the goal is to extract latent and valuable knowledge. The nature of data we can collect every day is unstructured: no predefined schema is present. Some examples are documents, messages, interactions in social media platforms and in the e-commerce web sites. Indeed, the unstructured nature of these data, inevitably, adds more challenges in the previously reported cases. Machine learning models need input data in real-valued vectors (feature or design matrix) for supervised and unsupervised learning. The same issue comes back again when we are interested in finding quantitative information (e.g., the similarity between two or more documents or measure any statistics). Another interesting case is the one related to network science techniques to analyze data having (or looking for) a graph structure. Several systems can be efficiently described as nodes that interact with each other (e.g., social networks, recommendation systems, interaction among proteins, financial market). In this case, we cannot apply machine learning techniques directly because of the lack of a vector representation.

The advances in the neural networks field, enable to learn feature vectors directly from the input data distribution. This task can be seen as a pre-processing step in a modern machine learning project and involve machine learning itself. This automatic feature extraction is also known as “Representation learning”. We can summarize it as learning a vector representation of input data in a supervised or unsupervised way.

We can learn these representations or embeddings for different kinds of data: from words, entire documents, nodes or edges in a graph, images, and signals.

Moreover, embeddings encode data preserving and providing more information, for example, semantic similarity for words that will be close in their vector representation.

The choice of how to achieve this preliminary task influences all of the next stages in a typical data-mining pipeline. The challenge consists into finding a way to preserve much information as possible when looking for a structure. Secondary, with the promising results achieved recently, it is interesting to exploit these resulting vector spaces also to make knowledge extraction.

In light of the current challenges and advances in the field of Representation Learning with unstructured data, my research activity has been focused on this topic. In particular, in this thesis are reported the achieved results in two main directions:

- **Using representation learning techniques based on neural networks to extract latent knowledge from documents**: If neural networks can capture fundamental aspects among data by learning a different representation in the output layer and considering that this representation makes the classification or clustering easier, can we exploit these techniques to extract new knowledge? Part of my research tries to answer to this question. In particular, two uses cases are reported: the first analyzes scientific documents from the public repository Scopus combining word embeddings and Human mobility metrics. The second one regards neural network embeddings to extract knowledge from financial reports of thousands of american companies in the stock market.

- **Overcome current limits of representation learning on graphs**: Machine learning models can benefit from input data derived from graph structures. However, to apply most of the available models it is necessary to get a vector form for nodes and edges. The most promising way is related to the neural

network embedding and the state of the art is represented by the Node2vec algorithm. However, there are still two open problems in this field: scalability (learning a representation in large-scale graphs) and the lack of support of dynamic contexts: if a new node joins the network it is necessary to compute again the representation of the entire graph. Part of my doctorate tries to address these two problems. A first contribution is represented by an actor-based version of Node2Vec that overcomes scalability issues by distributing the bottlenecks with agents that organize themself with different behaviors to achieve the embedding in large-scale graphs. A second contribution is related to the development of a novel algorithm for incremental feature learning over graphs. The algorithm exploits properties of scale-free graphs to encode new nodes without recurring to a re-train of the model over all the nodes. It computes a light embedding over 20% of nodes with the highest degree, and then it performs a supervised alignment by solving the orthogonal Procrustes problem.

Acknowledgments

First of all I want to thank my advisor prof. Agostino Poggi for never letting me lack his support during my doctorate and for believing in me and in some of my crazy ideas.

I also want to thank in particular all the other members of my laboratory: Prof. Monica Mordonini, Prof. Michele Tomaiuolo and Prof. Stefano Cagnoni, for their encouragement and trust, but also for the insightful comments that guided my research.

A sincere thanks also goes to Paolo, Giulio and Laura for their friendship despite the short but intense period of research we spent together. They gave me the opportunity to work on several projects, expanding my knowledge and curiosity.

My most sincere gratitude also goes to Prof. Panos M. Pardalos, co-advisor of this thesis, who by inviting me to the University of Florida contributed to making one of my dreams come true by giving me a unique opportunity for experience and growth.

My thanks also go to George for being a fantastic colleague during the American period and for the many afternoons spent until the evening in the laboratory discovering the secrets of financial reports.

Tina, Michele, Kiran, Seonho and Arsenios for the time spent together,trips and evenings that made my American experience unforgettable.

Gaia for her love and for the incredible support in all difficult times. Without her, the PhD experience would have been less unique.

Last but not least, my family. Thank you for supporting me in reaching this new milestone. My gratitude can only be infinite. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

Part I

Fundamentals

Chapter 1

Introduction

Data represent the common factor for most of the main technological phenomena in the last decades. We are going through an utterly data-oriented society with an increasing demand for solutions to manage, collect and exploit this new form of richness. Indeed, being able to analyze the so-called Big Data represents at the same time a challenge but also a need for several verticals, in the industry as for governments, institutions and public services.
The main reason underlying this development is related to the digitalization of services, the spread of the Internet, and the advent of social media. All of these factors lead to rapid daily growth in data generation. According to the International Data Corporation (IDC), the average amount of data generation is close to 3 quintillions ($3x10^{18}$) bytes per day [5]. Among these, the World Economic Forum (WEF) reported that most are the consequence of humans interactions, sale of services, and 10 from embedded systems. In particular, WEF estimated that every 60 seconds (see the infographic in Figure 1.1), there are:

- 188 million emails
- 18 million text messages
- 41 million Whatsapp messages
- 1 million logins on Facebook,

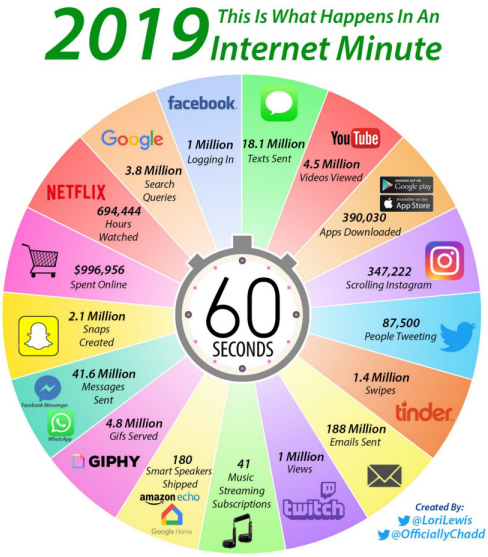


Figure 1.1: Average data generated each 60 seconds in 2019 according to the World Economic Forum. Infographic from Lori Lewis and Chad Callahan (Visual Capitalist)

- 347 thousand scrolls on Instagram
- 390 thousand apps downloaded from stores
- almost 900 thousand dollars spent on e-commerce [6].

Exploring the nature of the daily generated data is immediately clear that most of these do not present any kind of schema or model to be managed and exploited. Most of them are texts or the result of an interaction (e.g., scrolls on a platform, likes on Social media, etc.). However, they are rich in information when building a predictive system, a recommendation system, or finding latent relationships. However, dealing with data without a structure is not an issue only for web platforms, but also for other contexts. For example, in the scientific domain, an open problem regards the ability to automatically categorize publications depending on their topic or link an author to his specific research subject. During the first stage of the Covid-19 pandemic, a strong need for instruments to extract knowledge and differences among daily papers

about the virus has emerged to help researchers. In finance, finding latent information from thousands of reports might mean understanding a company's real health condition and avoiding fraud or negative investments. Think of the Enron company's famous failure that was predictable by analyzing the content and the social graph of emails exchanged among its employees. Tens of other application examples are possible from marketing to national security. It bears witness the increasing interest in this topic, especially with the novelties in the machine learning domain that require a large amount of data that are usually collected in an unstructured form. However, an open issue is present: how to represent information and knowledge so that statistical models can leverage despite the lack of a structure? Several attempts have been proposed in these years, and the most promising are related to the concept of semi-structured data and unsupervised machine learning techniques.

1.1 Unstructured and semi-structured data

It is possible to categorize data in three different ways, depending on how the information is stored and organized. When a data model is available and data are organized according to it, it is said that data are structured. A data model is an abstraction that determines how things are related to each other, providing at the same time a logic standardization that makes explicit the main properties of the real-world entities linked with data. Examples of structured data can be the ones stored in relational databases. When a structure is missing, we define data as unstructured. In this case, the information expressed is almost qualitative and is complex to retrieve knowledge directly. Examples are represented by texts, images, social media content, videos, and music. Finally, it is possible to count a third category that is the most common in the big data context: semi-structured data.

This last group contains data that we might include in the former categories in the first instance but with a certain amount of uncertainty. That is because although they don't have a rigid structure to be organized in, at the same time, it is possible to exploits metadata or relationships that make these not exactly unstructured. Several

examples are possible, depending on the case. Digital photographs are usually unstructured, but information concerning time and place of creation, or the author's name, can be used to organize them.

Another case regards documents that are usually unstructured. However, categorization is possible when metadata is available, such as the topic, the owner's name, etc. Finally, most user-generated content on social media can be semi-structured by considering the user's profile or considering the additional information provided by the topology of its social network (e.g., Mutual friends on Facebook, sharing on Twitter, comments, and replies in discussions). This last example covers an important set of instruments when dealing with semi-structured data, which are network science techniques to find a structure among tons of unstructured data in the form of a graph with nodes and links.

1.2 Main challenges

Big data's wealth of information represents what machine learning models need to learn concepts and tasks, providing enough generalization margins. Moreover, it can also be useful for data mining applications when the goal is to extract latent and valuable knowledge. However, the unstructured nature of these data, inevitably, adds more challenges in both cases. Indeed, machine learning models need input data in numerical vectors (feature spaces) for supervised and unsupervised learning. The same issue comes back again when we are interested in finding quantitative information (e.g., the similarity between two or more documents or measure any statistics). Another interesting case is the one related to network science techniques to analyze data having (or looking for) a graph structure. Several systems can be efficiently described as nodes that interact with each other (e.g., social networks, recommendation systems, interaction among proteins, financial market). In this case, we cannot apply machine learning techniques directly because of the lack of the euclidean structure.

In the past, a way to overcome these issues was to perform manual feature engi-

neering, which requires the use of domain knowledge to extract features from raw data with data mining techniques. The advances, especially in the neural networks field, enable generalization and automation of this task by learning features directly from the input data distribution. This task can be seen as a pre-processing step in a modern machine learning project and involve machine learning itself. This automatic feature extraction is also known as "Representation learning". We can summarize it as learning a vector representation of input data in a supervised or unsupervised way. Moreover, the vectors' dimension is usually a hyper-parameters that can be chosen arbitrarily. We can learn these representations or embeddings for different kinds of data: from words, entire documents, nodes or edges in a graph, images, and signals.

However, the resulting embeddings do not have to be confused with only a way of encoding unstructured data (like the Bag-of-words model). Indeed, embeddings encode data preserving and providing more information, for example, semantic similarity for words that will be close in their vectorial representation.

In conclusion, when dealing with unstructured data is often crucial to perform an auxiliary task of representation learning to get an exploitable form of data. The choice of how to achieve this preliminary task influences all of the next stages in a typical data-mining pipeline. The challenge consists of finding a way to preserve much information as possible when looking for a structure. Second, with the promising results achieved recently, it is interesting to exploit these resulting vector spaces also to make knowledge extraction.

1.3 Overview of this thesis

In light of the current challenges and advances in the field of Representation Learning with unstructured data, my research activity has been focused on this topic. In particular, in this thesis are reported the results achieved in three main directions:

- Using representation learning techniques based on neural networks to extract latent knowledge from documents. In particular, two use cases are reported: the first analyzes scientific documents from the public repository Scopus combining word embeddings and Human mobility metrics. The second one regards neural network embeddings to extract knowledge from financial reports of thousands of American companies in the stock market. This last one is a joint work with the University of Florida, where I spent my abroad period of research.

- Overcome computational limits of representation learning on graphs: A distributed version of the famous algorithm Node2Vec has been developed to overcome mainly issues related to representation learning over large networks

- Development of a novel meta-algorithm for feature learning over dynamic power-law graphs. One limit of the current approaches is that they are not suitable for dynamic and temporal graphs because they are intrinsically transductive. When a new node joins the network, in order to get its representation, it is necessary to train a new model over the entire graph. Thus, to take into account dynamic contexts, other approaches need to be developed. My contribution in this field regards a meta-algorithm that exploits properties of scale-free graphs to encode new nodes without recurring to a re-train of the model over all the nodes. It computes a light embedding over 20% of nodes with the highest degree, and then it performs a supervised alignment by solving the orthogonal Procrustes problem. It is a meta-algorithm since it can leverage initial static embeddings coming from different algorithms to achieve a representation in a dynamic context, as it will be presented later in section 7.2

Chapter 2

State of the art and related work

The way information is represented has a large impact on the type of operations that are easy or practical to perform with it. Several examples can be presented: for example, it is easier to achieve a division with Arabic numbers rather than with Roman ones. Decomposition in prime factors allows understanding a number's components to perform operations or simplify a calculus. Finally, it impacts how data can be linearly divided into clusters or sets to perform classification tasks and how dimensions can be reduced for further analysis or data visualization.

The performance of machine learning methods is heavily dependent on the choice of data representation on which they are applied. Thus, most of the effort in deploying this kind of technology goes into designing methods that can improve representation preserving much information as possible. In that sense, most of the machine learning algorithms are weak and cannot extract and organize the discriminative information from the input data, so it is often required manual extraction of features. On the other hand, artificial neural networks work precisely in the opposite direction.

2.1 Desiderata of a good representation

Machine learning models learn from a set of observations that describes the considered domain's relevant characteristics, also known as features or design matrix. The main assumption is that we would like to get, ideally, a model that catches the reasons underlying the data generation source from these observations. Practically, we want to learn the joint distribution $p(X|Y)$, where X is the feature set, and Y is the prediction's goal. Once we get this distribution, we can compute the probability $p(y|x)$ where y and x are elements of the labels set and design matrix, respectively. In light of this, the feature we choose to describe the domain acquire an important role. For many kinds of data, a vector representation is not directly available: for example, words, documents, manifolds, and graphs. Thus, we need to encode these data in some way. One approach, very popular until a few years ago, is feature engineering: a manual extraction of features. However, this approach has different limits:

- A good domain knowledge is required
- Difficulties into understanding which features are relevant for the task or not (feature selection)
- Difficulties for transfer learning: reusability of features for a different task starting from a similar distribution of input data

On the opposite side, another approach that grew recently, especially for the advance in the neural network field, is to learn a representation directly from data by solving an auxiliary task: the so-called Representation Learning. The main desiderata of a good representation of unstructured data:

- Dense representation: We would like to get a vector representation, but at the same time, we do not want that each component describes a single aspect. When we learn from data, we cannot have prior knowledge of the number of relevant characteristics. We would like to select an arbitrary dimension and being able to compress relevant information in a distributed way across these components. Within a dense representation, we can encode easier a higher number of features keeping the vector dimension reasonable.

- Easier separability of data: The new representation should make the next classification tasks easier, like any other operations we are interested in.
- Information preserving: The learned representation has to ensure information preservation, ideally, without loss. For example, we want to encode words by taking into account the context of the words.
- Representation learning should be unsupervised

Neural networks enable unsupervised Representation learning. Indeed, this automatic feature learning ability of ANNs is the main advantage of this model to the other machine learning algorithms. Some of these requirements are also ensured by different machine learning techniques like Principal Component Analysis (PCA) or the clustering algorithm K-means. However, only neural networks currently can satisfy all of them at the same time. This fact is motivated by the following other approaches limits:

- Non-dense encoding: K-means is an unsupervised method that produces a cluster label for each input with usually a binary encoding vector with all zeros and one in the element corresponding to the associated cluster. The same problem will be further analyzed in the context of text-mining with the Bag-of-words model.
- Dense-encoding with loss of information: The encoding produced by PCA, by contrast, is a dense encoding that maps the input X to its projections into K different directions. It is unsupervised so it does not take into account linear separation among classes. It is based on matrix factorization and it can work only with already real-value data. Indeed, it is almost known to reduce dimensions of representations rather to learn one.
- Not suitable for transfer learning
- Representation is gathered by transformation but it is not learned to make a classification task easier.

2.2 Representation Learning with Neural networks

Artificial neural networks (ANN) can be thought of as universal approximators of functions that can learn a function that is able to map input data X to a continuous or discrete variable Y. These networks are composed by unit called artificial neurons that are organized in layers Figure 2.1.

The approximated function is parametric and depends on the weights w of the network. These weights define a multiplier for each connection among two neurons and are the items that are learned solving an optimization problem with usually the Back-Propagation algorithm [7]. Generally speaking, we can say they approximate non-linear relationships between input and output by learning latent representations in the hidden layers [8].

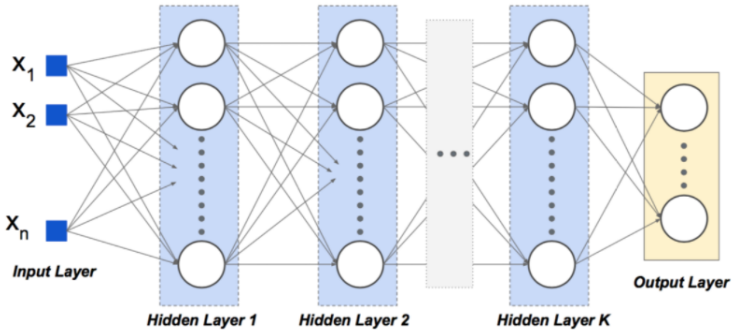


Figure 2.1: Example of artificial neural network's structure.

Specifically, the last layer of the network is typically a linear classifier, such as Softmax regression classifier [9]. Supervised training leads to a representation of the input at each layer, but the one in the last layer takes the properties that make the classification task easier [10]. In other words, classes that are not linearly separable in the input features may become linearly separable in the last hidden layer. Feature learning with neural networks enables also to reusability. Indeed, features learned by a neural network can also be used for Transfer learning: different supervised tasks

starting from the same input data. This last procedure is beneficial when data for one task is less than the necessary ones to train a new model.

Moreover, artificial neural networks can learn a dense representation starting from manifolds, text and graphs that have been previously encoded in some way (e.g., one-hot encoding), but adding more information to the representation by learning features with an auxiliary task. As it will be presented in the next sections, the success of the famous word embedding algorithm Word2Vec relies on training a model to predict the missing word in a context or vice-versa.

Despite all these premises, neural networks can learn a representation as a side effect of optimizing an objective function combining different non-linear layers. The recent results with deep learning architectures have powerfully demonstrated their ability to learn useful representations starting from raw data (e.g., Convolutional neural networks in Computer vision or Recurrent neural networks for speech recognition and translation [11],[12]. Moreover, it is not often required to design and train complex deep neural networks for this goal. In fact, one way relies on the use of shallow neural networks or auto-encoders by designing the auxiliary task accurately to learn good features [10],[2]. This last one is often referred in literature as unsupervised learning, although in this case, it has a meaning of learning as a result of another task.

The success of these models in feature learning also enables one of the research questions of this thesis: If the model can understand latent relationships among data, is it possible to use this representation to extract knowledge?

2.3 Word embedding

When dealing with words, the main problem is that they do not rely directly on real value feature vectors. Thus, an encoding of words is necessary to get the encoding of a document to apply later statistical learning algorithms. One of the most used techniques in the past is the Bag-of-word model with one-hot encoding. A dictionary

is built with all the words in the analyzed documents. If we have *n* different words, we can encode them with n-dimensional vectors that are zeros vectors except for the word's index in the dictionary (Figure 2.2). The same happens when we want to encode a document that will be an n-dimensional vector with ones corresponding to the document's words (Figure 2.4).

$$Vocab = \begin{bmatrix} \text{I} \\ \text{love} \\ \text{like} \\ \text{cats} \\ \text{dogs} \end{bmatrix}$$

$$v(\text{I}) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, v(\text{like}) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, v(\text{dogs}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Figure 2.2: Bag-of-words example for words encoding.

Figure 2.3: Example of document encoding with the Bag-of-words model

Bag-of-word is a sparse encoding but also a local representation. Indeed, we are using single components to describe each entity. This approach is thus inefficient in

terms of memory, but it also lacks a way to represent useful information. For example, words always have the same distance from each other. Context and order are not taken into account, and inevitably, there is a loss in terms of semantic similarity detection. This model can be improved by using techniques that allow a better valorization of words in the vectors using co-occurrences information, such as frequency-inverse document frequency (TF-IDF). However, the model remains a local representation, although it can seem a dense representation. The difference between local representation and distributed representation is a key-factor in representation learning.

Distributed representation concept has firstly introduced by Hinton in the early nineties [13]. To explain this concept in particular in the neural network's field, it is better to highlight the contrary: In non-distributed or local representation, each possible value has a unique representation slot. It is memory consuming, and usually, the resulting vector is sparse. Sparsity is a disadvantage since, for example, words with close meaning do not have closer representations. Despite that, distributed representation describes the same data features across multiple and scalable components. These components are usually implemented as the corresponding weights in neural networks' layers. Each layer defines the information with the same accuracy level, but adjusted for the level of scale. These layers are learned concurrently but in a non-linear way. This mimics human logic since each concept can be accessed by more than one neuron firing, and each neuron can represent more than one concept [13].

Word embedding techniques can overcome the limits of sparse encoding and local representation by learning vector descriptions of words in an unsupervised way using distributed representation and simple neural networks. The first exciting attempt in this direction was the Neural Language model from Bengio et al [1]. On this basis, a novel algorithm was introduced by Google in 2013: Word2vec [2].

2.3.1 Neural Language model

The main goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. However, in order to achieve this result,

it is necessary to deal with the so-called curse of dimensionality because of a large number of discrete variables results in exponentially large free parameters.

Indeed, a word that has never been seen during training could be very different from all the word sequences already seen. Traditional approaches are based on n-grams that concatenate short overlapping sequences seen in the training set. One way to solve this problem consists of learning dense and distributed vectors of features from each word. In this way, generalization is possible when new sequences have a nearby representation with words already seen.

Dealing with this problem, in 2003 Bengio et al proposed the Neural Language model [1]. This model simultaneously learns a distributed representation for each word and the probability function for the entire sequence with these representations. In particular, to learn a representation of a word is used a context window (a number of words surrounding from left and right the target one in a sentence). Each word is described as a point in a vector space. The probability function is expressed as a product of conditional probabilities of the next word given the previous ones using a neural network. Features vectors associated with each word are learned, maximizing the log-likelihood of training data.

The objective of the model is to estimate the parameters that minimize the perplexity of the training set. Where the perplexity represents the required dimension to encode a word. The Neural model has a hidden layer with tanh activation and the output layer is a Softmax layer. The output of the model for each input of (n-1) prev word indices are the probabilities of the $|V|$ words in the vocabulary.

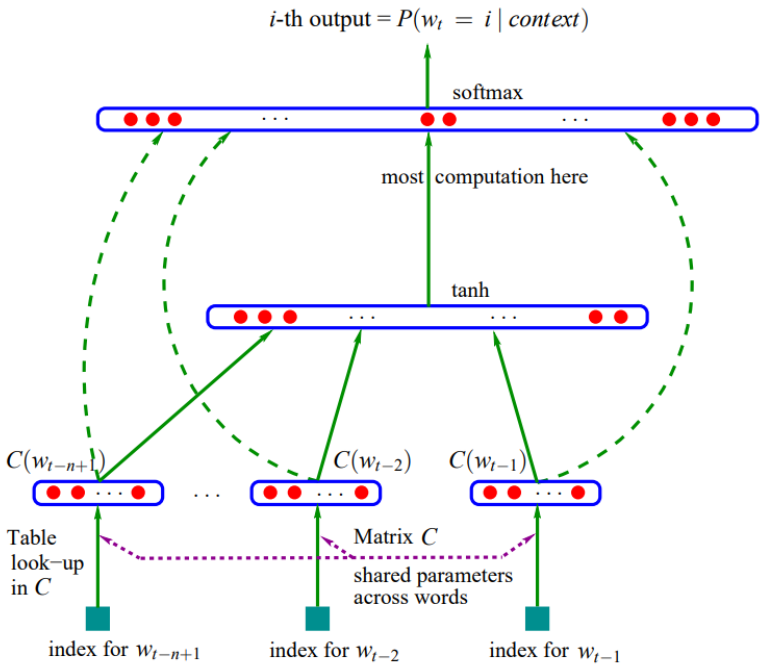


Figure 2.4: Neural architecture from Bengio et al [1]

2.3.2 Word2Vec

The main limit of the Neural Language model is related to the computational complexity since it involves non-linear projections of each word and different epochs of training. Indeed, this model is unable to process more than a few hundred million. For many tasks, this is a bottleneck since the corpus's size usually dominates the performance.

In [14], Mikolov et al proposed techniques for measuring the quality of the resulting vector representation and using linear projections to simplify the neural network. In particular, they require the expectation that not only will similar words tend to be close to each other, but that words can have multiple degrees of similarity. Among these degrees, a surprising result is that using a word offset techniques, it is possi-

ble to implement algebraic operations on word vectors. A famous example is that vector("King")- vector("Man")+ vector("Woman") results in a vector that is closest to the vector("Queen").

In light of this, Mikolov et al in 2013 proposed two novel architectures to learn vector representation from words that are the basis of the well-known Word2vec algorithm [2]. The two model architectures are the Continuous Bag-of-Words model (CBOW) and the Continuous Skip-gram model. They are both based on auto-encoders with linear activation function in the unique hidden layer. Thus, the projection matrix is unique and shared among words. Moreover, to learn a representation of a word, they consider past and future as a window on a context: some words from the left and an equal number from the right side. The scheme of the two architectures are presented in Figure 2.5.

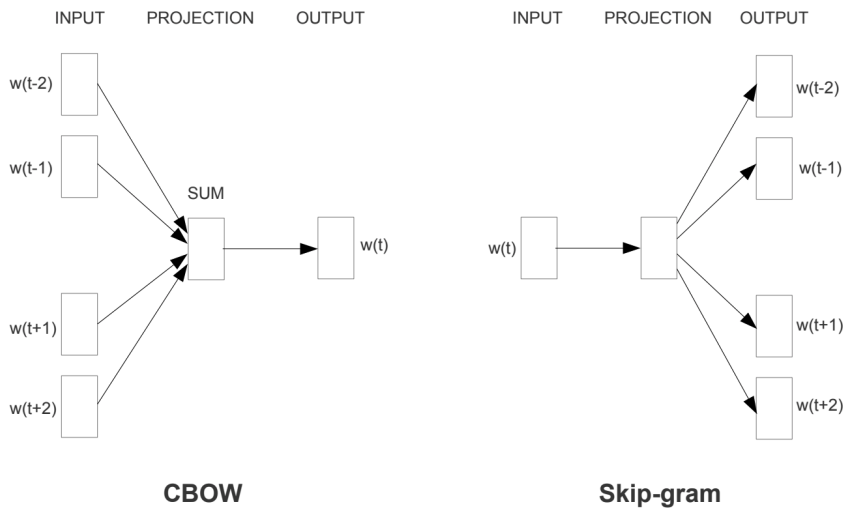


Figure 2.5: The architectures available in Word2vec. Source: [2]

The main difference is that CBOW takes the context as input and try to predict the missing word, while the Skip-gram tries to predict the surrounding context from an input world. The two models are linear feed-forward neural networks trained with

stochastic gradient descent and the back-propagation algorithm.

The CBOW model tries to predict a word given its context. It is called bag-of-words since the projection is not dependent on the order of the words in the history. The Skip-gram model is trained to predict neighbor words in the sentence. Here, since the more distant words are usually less related to the current word, the more distant the words are, the less they are sampled. The output layer is implemented with a hierarchical Softmax function. Values from the hidden layers are then the resulting word embedding vectors. This means that syntax and semantics are captured as the indirect result of predicting the next word in a sentence. In the Skip-gram given a sequence of training words $w_1, w_2, w_3, ..., w_T$, the objective is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p\left(w_t|w_{t-k},\ldots,w_{t+k}\right)$$

where:

$$p\left(w_t|w_{t-k},\ldots,w_{t+k}\right)=\frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

In other words, it learns a word feature vector by predicting its context in a window of surrounding other words preserving the order.

Each output of y_i is the unnormalized log-probability for each output word i and it is computed as:

$$y = b + Uh\left(w_{t-k},\ldots,w_{t+k};W\right) \tag{2.1}$$

where U, b are the softmax parameters. h is constructed by a concatenation or an average of the word vectors extracted from W. On the other hand, the CBOW learns feature vectors by predicting a word missing from its context. For this reason CBOW results to be faster than Skip-gram but less accurate in capturing some semantic aspects of the words.

Moreover, we can say that is one-shoot learning because only one epoch of training is required. Word2vec is presented as a neural network, but it can also be seen as

a simple matrix projection with elements learned by optimization. This simple architecture is the main advantage of this algorithm in terms of computational complexity. Finally, the high quality of word vectors is because the network is trained with an excellent auxiliary task. Similar words have similar representations when they co-occurred in a similar context. This kind of simple architecture is also called shallow neural networks.

Results of Word2Vec training can be sensitive to parameterization. The following list includes the most important parameters is very important in Word2Vec training:

- `size`, which defines the dimension of the feature vectors;
- `min count`, which defines the minimum number of occurrences below which words are not considered in the model;
- `window size`, which is the maximum distance between the current and the predicted words within a sentence. Each window is centered on a word in the sentence and it considers the same number of words from the left side and the right side, when it is possible.

2.3.3 Doc2Vec

Once we learned feature vectors for words in a document, we might desire to perform several document tasks. Some examples are topic detection and document similarity estimation. However, to achieve this result, especially for the document similarity, we need to find a way to get unique descriptions for the documents. One idea is to average or sum all word vectors to get a document vector. However, this approach can be expansive, particularly when long documents are considered and thinking that learning word embeddings is already computationally challenging in this case.

Another strategy consists of learning a document vector directly while learning word vectors. Indeed, in [3], Le et al came up with an extension of Word2Vec that is directly able to learn feature vectors for documents in an unsupervised way: Doc2Vec. This algorithm learns distributed vector representations for paragraphs, regardless of their length, while learning word feature vectors.

Practically, it exploits the same logic as the Word2Vec architecture but it introduces also the concept of the paragraph token. This token acts as an additional word and as memory to remember what is missing from the current context. Thus, the paragraph token has an associated paragraph vector to be learned. This vector is shared with all the windows context that the algorithm uses to learn the other word embeddings for the same document.

Instead each word feature vectors that compounds the W word vector matrix is shared across the other paragraphs. Also, in this case, Doc2Vec provides two different architectures, Figure 2.6. The first, Distributed Memory version of Paragraph Vector (PV-DM) is an extension of the CBOW model in Word2Vec. The only thing that changes in compared to the Word2Vec architecture is in equation (2.1), where h is constructed from both W and D (the document matrix that contains all of the paragraph vectors). Then it takes the concatenation of W and D to predict the next word.

The other version, as an extension of the Skip-Gram is the Distributed Bag of Words

version of Paragraph Vector (PV-DBOW). In both architectures, for each document vectors are unique, while the W are shared. Each document is mapped to a unique vector represented by a column in matrix D and each word is also mapped to a unique vector, represented by a column in matrix W. The D and W are concatenated to predict the next word in the context.

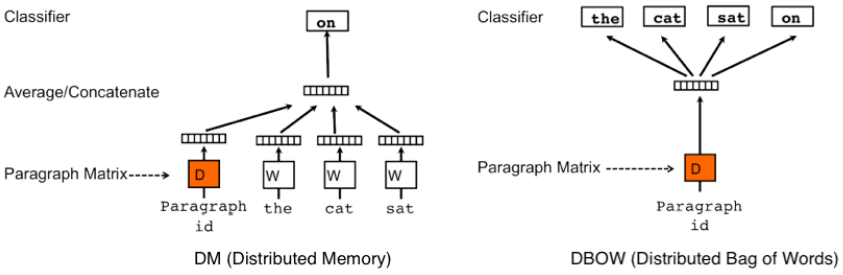


Figure 2.6: The architectures available in Doc2vec. Source: [3]

2.4 Graph embedding

For decades, graph-data study has been limited to analysis of the network topology with structural metrics aiming to extract connectivity patterns among the system components. More recently, with the progress of Machine Learning techniques, the idea of taking advantage of this kind of structure also emerged to perform prediction tasks or clustering. For example, in [15], the authors modeled the interaction between proteins as a network, intending to automatically predict a correct label for each protein describing its functionalities. In [16] the authors use a temporal network to model the US stock market to discover correlations among the dynamics of stocks' clusters and to predict economic crises. In [17] the authors analyze a social group of patients to extract new knowledge about their emotional state and temporal disease pattern by modeling them in two attributed networks: an interaction network and a friendship network.

However, the application of Machine Learning directly on graph-data is challenging

because of the necessary manual feature extraction. This limit leads to approaches that study the application of machine learning models directly on graphs. In particular, two approaches in this direction are currently in state of the art:

- **Ad-hoc learning models for graph**: Building new learning models that can directly take a network as input and are specialized in prediction tasks on graph-data. The Graph Neural Network Model represents the most important advances [18]. It makes a mapping of the network topology as units of a Recurrent Neural Network and Graph Convolutional Network [19] that takes the adjacency matrix and nodes specific features as input to learn prediction tasks by using different kinds of convolution operations in the graph domain.

- **Embedding techniques for non-euclidean data**: The idea of finding a latent vector representation that can capture the key features of the data is common in different Data mining and Machine Learning tasks (e.g Natural Language Processing, Computer Vision, Dimensionality Reduction). This approach's main reason lies in the need of using well-known machine learning models (e.g SVM, decision trees, regression models, and neural networks) that require a euclidean feature representation.

Graph embedding techniques can be divided in two general sectors: ***Factorization based Methods*** and ***Random walk based methods***. Algorithms of the first case obtain the embedding by factorizing a specific matrix based on the graph, like the Adjacency matrix, Laplacian matrix or the node transition probability matrix. Approaches to factorize the representative matrix vary based on the matrix properties. If the considered matrix is positive semi-definite, e.g. the Laplacian matrix, one can use eigenvalue decomposition. In other cases, it is possible to factorize using gradient descent methods. Some examples of this kind of embedding algorithm are LLE [20] which preserves first-order proximity, and LINE [21], which also preserves the second-order proximity in the graph. Instead, the second case algorithms obtain a graph embedding by generating walks on the edges randomly and processing them using different learning models to extract and preserve key features of the graph. The main advantages of

random walks are that they can approximate many properties in the graph, including node centrality and similarity, and easy generation. This approach is gaining interest for its simplicity and thanks to the adoption of the Skip-gram model [22] as a learning model, largely used already in text embedding. This approach's key innovation is optimizing the node embeddings so that nodes have similar representations if they tend to co-occur on short random walks over the graph. To the best of our knowledge the algorithm at state of the art in graph embedding is Node2Vec [23] which is a random walk based algorithm

This context can be seen as very similar to the one in text mining where the same problem has been addressed by performing a representation learning task over sentences, called Embedding, in order to learn a vector representation of each word. In light of this, several works have highlighted the importance of performing a similar task with the same models also to extract automatically features from networks that can preserve local and global information of nodes.

According to [24],the most common use cases for node embeddings are:

- **Visualization**: The problem of visualizing graphs in a 2D space is not recent with several other approaches presented in the past. Node embeddings offer a powerful new paradigm for graph visualization: because nodes are mapped to real-valued vectors that can be combined with generic techniques for visualization high-dimensional datasets (e.g., t-SNE or PCA)
- **Clustering and community detection**: Node embeddings are a powerful tool for clustering related nodes, with several applications from computational biology to marketing (e.g., discovering related products). It is possible to apply any generic clustering algorithm to the set of learned node embeddings (e.g., K-means).
- **Node classification and semi-supervised learning**: It is often the benchmark task to evaluate the quality of embeddings. The goal is to associate a label to each node in the graph with supervised or semi-supervised learning (where

labels are only available for a small proportion of nodes, with the goal being to label the full graph based only on this small initial seed set)

- **Link prediction**: The goal is to predict missing edges or edges that are likely to form in the future. This task is at the core of recommender systems (predicting missing friendships in social media platforms or suggest a product or a movie to a client). Finally, it is interesting for the prediction of missing relations between entities in knowledge graphs.

2.4.1 DeepWalk

DeepWalk aims to learn the latent representation of nodes in a network in a continuous vector space that can be easily exploited by traditional Machine Learning models. This algorithm generalizes the advancements in word embedding obtained with Word2Vec [2] by combining its neural language model, known as Skip-gram [22] and the generation of truncated random walks on the network that play a role equivalent to sentences in text mining. To achieve this result, it optimizes a neighborhood preserving objective function using the Stochastic Gradient Descent with a Hierarchical softmax in the form of a shallow Neural network. More details about the optimization function are discussed in the next section because also Node2Vec is based on the same function but with a different generation policy of random walks. From a computational point of view, DeepWalk results in a lightweight algorithm in terms of required memory for two reasons. The first one is that random walks are generated, considering only edge weights and a uniform probability distribution. The second one is that to avoid memory limits, it implements a streaming strategy to deal with large networks. However, this last aspect is responsible for the growth in time complexity and less representation quality. The main parameters of the algorithm are the number of dimension for the embedding space d, the number of walks to be generated for each node γ and the walk length ω.

2.4.2 Node2Vec

Node2Vec [23] extends the previous node embedding algorithm DeepWalk. From the last one, Node2Vec retrieves the Skip-gram model [22] and the idea of learning embeddings by analyzing the relationships among the entities in the form of sequences and the use of a shallow neural network model. Indeed, Word2Vec learns word embeddings by analyzing the context of each word in a corpus. It predicts the current word from a sliding window of surrounding context words. The key idea of applying this model in the graph domain is to build something similar to a text corpus (a set of word sequences) by performing random walks. The result can be thought of a set of word sentences where words are substituted with nodes that compose the walk. The innovation of Node2Vec with respect to DeepWalk is in the method that is used to build biased random walks on the network. In the previous one, in fact, random walking is obtained by a uniform random sampling over the linked nodes, while Node2Vec combines two different strategies for the network exploration: Depth-First-Search (DFS) and Breadth-First-Search (BFS), see Figure 2.7. It also uses a second-order Markov chain and the Alias sampling to select the next node to be visited in the walk. In order to learn latent representations, Node2Vec optimizes a neighborhood preserving function by also redefining the concept of the neighborhood as a set of visited nodes depending on the combination of BFS and DFS. The considered optimization problem is the following:

$$\max_{f} \sum_{u \in V} \log Pr(N_s(u)|f(u)) \tag{2.2}$$

where f is the function that maps each node $u \in V$ in the vector space and $N_s(u)$ is the Neighborhood of node u, sampled with a combination of the two exploration policies cited above. The algorithm's main parameters are the same as DeepWalk with two other more to manage the network exploration: P (or Return parameter) that controls the likelihood of immediately revisiting a node in the walk; Q (or In-out parameter) to manage the BFS. The main steps of the algorithm can be summarized as the following:

1. Probabilities computation over edges with a second-order Markov chain.

2. Alias sampling and random walks generation.

3. Embedding with the Neural network model.

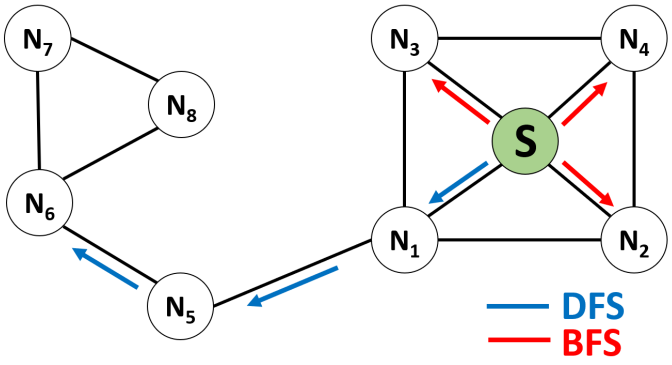


Figure 2.7: BFS and DFS search strategies from node S.

2.4.2.1 Probabilities computation

To feed the neural network for the embedding step, Node2Vec has to perform a fixed number of random walks starting from each node, with a fixed length. These walks are based on a second-order Markov chain that considers the parameters p and q to mix the two different search strategies introduced previously. The algorithm's first duty is to generate all of the required probabilities over the edges for the next random walks generation. Formally, given a source node **t**, that can be also an intermediate step for other walks, a 2nd order walk has to be generated by considering the 2nd order neighborhood of node t. Let C_0 = t and C_1 = v, where v is a first order neighbor of node t. The next edge to be traversed is chosen by the following distribution:

$$P(C_i = x | C_{i-1} = v) = \begin{cases} \frac{\omega_{vx} \cdot \alpha_{pq}(t,x)}{Z} & if (v,x) \; \varepsilon \; E \\ \\ 0 & otherwise \end{cases} \tag{2.3}$$

where ω_{vx} is the edge weight, Z is the normalizing constant and $\alpha_{pq}(t,x)$ is the search bias function. The bias function α is defined with the aim of joining BFS and

DFS strategies:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & if\ d_{tx} = 0 \\ 1 & if\ d_{tx} = 1 \\ \frac{1}{q} & if\ d_{tx} = 2 \end{cases} \tag{2.4}$$

Where d_{tx} is the geodesic distance between the starting node t and the considered node x. An example of this step is presented in Figure 2.8. At the end of this phase, each edge of the network has a set of values that describe the edge's probabilities to be traversed during a random walk, depending on the source node. This computational step represents the first bottleneck of the algorithm in terms of memory and required computational times because it depends on the dimension of the network (nodes and edges number) and on its density. How to overcome the emerging limits of this calculus, while analyzing a large network, will be further explained in Section 5.

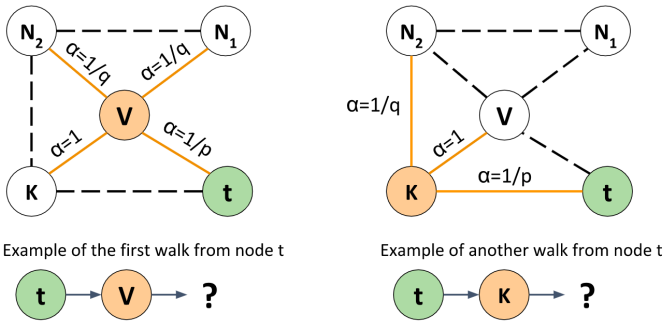


Figure 2.8: Example of the calculus of probabilities over edges with a second-order Markov chain

2.4.2.2 Generation of the random walks

Once a set of probabilities is associated with each edge depending on each node's second-order neighborhood, Node2Vec is ready to perform the random walks effec-

tively based on the number of walks and Walks length parameters. To sample the edges to be traversed from the probability distribution obtained in the previous step, the algorithm uses the Alias method [25]. This method enables the algorithm to sample efficiently from the discrete probability distribution by building and consulting two tables: the probabilities table and the alias table. The algorithm uses a system based on a biased coin to decide. At the end of this phase, random walks get collected. The collected sequences of nodes play the same role as sentences in a text mining corpus. In this analogy, sentences are built from the nodes that have been traversed by each walk.

2.4.2.3 Embedding with Neural networks

The neural network model used by Node2Vec is the same as Word2Vec [2]. The basic idea is to train a simple neural network with a single hidden layer to predict the most probable word that follows the input one, but then the algorithm does not use this model for the task we trained it on. Instead, the goal is actually just to learn the weights of the hidden layer that will represent our euclidean representation of the input. The training set is composed of word pairs (x,y) where x is the input word and y is a nearby word of x in a fixed window in each sentence of the corpus. In the case of Node2Vec the operation is the same but the algorithm uses node pairs and the corpus is composed of the previously computed random walks.

2.4.3 Open problems

Several open problems are still open in the field of graph embedding. According to [26], there are three important challenges to be dealt in this field:

- **Scalability**: Most of the available approaches are claimed to be highly scalable in theory. However, significant bottlenecks in the algorithms lead to unfeasible requirements when dealing with truly massive datasets. Most evaluation setups assume that the attributes, embeddings, and edge lists of all nodes used for both training and testing can fit in main memory, an assumption that is at odds with

the reality of most application domains, where graphs are massive, evolving, and often stored in a distributed way.

- **Modeling dynamic and temporal graphs**: Several application domains involve dynamic graphs where timing information is critical—e.g., instant messaging networks or financial transaction graphs. However, we lack embedding approaches that can cope with the unique challenges presented by temporal graphs, such as the task of incorporating timing information about edges. Moreover, the node embedding with current approaches is based on trasductive learning. Indeed, to learn about a node, the model has to see all the nodes in the dataset. When a new node joins the network, it is necessary to train the model over the entire network to get its feature vector.

- **Improving interpretability**: Representation learning is attractive because it relieves much of the burden of hand designing features, but it also comes at a well-known cost of interpretability. We know that embedding based approaches give state of the art performance, but the fundamental limitations—and possible underlying biases—of these algorithms are relatively unknown. Given the complexities and representational capacities of these approaches, researchers must be ever vigilant to ensure that their methods are truly learning to represent relevant graph information and not just exploit benchmarks' statistical tendencies.

In this thesis, two different solutions are presented to deal with the first two points. Finally, the idea of using embedding spaces to extract knowledge goes in the direction of improving interpretability in general for representation learning.

2.5 Knowledge discovery from unstructured data

Since digital documents are being produced and collected at increasing volumes, automatic knowledge discovery and data mining are becoming more and more important. In particular, in the context of Big Data mining, the ability to retrieve information, insights and values from documents, in their heterogeneous forms, is cru-

cial. This task represents a challenge for decades, with the aim to find a trade-off between automatic knowledge discovery and performance of tools and techniques able to give a productive representation of knowledge. In [27], the authors were already discussing the need for computational theories and tools to assist humans in the extraction of useful information from the rapidly growing volumes of digital data. Several attempts have been presented in literature in different domains, with a particular attention on ways to extract information from text data exploiting semantic properties. For example in [28], the authors analyzed healthcare scientific publications to understand directly which methodologies are most used in a field and what is the status of research reached to help stakeholders to plan research funding. In [29] word embedding is used for topic modeling to analyze and explain the behavior of the CiteScore metrics for journals indexed in Scopus in 2017 and to get statistics about the impact of topics. In [30], the authors demonstrate how the semantic space built with Word2Vec on large corpora, combined with network science tools (such as degrees, distances and clustering coefficient) can be relevant for knowledge discovery. In the last years, other researchers tried to address unsupervised knowledge discovery with different approaches but all based on machine learning models [31, 32, 33, 34]. Finally, in 2019, Tshitoyan et al, come up with a novel research work [4] published in Nature where they apply word2vec to the analysis of the materials science scientific literature for knowledge discovery. Indeed, publications contain valuable knowledge regarding the connections and relationships between data items as interpreted by the authors. They show that materials science knowledge present in the literature can be efficiently encoded as information-dense word embeddings.

The authors collected approximately 3.3 million scientific abstracts published between 1922 and 2018 in more than 1,000 journals deemed likely to contain materials-related research, resulting in a vocabulary of approximately 500,000 words. They use the Skip-gram model available in word2vec encoding each word in 200-dimension vectors.

They highlight that neural network embedding can learn different insights by learn-

ing word vectors in their context. They found an algebraic relationship among the material similar to the one found between words King and Queen in Mikolov et al in [14]. In particular, many words in the corpus represent chemical compositions of materials, and the five materials most similar to LiCoO2 (a well-known lithium-ion cathode compound) can be determined through a dot product (projection) of normalized word embeddings (See Figure 2.9.

In particular, these embeddings capture complex materials science concepts such as the underlying structure of the periodic table and structure–property relationships in materials. Furthermore, they demonstrate that an unsupervised method can recommend materials for functional applications several years before their discovery.

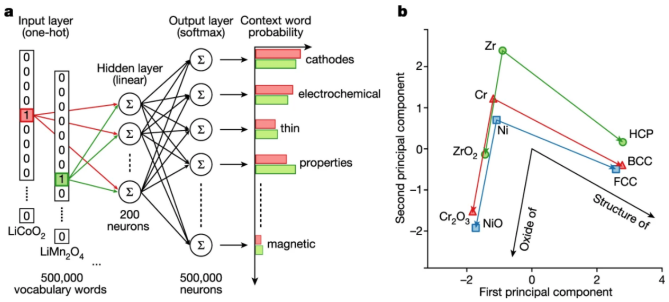


Figure 2.9: The projection found for LiCo02 [4]

Part II

Knowledge discovery with word embedding

Chapter 3

Proposed methods for Knowledge discovery

In light of the neural networks' ability to capture fundamental aspects among data by learning a different representation in the output layer and considering that this representation makes the classification or clustering easier, can we exploit these techniques to extract new knowledge? Most of my research works aim to answer this question. This sections presents two methodology contributions in this direction. In the next chapters two applications of the following methods will be presented. The first one addresses the problem of knowledge extraction from scholars' activities and bibliographic databases as a practical example of a group of agents travelling in a semantic space and how their mobility can be used to extract new knowledge. The second one is related to the analysis of financial reports (10Ks) from the main American companies in the stock market in order to evaluate how changes in these reports over the years can be used to predict abnormal returns.

3.1 Mobility trajectories in a word embedding model

In the field of big data, a great interest has rapidly emerged, concerning the study of geographic positioning and mobility information. The main contribution presented in

this section is to show how analytical tools, traditionally adopted to deal with geographic data, can be effectively applied to extract knowledge in the semantic realm, exploiting the properties of neural network embedding. In fact, word embedding algorithms such as Word2vec, allows to map textual data into points of an arbitrary multi-dimensional space, in which the notion of proximity reflects an association among terms or topics.

Although the adoption of such algorithms to mine features is, nowadays, a well-establish practice, with the following approach we want to take a step further, investigating a novel approach to discover knowledge in a target domain, based on textual big data, word embedding and mobility techniques. At the same time, the study of human mobility has become very important for applications such as estimating migratory flows, traffic forecasting, urban planning, and epidemic modeling [35, 36]. Mobility is often estimated using the so-called Radius of Gyration (RoG), which is a measure of mobility volume and indicates the characteristic distance traveled by an individual [37]. In [38], the Radius of Gyration is used to characterize human mobility patterns emerging from available GPS trajectories. In [39], starting from nation-wide mobile phone data, authors investigate the correlations between the Radius of Gyration and external socio-economic indicators.

In light of this, the idea of combining tools from human mobility into knowledge extraction from text is a promising research direction. In [40], RoG is used on geo-localized data from Twitter to study urban systems and human dynamics to help urban planning and policy making. However, these tools can be an appropriate choice also when a geographic or spatial reference is missing in the given domain. In this case, the definition of a geometric space depends on the task and on the available data. Recently, several works leverage word embedding spaces to get insights from text.

All of these progresses in knowledge extraction, human mobility and in Natural Language Processing, motivate our choice of investigating neural network embedding spaces with Radius of Gyration, to exploit the semantic space and its vector proper-

ties.

3.1.1 Radius of Gyration

The Radius of Gyration can be defined as the physical length that represents the radial distance in a rotating system of a body from the axis of rotation itself. Assuming that the mass of the body is concentrated in a point, the Radius of Gyration is computed such that the moment of inertia about the given axis is the same as considering the actual distribution of mass.

In the last years, this metric has been used in different fields, including the estimation of mobility volumes from geographic data [37, 39]. In this field, the Radius of Gyration of an individual can be expressed as:

$$r_g = \sqrt{\frac{1}{N}\sum_{i \in K}(\mathbf{r_i} - \mathbf{r_{gm}})^2} \tag{3.1}$$

where:

- r_g is the Radius of Gyration.
- $\mathbf{r_i}$ is a vector describing the geographic coordinates of location i.
- $\mathbf{r_{gm}}$ is the center of mass of the individual.
- K is the set of locations visited by the individual.
- N is the total number of visits or time spent in a location.

Hence, for each user, the Radius of Gyration can be seen as the standard deviation calculated among his/her positions with respect to his/her center of mass (that is the average location overall positions).

3.1.2 Evaluation of the semantic locations

Our goal here is to learn a representation for each document that is involved in the target analysis in a dynamic context. For this reason we can link a temporal information to the documents (e.g., year of release or publication) and an agent that represents the owner of these documents (e.g., the author) and the traveller in this semantic space over the time. In the following, it is described how to build these trajectories starting from a semantic space.

Starting from any representation learning algorithm it is possible to build a semantic space from the collection of documents. In the application part, the Word2Vec algorithm is used to extract the semantic coordinates of each topic, creating the semantic space. The training phase needs a sequence of sentences and produces a model which can be used for multiple purposes.

After the training phase, the model can express a numerical distance between two vectors, which represent two words (topics). Some examples of distances expressed as cosine similarity values are shown in the following:

- care, health_care = 0.302976
- volume, distribution = 0.004110

The obtained model holds a vocabulary of words. Each word is represented by an arbitrary multi-dimensional vector of several coordinates. For each active year of an agent(author), words that come from his documents are collected, if they belong to the model. Then, the geometric center (r_i, for the i-th year) of the coordinates of the selected words is calculated. Figure 3.1 shows an example of mobility of an author during 3 years. For each of the 3 years, the geometric center (in red) of all the topics (words) of that year is computed (r_i).

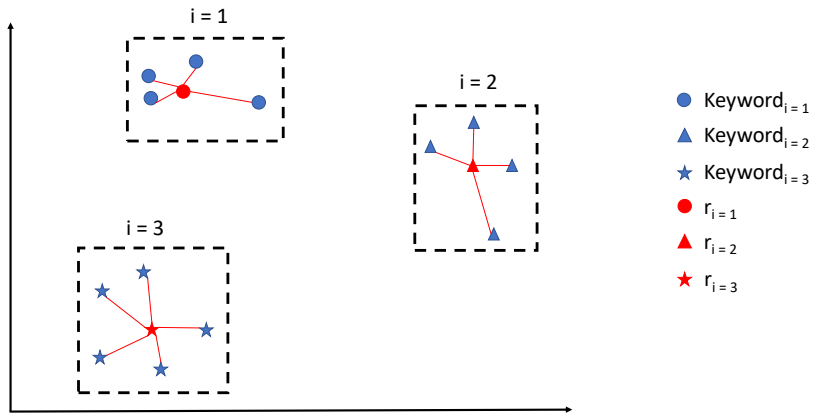


Figure 3.1: An example of mobility of an author during 3 years. For each year, the geometric center (in red) of all the topics (words) of that year is computed (r_i).

3.1.3 Estimation of Mobility

This phase aims to estimate the "mobility" of an agent in the semantic space created by the Word2Vec algorithm. The mobility volume can be estimated through the Radius of Gyration. In the application section, this metric will be used fro example to estimate semantic mobility across different research topics, measuring how the works of a researcher are distributed over different research topics.

The geometric center (r_{gm}) of all geometric centers (r_i) of an author's active years is computed (Figure 3.2).

The Radius of Gyration corresponds to the mean squared error of the distances between each r_i and r_{gm}, which is the geometric center of all geometric centers (in green).

In this way, it is possible to estimate a mobility of an agent in a semantic space with the same metrics that usually are adopted in geographical context.

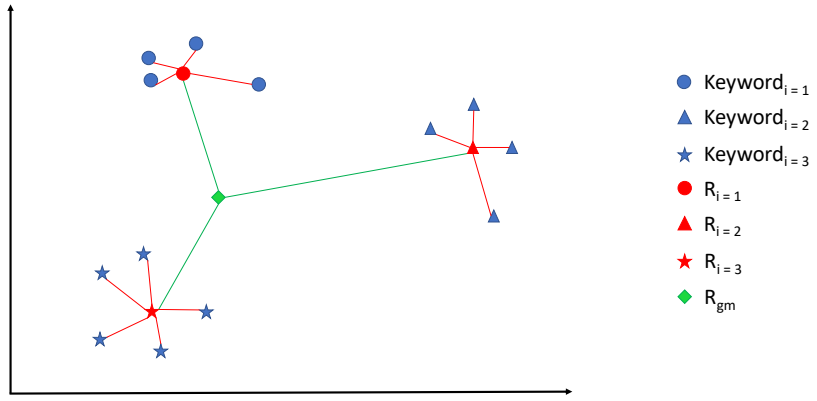


Figure 3.2: Schematic representation of data used for tracing an author, including: topics, geometric centers by year and overall center.

3.2 Semantic changes over time with word embedding proximity

Document similarity is a well-known task with several approaches in literature. However the concept of similarity is problem-dependent and the nature of this task is often unsupervised. Indeed, two documents can be similar or not depending on the elements that are took into account in the analysis. The most common task defines similarity on the basis of the common words in the two documents, with metrics derived from the Set theory (e.g., Jaccard similarity). When documents are encoded in vector representation, for example with the Bag-Of-Words and Tf-IDF, it is possible to compute the similarity with the cosine similarity between the two vectors. However the concept of similarity can be more complex: two documents can be similar because they focus on the same topic or they use synonyms that Jaccard similarity or Bag-of-words will ignore.

At the same time, neural network embedding techniques such as Word2vec and Doc2vec are able to learn synonyms keep into account the semantic of the words by analyzing window context. However, the design of these networks in terms of their hyper-parameters is often difficult and again problem-dependent.

3.2.1 Extraction of hidden information from similarity

In this section, a general methodology is presented to extract knowledge with document similarity when the goal is to understand if two documents are similar when the intention of the authors is deliberately to make difficult to understand the difference by using different words and using different lengths. An application of this methodology will be presented later in the financial domain, in particular in the analysis of American companies financial reports. Indeed, in this domain it is common that managers are encouraged to provide boilerplate information, avoid giving accurate signals of the company's status by extending the document length and obfuscating important information. This managers attitude tries to minimize (maximize) the stock price effects of negative news (positive news) about their firms. However the same approach could be used in different domains for example in fraud detection or to understand if two documents have the same author.

3.2.2 Word embedding approach

The challenging research question here is to understand which methodology could better address this concept of document similarity by comparing the traditional Bag-of-words with Jaccard similarity and neural network embedding techniques with the Cosine similarity. We took into account Word2Vec, learning a document representation by averaging word vectors and Doc2Vec in both version presented in the State of the Art chapter. The results will be presented in details in the ad-hoc section but can be resumed here by looking to a well-known case of document similarity in financial domain. In "Lazy Prices" (Cohen et al), [41] argued that a simple comparison of consecutive reports hides a lot of valuable information. It is true that while tables in financial statements are always presented with the current year's numbers accom-

panied by several previous years' corresponding numbers, the same is untrue for the text. The management being "lazy" a lot of times uses last years filings verbatim in constructing the current year's reports while making only the necessary changes so as to be within the boundaries of fiduciary responsibility. Observing these changes yields an important, and robust indication for future firm performance.

We argue that since the methods used in the "Lazy Prices" paper to measure document similarity ignore syntax or semantics, these differences can be better captured with a model that does, especially in cases where the CEO/CFO strategically obfuscate risks and corporate issues ([42]). The main observation that encouraged us to further analyze this context is related to the ability of Doc2Vec to better highlight changes among consecutive documents in this context with respect to the other methods. This result can be seen in Figure 3.3 and Figure 3.4 where it is possible to compare the similarity score obtained for 2010 in Lazy Prices for the famous case of the Baxter company financial reports and our result with neural network embedding. Indeed, the financial reports in 2010 for this company had valuable information for the investors because although the report was very similar to the one of the previous year as usual, important information about a request of more controls on their products from the Food and Drug Administration (FDA) were hidden in different section. Understanding this fact at the beginning of the year would have been important in predicting the slump in earnings following the subsequent FDA suspension. In Lazy prices with the BoW model with Jaccard similarity it is possible to see a difference (red rectangle in Figure 3.3 between the financial report in 2010 and 2009 but it is marginal and it is difficult to understand if it is noise or not. On the other hand, a document embedding model trained on the past reports is able to detect a more robust difference in the similarity as shown in Figure 3.4

Doc2vec outperforms also Word2Vec in this context because it does not need to average word vectors but learn a representation of the document while learning word vectors inside it. More details will be presented in the application section where the Bag-of-words model, Word2Vec and Doc2Vec are compared with the same data set in order to maximize investments gains when companies for the portfolio are selected

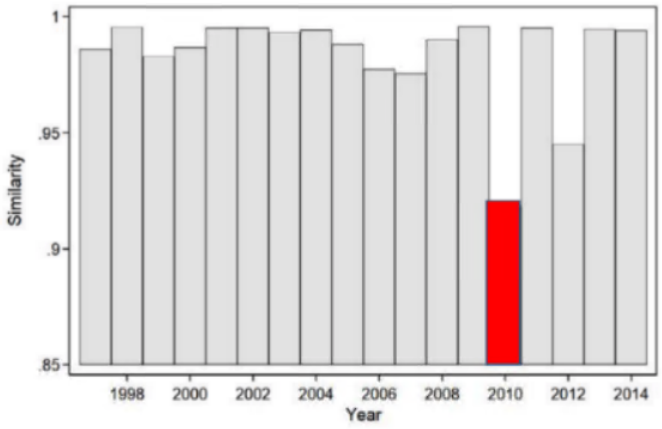


Figure 3.3: Similarity between Baxter's financial reports from year to year with Bag-of-word and Jaccard similarity

considering these similarities in their consecutive financial reports.

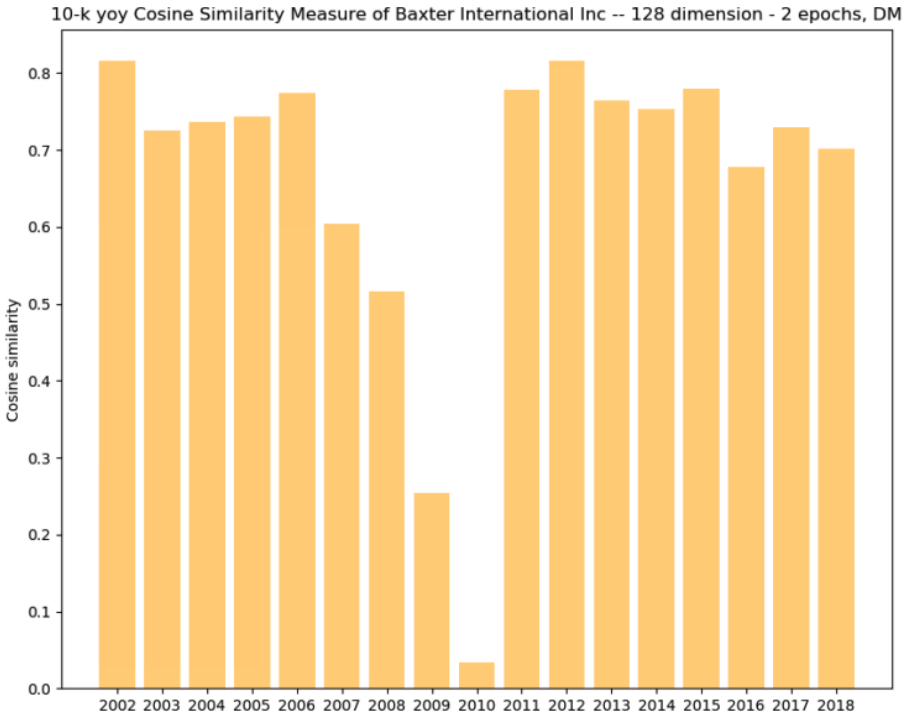


Figure 3.4: Similarity between Baxter's financial reports from year to year with the Distributed Memory of Doc2Vec and Cosine similarity

Chapter 4

The case of scholars' travelling across research topics

As a case study, the Scopus database has been queried about works of highly cited researchers of 2017. On this basis, we have conducted a dynamic analysis, for measuring the Radius of Gyration as an index of the mobility of researchers across scientific topics. Results show some latent properties of this model, which also represent new scientific contributions of this work. These properties include: (*i*) the correlation between the scientific mobility and the achievement of scientific results, measured through the H-index; (*ii*) differences in the behavior of researchers working in different countries and subjects; and (*iii*) some interesting similarities between mobility patterns in this semantic realm and those typically observed in the case of geographic mobility.

4.1 Context

In the context of Big Data mining, the ability to retrieve information, insights and values from documents, in its heterogeneous forms, is crucial. This task is challenging and require to find a trade-off between automatic knowledge discovery, performance of tools and techniques able to give a productive representation of knowledge. Several

works in this area have been already presented in Section 2. In particular, we highlighted the key-role that neural network embedding techniques can play in this field. Indeed, Word embedding algorithms provide a framework where semantic distances among terms can be evaluated. These computationally-efficient predictive models turn words or phrases into vectors of real numbers in a multi-dimensional space, where semantically related words are mapped to nearby points [43, 44]. Among those algorithms, Word2Vec [2] has reached the status of a consolidated tool.

As a separate research area, several works are conducted in the field of knowledge discovery from bibliographic databases [45, 46]. Scopus, together with Web of Science, represents one of the most authoritative bibliographic databases [47]. In addition to the abstracts of articles, it provides access to the references included in those articles, allowing users to search both forward and backward in time [48].

In [49], data mining techniques are applied to gather and analyze data from the Scopus repository. A social graph of scientific authors is created, starting from citations among the articles. Moreover, using data mining techniques, some relevant research topics are inferred for each author, from the textual analysis of the abstracts of his/her articles. The study shows the emergence of some clusters of topics, which are studied by distinct groups of authors.

In [50], the relation between interdisciplinarity and citation impact of individual publications is analyzed for four different scientific fields. Thus, the study does not analyze researchers' mobility across topics, but the impact of multidisciplinary and cross-topic researches. The article is relevant because it finds results that are somewhat similar to those highlighted in our own study. In general, interdisciplinarity is seen as a highly positive criterion for obtaining the most prestigious grants. But the authors observe that, although the combination of multiple fields has a positive effect in creating new knowledge, successful research is best achieved by focusing on related fields of knowledge.

Instead, interdisciplinary research in fields that are very different could have a higher chance to fail. On the other hand, those results may suggest that the scientific community is reluctant to cite heterodox articles that mix highly disparate bodies of knowledge, thus giving less credit to publications that are too innovative or stimulating. The conclusion of the article is that the practice of "proximal" interdisciplinarity pays off in quotations, while highly interdisciplinary researches are not rewarded with citation success.

Similarly, our research shows that although some mobility may demonstrate some curiosity in the academic world, this does not pay in terms of a scholar's impact factor. A stimulating study, which analyzes the reasons for these results, is described in [51]. The article analyzes the way in which scholars explore the scientific landscape, or choose research topics to work on. Scientific literature is seen as a partially revealed landscape, in which scholars, considered as agents in this context, continue to reveal hidden knowledge by exploring new research themes. Through the researchers' strategic behavior in the choice of topics to work on, the authors bring out, grow, support or decay the topics characterizing the evolution of scientific research. The proposed framework assumes that scholars have different goals, such as surviving for a long time in the academic community or achieving greater scientific significance. Eventually, the strategies they put in place can provide a balance between individual scientific success and the efficiency and diversity of the whole academic society. The article proposes four types of scholars who play different roles: experts lead scholars to topics with high research potentials, mavericks are the pioneers of novel topics, followers and conservatives embrace the wisdom of crowds. The authors concludes that the ratio of scholars adopting certain strategies has a significant impact on the health and progress of the scientific community, and could be a factor to be taken into consideration if we wish to promote multidisciplinary research and exchange of knowledge among researchers in different fields.

Thus, many research works dealing with big data are based on geographic positioning and mobility information, which are being studied with different analytical

Table 4.1: Related works, organized according to their topic and methodology.

Scholars' discipline	Word embedding	Mobility (RoG etc.)	Related works
		✓	[35, 36, 37, 38, 39, 40]
	✓		[29, 30]
	✓	✓	Nothing, to our knowledge.
✓			[29, 45, 46, 49, 50, 51]
✓		✓	Nothing, to our knowledge.
✓	✓		[29]
✓	✓	✓	Nothing, to our knowledge.

tools and methodologies. However, all those approaches rarely find an application outside of the geographic realm and virtually none tries to apply the concept of mobility in the semantic space of research topics, as shown in Table 4.1. On the other hand, studies on knowledge discovery from bibliographic databases are conducted as a completely separate research area. Also among these many relevant researches, to our knowledge, virtually none has taken into account any semantic mobility metrics, or even the simple inclination to change the focus on different research topics, during a scholar's academic career. In fact, this is the main aim and the main scientific contribution of this research work. Table 4.1, which organizes related works according to their topic and methodology, also highlights the novelty of our work with respect to the state of art.

4.2 Methodology

The methodological steps have been structured along the following pipeline:

- **Data Collection**: this step aims to collect the raw dataset that is likely to contain relevant patterns;
- **Data Preprocessing**: the preprocessing phase increases the quality of the data and allows one to reach better results in the following steps, removing noise and incomplete data;
- **Generation of the Semantic Space**: in this step, the Word2Vec algorithm is used to create the semantic space;
- **Evaluation of Scholars' Locations**: for each author, the scientific production is used to compute the "average topic" in the semantic space for each year; This part represent an application of the mobility estimation methodology presented in the previous chapter.
- **Estimation of the Mobility**: a dynamic analysis is conducted, measuring the Radius of Gyration as an index of the mobility volume of each researcher across scientific topics.

4.2.1 Data Collection

In order to investigate the correlation between the mobility in the topic space and the success in scientific research, 2998 authors have been selected from the 3500 "Highly Cited Researchers of 2017" list provided by Clarivate Analytics[1]. This annual list collects leading researchers in the sciences and social sciences from around the world. Starting from this set of researchers, a data collection phase is performed from Scopus using the Elsevier developers API. For each author, the following information is collected:

- Details of each article, such as abstract, keywords, year of publishing;

[1] https://clarivate.com/

- Temporal range of activity;
- H-index metric;
- Nationality and affiliation.

This primary data collection phase has two main purposes:

- Collect textual data to train the Word2Vec algorithm, in order to obtain an accurate semantic space of topics;
- Collect information about researchers, which will be used in the Data Mining phase to study the hidden relation between an author's mobility across topics and the success in scientific research.

After performing the collection phase, the data is further processed with the aim of collecting comparable information. In fact, in order to have a fair comparison among researchers success, it is necessary to consider a common period of work activity. In light of this, only authors who have published for at least six years in the period between 2005 and 2015 are selected. For each author, the H-index metric is recalculated, considering only the scientific contribution in the considered period.

The resulting number of selected works is 606605, with 113 millions of *non-unique* words from abstracts and keywords of the collected papers.

4.2.2 Data Preprocessing

This phase aims to clean the data previously collected in order to increase the accuracy of the Word2Vec model. With this aim, standard Natural Language Processing (NLP) techniques are applied to the textual data. In particular, special characters and capitalization are removed. A stop words filter is applied to remove words which do not convey significant meaning (i.e., articles, prepositions). In the end, a stemming algorithm is used to reduce inflected words to their word stem, base, or root form.

Generally, in an NLP preprocessing phase, it is also important to deal with N-grams, which are words obtained by a composition of *N* words, i.e. “New York” or “New York Times”. For example, in the sentence “I’ve bought a new car”, the word “new” has its meaning (adjective in this case). Conversely, in the sentence “I’ve been in New York” we should consider the word “New_York” as a whole. The aim is to preserve the words “New” and “York” when they appear separately, but also to recognize the name “New_York” as a single entity (in the model, all letters are converted into their lowercase form).

The dataset has been processed using Gensim[2], a Python library to realize unsupervised semantic modeling from plain text. In particular, Gensim has been used to detect co-occurring multiword expressions from a stream of sentences automatically. In light of this, textual data is preprocessed in order to consider words composed of both bigrams (2 words) and trigrams (3 words), in addition to unigrams (1 word).

4.2.3 Semantic space generation

In this phase, the Word2Vec algorithm is used to extract the semantic coordinates of each topic, creating the semantic space. Word2Vec has been chosen among other word embedding algorithms because of its accuracy and low computational cost. We have used the Gensim library, which implements the Word2Vec algorithm.

In the Word2Vec model, the coordinates of a topic represent the weights of the hidden layer in the neural network [52] (`size` parameter).

The training phase needs a sequence of sentences and produces a model which can be used for multiple purposes. The training set used by the Word2Vec learning phase is the output of the data preprocessing phase, described in section 4.2.2.

The training parameters[3] of the Word2Vec model are set according to [53] and [54].

[2] `https://radimrehurek.com/gensim/`

[3] `https://radimrehurek.com/gensim/models/word2vec.html`

The most important parameters of the Word2Vec implementation in Gensim library, that have been changed from the default value, are:

- `size` = 600;
- `min count` = 300;
- `window size` = 132.

We have experimented systematically with different values for *min count* and *size* before selecting the most suitable for our model. To control this process, we have used a Python implementation of t-SNE (sklearn.manifold.TSNE[4]) to visualize the Word2Vec model topics data in a 2D plot. The *window size* parameter is set as the average number of words (132) contained in all the abstracts and keywords of the considered papers. With this choice, words coming from the same paper (words of abstracts and keywords) tend to be related to each other.

The model is trained with a corpus of about 113 millions *non-unique* words (coming from abstracts and keywords of all the preprocessed papers) and with the capability to recognize bigrams and trigrams. After the creation of the model, the vocabulary contains 13187 unique words. The obtained model holds a vocabulary of words. Each word is represented by a multi-dimensional vector of 600 coordinates.

4.3 Experimental results

After having generated a semantic space where scholars move in, we have further analyzed it from different points of view with the aim of extracting latent knowledge from scholars' research activities. In particular, we aim to answer the following intriguing questions:

[4] `http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`

- Is it possible to predict a researcher's success just by observing his/her attitude to explore new research topics?
- Is there any pattern in the scholars' mobility across topics over time?

In this section, we present the different analyses that have been performed and the related results to answer these questions. The first one can be related to the concept of mobility volume (Radius of Gyration), computed in the semantic space for each researcher. Moreover, some differences in the behavior of researchers working in different countries are investigated. The second one has been investigated looking for the probability density function (PDF) that could better approximate the scholars' mobility in the semantic space over the analyzed time.

The analyzed dataset is composed of about 3000 researchers. For each author, information about articles, country, and H-index is collected, considering only the selected reference period (4.2.1).

4.3.1 Correlation Analysis

In general, the linear correlation between two variables X and Y is measured by the *Pearson correlation coefficient*, $Corr(X,Y)$, which assumes a value between $+1$ and -1, where $+1$ is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation.

$$Corr(X,Y) = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2}} \tag{4.1}$$

The considered variables for computing the linear correlations are:

- X : The Radius of Gyration of each researcher;
- Y_1 : The H-index of each researcher;
- Y_2 : The H-index of each researcher, normalized for subject.

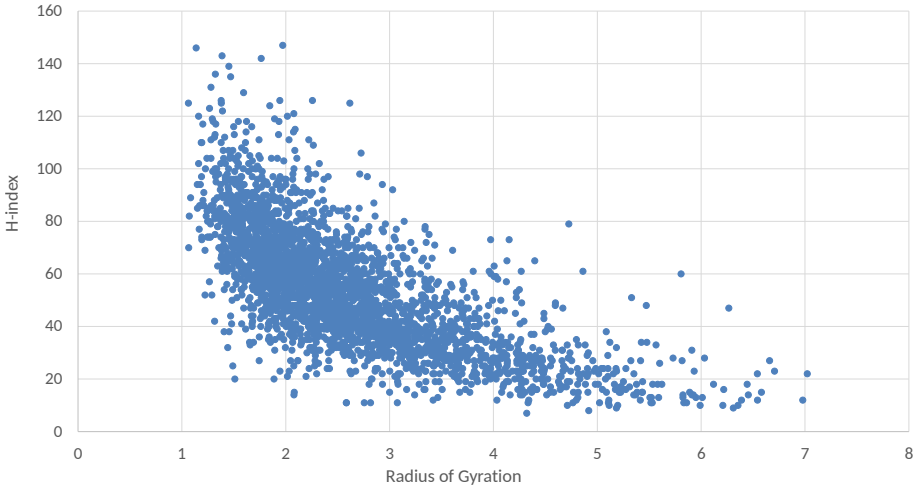


Figure 4.1: Scatter plot of authors, represented by their Radius of Gyration (x-axis) and their H-index (y-axis).

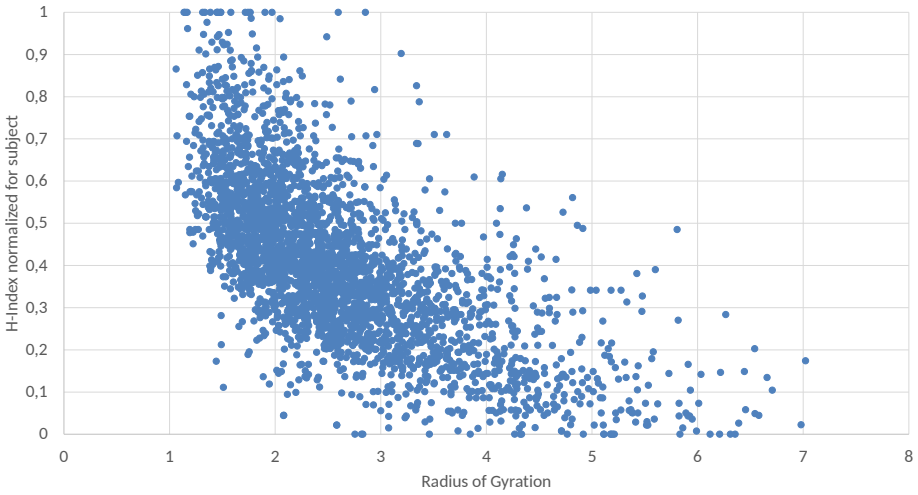


Figure 4.2: Scatter plot of authors, represented by their Radius of Gyration (x-axis) and their H-index (y-axis), normalized for subject.

Since the researchers' H-index ranges are related to their research subjects, all H-index values are grouped by subject and normalized (Y_2) to bring all values into the

range [0,1]. In this way, it is possible to make a fair comparison among researchers working in different fields.

Table 4.2 shows the correlation values $Corr(X,Y_1)$ and $Corr(X,Y_2)$. It is worth noting that correlations assume significant negative values.

Correlation	Value
$Corr(X,Y_1)$	-0,68418
$Corr(X,Y_2)$	-0,65167

Table 4.2: Linear correlation values $Corr(X,Y_1)$ and $Corr(X,Y_2)$.

These negative correlations can also be observed in Figure 4.1, which shows the distribution of the authors according to their Radius of Gyration, along the x-axis, and their H-index, along the y-axis. Since the statistical distribution of citations may depend on the subject area, in Figure 4.2, the H-index of each scholar is normalized in the observed range of his/her subject area.

Indeed, results show that rarely a single researcher can have a significant impact on several different research topics. Instead, the most successful researchers tend to focus on a quite narrow research area.

4.3.2 Significance Analysis

Statistical hypothesis testing has been used to determine if results are statistically significant. In particular, we evaluated the following significance:

- Correlation, considering the results shown in Table 4.2.
- Two Regression Models, considering the H-index as the dependent variable for the first regression, and the H-Index normalized for the subject for the second one. Both cases use the Radius of Gyration as the independent variable.

For the correlation analysis, the *p-value* represents the probability that you would have found the current result if the correlation coefficient were in fact zero (null

Statistic Regression		
	Y = H-Index	Y=H-Index normalized for subject
R	0.688247525	0.627967253
R^2	0.473684656	0.394342871
$R^2(adj)$	0.473508395	0.39413997
$StandardError$	0.715257263	0.767136911

Table 4.3: Statistic Regression Results.

hypothesis). If this probability is lower than the conventional 5% ($P < 0.05$), the correlation coefficient is called statistically significant. Conventionally the 5%, 1% and 0.1% ($P < 0.05$, 0.01 and 0.001) levels have been used. Most authors refer to statistically significant as $P < 0.05$ and statistically highly significant as $P < 0.001$ (less than one in a thousand chance of being wrong). In our case, we have found $P < 0.001$, and we can therefore state that the correlation is statistically highly significant.

We have also performed a significance analysis of the regression model, using the F-Test. The F-test indicates whether a linear regression model provides a better fit to the data than a model that contains no independent variables. In this case, the *p-value* indicates if there is a significant relationship described by the model, and the R^2 (coefficient of determination) measures the degree to which the data is explained by the model. In particular, R^2 is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). The Adjusted Coefficient of Determination (R^2adj) is an adjustment for the Coefficient of Determination that takes into account the number of variables in a data set. It also penalizes you for points that don't fit the model. As in previous cases, we have considered two different scenarios, choosing the H-index as the dependent variable for the first regression, and the H-Index normalized for the subject for the second one. Results are shown in Table 4.3.

For both cases, we have found $P < 0.001$, and we can therefore state that the regression models are statistically highly significant. The results also show that the Radius of Gyration explains the 47% of the H-Index variance and the 39% of the

H-Index normalized for the subject.

Even if there is a strong connection between the two variables, determination does not prove causality, and we can not deduce a cause-and-effect relationship solely based on an observed association or correlation.

4.3.3 Subject Analysis

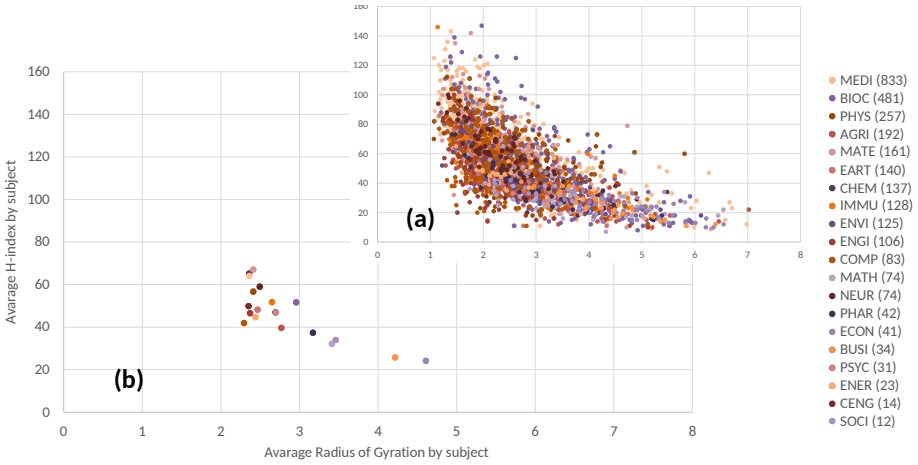


Figure 4.3: Scatter plot of single scholars (a) and their average values when grouped by research subject (b). Radius of Gyration is on the x-axis. H-index is on the y-axis. Labels are associated with the corresponding number of instances.

Since this analysis regards scholars working in very different subject areas, in Figure 4.3 data are distinguished by the subject areas associated with scholars on Scopus. Subject codes are defined in the Scopus API[5]. In particular, in graph (a) the analyzed scholars are colored by subject area. It can be observed that some groups of scholars of certain subject areas are represented by nearby points and, also selecting each subject area, the two considered variables are anti-correlated. In fact, considering the most represented subjects, the following important values of anti-correlation can be observed: MATH, -0.636; MEDI, -0.677; ENGI, -0.531. To better highlight

[5] https://dev.elsevier.com/documentation/ScopusSearchAPI.wadl

some possible differences among subjects, graph (b) also shows the average values of the Radius of Gyration and the H-index, grouping scholars by subjects. According to the results, researchers of different subjects behave differently, and also these average values are scattered approximately along a line with a negative slope. The subjects with the highest values of the Radius of Gyration are Economics and Business. In [55], it is shown that the differences among the average H-index calculated for each disciplinary field (or Scopus subject) can be largely accounted on both the career length and the average number of authors per article in the various fields. Figure 4.3 shows that in correspondence with the lower average H-index in the cases of Economics and Business, the mobility across research topics measured by the Radius of Gyration is higher.

4.3.4 Scholars' mobility pattern

Modeling the authors' research activities as a travel in a semantic space enables investigating another kind of result, related to the emergent patterns in the authors' trips in this space over the time. In particular we looked for a mobility pattern considering the displacements between visited locations in consecutive years (*flight length*) by each author. Understanding these patterns may be of interest for different stakeholders because it is possible to estimate the probability that a change of a topic occurs among the top scholars analyzed. For example it could be considered in the evaluation criteria that occur in many different national academic systems or even to evaluate research projects or their management. Before discussing the specific findings, in this section we present the methodology that we have used and the kind of results obtained. This kind of analysis can be seen as borrowed from the Human Mobility field, where it emerges that the length of human travels can be described by a Power Law distribution in general [37, 56] and by a Lognormal distribution if analyzed with a transport modality decomposition (walk, car, bike, train, airplane) [57, 38]. In our analysis the scholars travel among the topics, modeled as the estimated locations. We have studied the mobility for each scholar and for each year, as discussed in Section **??**. The distance between two consecutive locations is measured with the Euclidean distance. The Probability Density Function (PDF) has been computed over the fre-

quency of displacements, in order to estimate which distribution can better describe the authors' mobility across the topics. We have considered the following distributions: Power law, Truncated Power Law, Exponential and Lognormal. The following methodology has been adopted to select which distribution best fits the scholar data:

1. For each distribution, the specific parameters have been selected using the Maximum Likelihood Estimator (MLE) between the candidate distribution and the empirical data [58, 59].

2. In order to quantify the goodness of each fit, the Likelihood Ratio test [60] has been performed among each couple of distributions. As reported in literature, this comparative test is preferable to an individual one (e.g Kolmogorov-Smirnov test) for its robustness to noise in the data [61]. The result of this test is the Log-Likelihood ratio **R** between the two candidate distributions and a p-value. The result is considered acceptable if the $p\check{}value < 0.1$.

3. Once the best distribution has been selected, we have performed an individual test for the resulting distribution with the Kolmogorov-Smirnov (or KS statistic) and bootstrapping [62] anyway. The result is considered acceptable only in the case of $p\check{}value < 0.1$.

Figure 4.4 presents the PDFs describing the empirical data from scholars trips and the candidate distributions, whose parameters have been fitted with the MLE. It is possible to note that the black curve related to the Lognormal distribution is the one that best approximates our data. This evidence is also shown by the Log-Likelihood ratio test, whose results are presented in table 4.4 for each comparison. The Lognormal distribution parameters estimated with the MLE are $\mu = 2.967$ and $\sigma = 0.449$. The goodness of the Lognormal fit has been also evaluated for the sake of completeness with individual Kolmogorov-Smirnov tests. As presented in table 4.5, the only distribution fit that is acceptable is the Lognormal one with a $KS - distance = 0.03$. This result is interesting also considering that several patterns related to the human mobility follow the same distribution [38]. Once detected that scholars' travels around research topics are well approximated by a Lognormal, we try to explain what are the

implications of this result. First of all, this distribution results to be concordant with the previous results concerning the Radius of Gyration: top scholars tend to obtain better academical results by focusing on specific research areas. Lognormal distribution confirms this result and an interpretation can be resumed in this way: top scholars obtain better academical results by focusing on a specific research area but moving enough around all of the sub topics. Most of the top scholars perform medium flight length travels each year. While it is less probable that they make important shifts over the space. The same condition for definition of Lognormal occur with very small shift over the space.

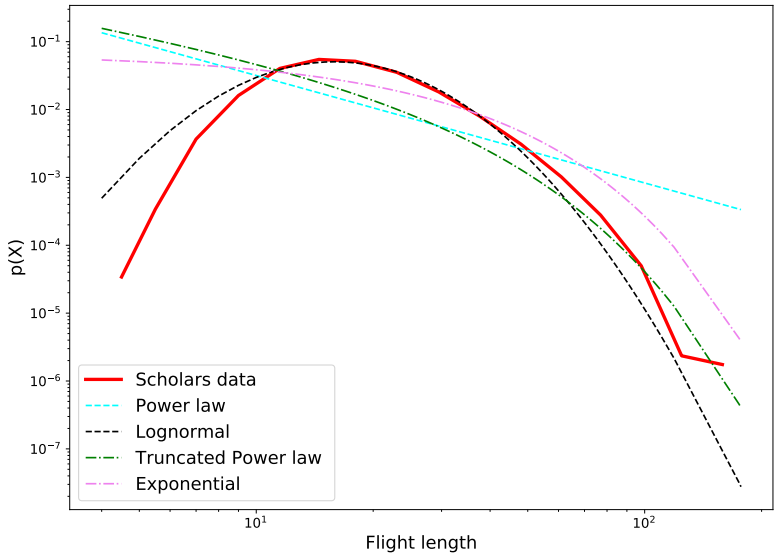


Figure 4.4: Probability Density Function (PDF) that describes the distribution of displacements between the topic locations of consecutive years (*flight length*) for each author and comparison with other distributions.

First candidate	Second candidate	R	P-value
Power law	Lognormal	**222.65**	0.0
Lognormal	Truncated Power law	**206.68**	0.0
Lognormal	Exponential	**103.61**	0.0

Table 4.4: .
Loglikelihood Ratio test results When $R > 0$ the first candidate fits better than the second; vice-versa in the other case.

Candidate distribution	KS-distance
Power law	0.40
Truncated power law	0.24
Exponential	0.52
Lognormal	0.03

Table 4.5: Results related to the Kolmogorov-Smirnov test applied to each distribution fit in the Cumulative Density Function form.

4.4 Final remarks

The main objective of this work was to bring some analytical tools, traditionally used with geographic data, in the semantic realm. We have applied a word embedding algorithm to a large corpus of abstracts of scholarly articles. In particular, the resulting vector space is obtained through the Word2Vec algorithm, in which a neural network is used to learn latent relationships among terms. In this kind of space, the notion of proximity reflects an association between terms or topics.

The case study that we have analyzed deals with articles written by highly cited authors, of many different countries and research fields, systematically collected from the Scopus database. On this basis, we have conducted a dynamic analysis for measuring the Radius of Gyration as an index of the mobility of researchers across scientific topics. The scientific mobility in the topic space of the case study has been com-

pared with the reach of scientific results, measured through the H-index. Moreover, some interesting differences have emerged in the behavior of researchers working in different countries. Finally, some interesting similarities have emerged, between mobility patterns in this semantic realm and those typically observed in the case of geographic mobility.

Results show that rarely a single researcher can have a significant impact on several different research topics. Instead, the most successful researchers tend to focus on a quite narrow research area. These results are compatible with the framework proposed in [51], in which scholars were simulated by agents with different strategies and objectives, depending on the context in which they operate and the academic age. Furthermore, similarly to the conclusions proposed in [50], the results of this research show the risk of failure to which multidisciplinary researchers are exposed. Thus, their curiosity and skills must be adequately supported, in order to flourish in a knowledge society, being characterized by multiple contradictory objectives, but having much to gain from multidisciplinary research and achievements.

We believe that the main scientific contribution of this research work is methodological, demonstrating that the semantic spaces obtained through word embedding techniques are amenable to mobility measures, which are typically applied to the geographic realm. In fact, these initial results demonstrate that the proposed approach is effective for the domain of scholar data, with promising developments which may find application also in other domains.

Chapter 5

Financial reports analysis with Neural network embedding for portfolio selection

In recent years, there has been an increased interest from both academics and practitioners in automatically analyzing the textual part of companies' financial reports to extract meaning rich in information for future outcomes. In particular, tracking textual changes among companies' reports can have a large and significant impact on stock prices. This impact happens with a lag implying that investors only gradually realize the implications of the news hinted by document changes. However, the length of these documents as well as their complexity in terms of structure and language have been increasing dramatically making this process more and more difficult to perform.

In this work, we analyzed how to face this complexity by learning arbitrary dimensional vector representations for US corporate filings (10-Ks) from 1998 to 2018, exploiting and comparing different neural network embedding techniques which take into account words' semantics through vectors proximity. We also compared their ability to capture changes associated with future risk-adjusted abnormal returns with

other more commonly used approaches in literature. Finally, we propose a novel investment strategy named Semantic Similarity Portfolio (SSP) that exploits these neural network embeddings. We show that firms that do not change their 10-Ks in a semantically important way from the previous year tend to have large and statistically significant future risk-adjusted abnormal returns. We, also document an amplifying effect when we incorporate a momentum-related criterion, where the companies selected must also have had positive previous year returns. Specifically, a portfolio that buys “non-changers" based on this strategy earns up to 10% in yearly risk-adjusted abnormal returns (alpha).

This work has two goals. First, we show that the neural network embedding techniques represent an interesting approach that is able to address the increasing complexities of annual reports’ textual analysis. In light of this, we construct a portfolio, named Semantic Similarity Portfolio (SSP), that exploits the Distributed Memory Model of Paragraph Vectors (PV-DM) mode of Doc2Vec which we found to be the best performing technique for this task. The neural network embedding approajhch produced a superior result compared to the popular alternative Bag-of-Words (BoW) model [63] in capturing changes in consecutive 10-Ks found significant to future abnormal portfolio returns. The second goal is to show that incorporating a momentum-related criterion, based on a "non-struggling" companies attribute computed on prior companies’ returns, can have a significant amplifying effect on excess risk-adjusted returns. It can be argued that this criterion can signal the nature of these changes since a struggling company would keep it’s 10-K semantically unchanged if it’s management believed that the challenges they are currently facing will persist in the upcoming year as well. In other words, in this portfolio setting, we also avoid companies with persistent risks and difficulties that are documented in the 10-K-s but are not being removed by the CEO/CFO leaving the 10-Ks semantically unchanged.

For more details, this work has been published with Professor Panos Pardalos and George Adosoglou on Expert Systems with applications Jorunal [64]. Moreover, it represents the result of my research visiting period to the University of Florida.

5.1 Context

Published in 1998 by the Securities and Exchange Commission (SEC), the Plain English Handbook was the first publication providing guidelines to help public companies create clear SEC disclosure documents. This publication and the Sarbanes–Oxley Act of 2002, which was constructed to supervise the financial reporting, have made corporate filings an increasingly reliable source of information.

Cohen et al [41] showed this by computing quintiles from the distribution of the similarity scores from all companies and then constructing long-short equally and capitalization weighted portfolios. Specifically, they found that going long the "non-changers" and short the "changers" yields statistically significant 5-factor alphas proving that breaks from previous standardized reporting can have significant implications for firms' future stock returns.

It is normal, however, for managers to be incentivized to minimize (maximize) the effect on their companies' stock prices from negative news (positive) news about their firms respectively.([65]). Previous works [66] showed that the managers provide boilerplate information and avoid giving accurate signals of the company's status by extending the document length. However, the SEC prohibits any misleading statement or omission under Rule 10b-5 and demands a company's CEO and CFO to certify the accuracy of the 10-K. This means that even though valuable information about the company and the industry does exist in the 10-Ks, the management has incentives to hide it.

The novelty in our approach is to represent each company, its activities and current affairs as a vector by applying neural network embedding techniques to the financial annual reports of these companies. In this way, we are able to capture changes associated with future risk-adjusted abnormal returns by taking into account semantics and temporal dynamics. We also demonstrate that incorporating a momentum-related component into our portfolio selection method provides significant synergies further

adding to the originality of our work.

Machine learning applications on text have almost four decades of history. However, only in the last decades a set of machine learning techniques known as neural networks (NNs) have continued to advance and start to prove highly effective for a great number of natural language processing (NLP) tasks.

Financial news, in particular, have been extensively exploited to make predictions regarding the markets. While, more recently, social media and corporate disclosures have also been utilized in various applications.

[67], for example, produced a multi-layer algorithm testing three machine learning models, namely: SVM, k-nearest neighbors (*k*-NN) and Naïve Bayes, that exploit semantics and sentiment of news-headlines for a FOREX market prediction task. [68] proposed a novel fine-grained approach that captures explicit and implicit topic-dependent sentiment in company-specific news text. [69] trained a Naive Bayes classifier with daily news articles to predict the direction of the BIST100 Index for the day following.

Classification techniques such as Naïve Bayes and SVM have also been exploited by [70] on news text to predict the volatility of financial assets. Market volatility related to news and exploited with neural networks (NNs) has also been studied extensively by [71] [72], on the other hand, proposed an ontology based framework to mine dependence relationships between financial instruments and news. Finally, [73] presents a semantic search engine for financial news using Semantic Web technologies customized on the Spanish stock market.

Neural networks (NNs) have, also, been applied to financial news by [74], and for sentiment analysis tasks and predictors of volatility by [75]. A version of Kohonen's self-organizing map, called spiral spherical neural network, has been applied by [76] to investigate the European Union banking sector and proving interesting insights about their reciprocal action and integration.

Word Embeddings have been used by [77] in leveraging financial news to predict stock prices. Neural networks have also been used to improve the performance of sentiment analysis for StockTwits by [78]. Finally, [79] apply Doc2Vec ([80]) to detect bank distress by mining news and financial data. News analytics for buy and sell decisions have also been studied extensively in [81] where models such as k-nearest neighbour, feed-forward NNs, SVM and Naive Bayesian classifiers are compared in classification tasks and sentiment anaysis. Various other computational approaches for asset trading have also been compared with sentiment analysis and news text analytics by [82].

Furthermore, motivated by the works of [83], who showed in their paper using the Bag-of-Words (BoW) model that firms that undergo significant economic changes modify the Management Discussion and Analysis (MD&A) section of the 10-K reports in a much greater way than the ones that do not. [84] uses Naıve Bayesian machine learning algorithm to associate MD&A tone with future firm performance. [85] apply the Latent Dirichlet allocation (LDA) model to a large panel of CEO diary data to estimate behavioral types and predict firm performance.

More recently, neural networks (NNs) have been also used in analyzing corporate filings. [86] use deep learning on the item 1A (Risk Factors) of various banks' 10-Ks for the classification task of predicting bank failures. Deep learning models have also been used on disclosures to predict corporate bankruptcies ([87]). Furthermore, [88] uses the Word2Vec model to learn the continuous-vector word representations in order to discover new finance keywords and update a financial dictionary.

5.2 Methodology

We tested two different approaches: Word embedding with Word2Vec algorithm by averaging these vectors to embed the entire document, and Doc2Vec algorithm that is able to directly learn document embeddings following different approaches than

averaging word vectors. Both methodologies are based on shallow neural networks with linear activation function and unsupervised learning approach that can preserve words' order and semantics. Then, the similarity between documents can be measured using the cosine similarity measure. Namely, compared to "Lazy Prices" that focuses more on exploiting the unattended disclosed information like adding or deleting sentences in the document, our model focuses more on the changes in the topic covered and writing style in the 10-Ks by representing arbitrarily the entire documents with vectors in a fixed-dimensional semantic space.

The proposed methodology can be resumed in five main steps:

- **Data collection**: We collected all the available SEC 10-K filings for the years from 1998 to 2018. For the firms where the 10-Ks were available we also collected all the monthly returns and market capitalizations.
- **Data selection**: In order to avoid bias in our dataset and to avoid extreme returns as outliers, we filtered our collection on the basis of market value and annual return
- **Text pre-processing**: Every SEC filing has been processed in order to clean it from tables, urls, HTML tags. Finally, we applied english stopwords removal and Stemming to get the root form for each word and reduce the globally size of the dictionary.
- **Models training**: Both Word2Vec and Doc2Vec have been evaluated separately to get SEC filings embeddings
- **Portfolio construction and evaluation**: We build different weighted and equally weighted portfolios using cosine similarity among the documents to compare the different models. We also evaluated the impact of combining cosine similarity with Momentum strategies

5.2.1 Data Collection

We collected all 10-K and 10-K related SEC filings from the Loughran-McDonald dataset for years from 1999 to 2018 ([89]). We selected these year range as 2018

marks the end of the second decade after the SEC published the Plain English Handbook in 1998. For these firms we collected also monthly stock data from the Center for Research in Security Prices (CRSP) using the WRDS linking tables since the SEC identifies companies only through the CIK codes. With this data we compute monthly returns and market capitalizations for all the firms. We then computed annual returns (including dividends) for the fiscal year starting on April 1st which is when the majority of US 10-Ks have already been filed.

5.2.2 Data Selection

For each year we selected data on the basis of two policies: (1) We kept companies where the market capitalization is above than 300 million dollars. This is necessary since otherwise our results would be largely dominated by micro-caps, given that these companies encompass more than half of the publicly traded stocks while also tend to have more extreme returns (see discussion in [90]); (2) companies whose annual return value crosses the 1000% annual return threshold have been excluded from the analysis in order to avoid outliers created by small scalars. After the screening, we are left with 45,516 firm-year observations. Our resulting dataset is very similar to the CRSP stock universe both in value-weighted and equally-weighted returns which verifies that no bias of any kind has been introduced. This will be further discussed in the results section (see figure 5.1).

5.2.3 Models training

We use the corpus of all the firms' 10-Ks to train both the PV-DM and PV-DBOW Doc2Vec model with various vector dimensions and epochs. In all experiments, we use concatenation as the method to combine the vectors. The vector size, number of epochs and other hyper-parameters were selected based on the suggestions of [91]. We end up selecting the PV-DM Doc2Vec model trained with 256 dimensions and 10 epochs as the best model. For these hyperparameters we also train a Word2Vec model and develop word embeddings for all the words in all 10-Ks. We take the average of these words in each filing and use it as the representation of the document.

5.2.4 Portfolio construction

In an attempt to measure the semantic differences between two consecutive financial reports of a company we experimented with various metrics. After we represented all companies' 10-Ks with vectors, we tried to measure consecutive changes by considering the cosine similarity, the euclidean distance, the Radius of Gyration and finally the Jaccard similarity. In the light of results, we chose at the end to only use the cosine similarity as it proved to be the most effective one in capturing semantic changes with neural network embeddings. Our hypothesis is that because of the embedding vectors' nature, their orientation is much more stable and reliable than their magnitude which suffers from the random initialization of the weights of the neural networks.

We then compute for each of the companies the cosine similarity measures between their year-on-year 10-K fillings' embeddings generated by the three neural network embedding models discussed: PV-DM, PV-DBOW and the Word2Vec-based model. For each of these three cases we built a long-only portfolio consisting of stocks whose cosine similarity measure was higher than 0.95. In all cases, stocks are held for a year and the re-balancing occurs annually as well.

After computing the corresponding calendar time portfolios, we find that the PV-DM version of the Doc2Vec model is the best model out of the neural network embedding models we tested in capturing semantic changes in 10-Ks associated with future risk-adjusted abnormal returns. We term this strategy "Semantic Similarity" and report its performance against a respective strategy that uses instead the bag-of-words model to represent these documents.

Furthermore, driven by an effort to reduce selecting companies whose 10-Ks remain semantically unchanged but the companies themselves are facing persisting challenges, we incorporate a momentum-related criterion where the companies selected must also have had positive previous year returns ($Ret(-12,0) > 0$). This criterion attempts to exclude struggling companies whose CEO/CFO have reported the persisting challenges in the previous year's 10-K and have not removed them in

the current 10-K leaving these reports semantically unchanged. It could be argued that this momentum amplified strategy, termed as "Non-struggling" attempts to select well performing stable companies that face no great risks. We, finally, report its performance against the same strategy using the bag-of-words model, while we also compute returns for the raw-momentum strategy (buying stocks with positive previous year stock returns and holding them for one year) and find that there are no statistically significant abnormal returns associated with it.

5.2.5 Financial models for the evaluation

For the performance evaluation we use multi-factor alphas since the large returns found in this study might have resulted from large exposures to systematic risk factors. We investigate this hypothesis by adding to the Capital Asset Pricing Model (CAPM) the two most influential systematic risk factors: the size based factor small-minus-big (SMB) and the high-minus-low book-to-market factor (HML).([92]). Furthermore, for a 5-factor analysis we also include the up-minus-down momentum factor (UMD), as well as the Pástor and Stambaugh's traded liquidity factor (PS_VWF)
.

The CAPM, Fama-French and 5-Factor alphas along with the corresponding betas are empirically estimated via a linear regression as:

Capital Asset Pricing Model (CAPM):

$$R_t - r_t^f = \alpha + \beta_{MKT}\left(R_t^M - r_t^f\right) + \varepsilon_t \tag{5.1}$$

3-Factor Fama and French Model:

$$\begin{aligned} R_t - r_t^f = \alpha + \beta_{MKT}\left(R_t^M - r_t^f\right) + \\ \beta_{HML}HML_t + \beta_{SMB}SMB_t + \varepsilon_t \end{aligned} \tag{5.2}$$

5-Factor Model:

$$\begin{aligned} R_t - r_t^f = \alpha + \beta_{MKT}\left(R_t^M - r_t^f\right) + \beta_{HML}HML_t + \\ \beta_{SMB}SMB_t + \beta_{UMD}UMD_t + \beta_{PS\_VWF}PS\_VWF_t + \varepsilon_t \end{aligned} \tag{5.3}$$

where $R_t - r_t^f$ is the excess return from each strategy, $R_t^M - r_t^f$ is the market risk premium, r_t^f is the risk free rate based on the one-month Treasury bill rate, HML_t is the difference betweenthe high book-to-market value companies' returns minus low book-to-market value companies' returns, SMB_t is the difference between the small capitalization and large capitalization portfolios' returns and UMD_t is the momentum factor , i.e. the returns or the winners minus losers portfolio based on the past 11 months and PS_VWF_t the Pástor-Stambaugh liquidity traded factor constructed from the returns of the top decile liquidity beta portfolio minus the returns of the bottom-decile liquidity beta portfolio.

The parameter α is the measure of the abnormal risk-adjusted return that captures the excess return above what is expected based just on the risk of the portfolio. The five risk factors' time series as well as the risk-free rates are gathered from Whartons Research Data Services (WRDS), Fama-French Portfolios and Factors dataset.

5.3 Experimental results

In this section we first report the performance of various neural network embedding techniques in capturing future abnormal returns associated with 10-Ks consecutive changes. We, then report, the performance of the "Semantic Similarity" and the "Non-changers" portfolios, termed SSP and NSP respectively and compare it to the corresponding portfolios built using the BoW model instead. We term this portfolios the "BoW" portfolio and the "BoW-Mom" portfolio. We also compare our performance results with the results derived from the "Lazy Prices" analysis which uses the BoW model to capture changes in consecutive 10-Ks.

In all portfolios we hold stocks for 12 months (from April 1st to March 30th) and re-balance every 12 months on April 1st. Note that for the value-weighted portfolio returns each stock in the portfolio is weighted by its lagged market capitalization.

We, also, display the "Lazy Prices" respective results for a direct comparison.

Finally, in Figure 5.1 we display the cumulative returns of the "Semantic-similarity" and "Non-struggling" strategies over the two decades. We display two figures, one with equal weighted and one with value weighted cumulative returns. The returns are compared with the buy-and-hold cumulative return of the S&P 500, the available stock universe, the CRSP stock universe and for a more direct comparison, with the corresponding BoW-based with and without momentum strategy returns. Note that all of the reported returns include dividends.

5.3.1 Neural Network Embeddings Performance Evaluation

Table 5.1 presents the performance of the PV-DM and PV-DBOW versions of Doc2Vec as well as the document embedding with Word2Vec model in terms of capturing changes associated with these future risk-adjusted abnormal returns. The table includes the equal-weighted and value-weighted annual portfolio abnormal returns as well as the statistical significance. These are computed by regressing in each case the twenty years of compounded returns on the market, the SMB and HML factors as well as the UMD and PS_VWF factors. The average number of companies selected each year is 60 for the PV-DM model, 240 for the PV-DBOW model and 850 for Word2Vec.

We see that the best performance lies with the PV-DM model. Specifically, the long-only portfolio using PV-DM model earns a large and significant abnormal return of 11% per year (t= 2.75). This proves the superiority of the PV-DM model. Furthermore, our results with the PV-DM model are mostly unaffected from controlling for the three Fama-French factors (market, size, and value). This suggests that the returns we see between the portfolio is not driven by systematic loadings on the most commonly used risk factors. Furthermore, controlling for two additional factors: momentum and liquidity, the equally-weighted portfolio earns significant abnormal return of

7.45% per year but the value-weighted portfolio return is statistically insignificant.

Table 5.1: **Portfolio Returns Exploiting Neural Network Embeddings:** This Table reports the annual portfolio excess return, 3-Factor alphas, and 5-factor alphas (market, size, value, momentum, and liquidity) for the three long-only portfolios constructed based on the three similarity measures: Doc2Vec's two versions (PV-DM and PV-DBOW), Word2Vec average. All portfolios select companies whose cosine similarity of the vector representations is higher than 0.95. Returns are annualized and multiplied by 100. The left part of the table presents value-weighted portfolio returns and the right part presents equal-weighted portfolio returns. The t-statistics are shown below the estimates, while the statistical significance is indicated by ***, **, and * for the 1%, 5%, and 10% levels, respectively.

	Value-Weighted			Equally-weighted		
Portfolio	CAPM Alpha	3-Factor Alpha	5-Factor Alpha	CAPM Alpha	3-Factor Alpha	5-Factor Alpha
PV-DM	11.04^{**}	7.93^{**}	5.87	10.94^{**}	6.60^{**}	7.45^{**}
t-stat	(2.75)	(2.27)	(1.59)	(2.56)	(2.50)	(2.69)
Word2Vec	3.17	1.35	-0.98	1.26	1.25	-0.66
t-stat	(1.54)	(0.87)	(-0.84)	(0.79)	(0.89)	(-0.52)
PV-DBOW	1.34	1.27^{*}	1.13	5.48^{*}	2.83^{**}	3.8^{***}
t-stat	(1.20)	(1.71)	(1.27)	(1.90)	(2.18)	(3.22)

5.3.2 Main Results

In an effort to improve our five-factor alphas and for further reasons discussed in section 5.2.4, we added a simple momentum-related criterion to the portfolio selection. We termed this strategy as "non-struggling" and the portfolio associated with it as "non-struggling" portfolio or NSP. In this framework we select companies with very similar consecutive 10-Ks (cosine similarity of the PV-DM paragraph vectors is

higher than 0.95) but also with positive previous year returns (starting from April 1st and ending March 30th) It is important to keep in mind that the momentum strategy by itself does not yield any excess returns, meaning there is no momentum premium (see figure 5.1). In fact, over the two decades under study the cumulative returns with the momentum strategy were slightly less than the available universe of stocks returns implying that the criterion has actually a small value effect, also referred to as the mean reversion effect. This means that our portfolio results are not driven by momentum effects which can also be further validated by the fact that the portfolio does not have a statistically significant momentum beta (see Table 5.3).

As seen in Table 5.2 all excess returns, 3-factor alphas and 5-factor alphas of the NSP are higher and statistically more significant compared to the "Semantic Similarity" strategy. Specifically, the value-weighted portfolio reaches a statistically significant 9.75% per year in 3-factor alpha (t= 3.24) and 8.45% per year in 5-factor alpha (t = 2.61). These are extraordinary alphas. In fact, for a comparison, a regression on the highly used, both in academia and the industry, UMD portfolio's returns for the same dates produces a smaller and less statistically significant 3-Factor annual alpha of 6.29 (t=1.84).

In the same table (Table 5.2) we also compare our performance with the corresponding BoW-based with and without momentum strategy performances. In these cases, the only difference is that the companies are selected if the cosine similarity of the BoW vector representations of the previous year's 10-K with the current year's is higher than 0.95. The BoW-based portfolio with and without the same momentum-related criterion show, however, no significant abnormal returns in either case. This further validates the superiority of the PV-DM model in capturing semantic changes in the 10-Ks as well as a synergistic value created by incorporating the previous year's returns

The equal-weighted and value-weighted annual portfolio abnormal returns as well as the statistical significance in Table 5.2 are computed by regressing in each case

the twenty years of compounded returns, while the average number of companies selected each year is 40 for the NSP, 190 for the Bow-based without momentum portfolio and 125 for the BoW-based with momentum portfolio

For the portfolio with the best performance, the NSP portfolio, we also report in Table 5.3 all the factor loadings derived from the time-series regressions using the capital asset pricing model (CAPM), 3-factor and 5-factor models. These loadings are measures of the exposure to the market, size, value, momentum and liquidity risks. We observe statistically significant very small betas which suggest much lower risk as well as statistically significant exposure to the HML factor which shows that our strategy has a value tilt. These observations show that our strategy avoids high beta, high growth stocks while it also selects the least struggling value stocks. This could mean that these companies have moats, i.e. sustainable competitive advantages protecting them from external threats such as rivals or industry disruption. It is extraordinary to see information derived from text to relate to the Fama and French value premium. The rest of the factor loadings, SMB, UMD and PS_VWF are statistically not significant.

5.3.3 Returns evaluation

Finally, Figure 5.1 plots the value and equally weighted cumulative returns for the various portfolios that were constructed. Specifically, it plots the cumulative returns for the "Semantic Similarity" and "Non-struggling" strategies as well as the BoW-based with and without momentum-related criterion portfolios, the whole universe of stocks under consideration portfolio as well as the raw-momentum (Momentum) portfolio and the CRSP market index. For each of these portfolios we plot one chart with the value weighted and one with the equally weighted cumulative returns. In both charts we also add the cumulative returns of the S&P500 and the risk free cumulative returns for comparison. The first thing to notice is that the available universe of stocks is a representative data set with no survivor-biases or other biases of any sort

Table 5.2: **"Semantic Similarity" and "Non-struggling" annual portfolio returns:** This Table reports the annual portfolio excess return, 3-Factor alphas, and 5-factor alphas (market, size, value, momentum, and liquidity) of the long-only portfolio, termed "Non-struggling" portfolio (NSP) which selects companies whose previous year's returns were positive and whose cosine similarity of the Doc2Vec (PV-DM version) vector representations is higher than 0.95. SSP refers to the "Semantic-similarity" portfolio. The performance is compared to portfolios constructed using the BoW model instead. Returns are annualized and multiplied by 100. The t-statistics are shown below the estimates, and the statistical significance at the 1%, 5%, and 10% levels is indicated by ***, **, and *, respectively.

	Value-Weighted			Equally-weighted		
Portfolio	CAPM Alpha	3-Factor Alpha	5-Factor Alpha	CAPM Alpha	3-Factor Alpha	5-Factor Alpha
SSP	11.04**	7.93**	5.87	10.94**	6.60**	7.45**
t-stat	(2.75)	(2.27)	(1.59)	(2.56)	(2.50)	(2.69)
BoW	3.88	3.84	5.1	5.09	3.14	3.57
t-stat	(1.20)	(1.25)	(1.54)	(1.60)	(1.29)	(1.30)
NSP	11.11***	9.75***	8.45**	9.88**	7.40**	6.28**
t-stat	(3.30)	(3.24)	(2.61)	(2.57)	(2.45)	(2.15)
BoW-Mom	3.92	4.33	1.26	2.16	1.74	−1.35
t-stat	(1.51)	(1.56)	(0.44)	(0.70)	(0.52)	(0.71)

as the returns do not deviate from the CRSP market index. Second, the momentum strategy by itself (buying stocks with positive prior year returns and holding them for the next year) by itself does not present any excess returns whatsoever. This shows that "Non-struggling" results are not driven by the momentum effect rather by the changes in year-on-year 10-K-s and the synergistic value that is created.

The final and main thing to notice in Figure 5.1 is the historical performance of both the "Semantic-similarity" and the "Non-struggling" portfolios, SSP and NSP relative to the S&P 500 benchmark, the available universe of stocks and the raw-momentum

Table 5.3: **Regression of 3 Factor and 5 Factor model with the "Non-struggling" portfolio:** This table reports the factor exposure of the long-only "Non-struggling" portfolio. This portfolio selects companies whose previous year's returns were positive and whose cosine similarity of the Doc2Vec (PV-DM version) vector representations is higher than 0.95. Returns are annualized and multiplied by 100. The t-statistics are shown underneath the estimates, while the statistical significance is indicated by ***, **, and * for the 1%, 5%, and 10% levels, respectively.

	Value-Weighted		Equally-Weighted	
Factors	3-Factor	5-Factor	3-Factor	5-Factor
Intercept (α)	9.75^{***}	8.45^{**}	7.40^{**}	6.28^{**}
t-stat	(3.24)	(2.61)	(2.45)	(2.15)
MKTRF	0.48^{***}	0.45^{**}	0.61^{***}	0.59^{***}
t-stat	(3.30)	(2.66)	(4.23)	(3.34)
SMB	0.04	-0.19	0.31	0.10
t-stat	(0.14)	(-0.62)	(1.08)	(0.30)
HML	0.45^{***}	0.34^{**}	0.61^{***}	0.49^{**}
t-stat	(2.96)	(1.74)	(3.94)	(2.48)
UMD	–	-0.12	–	-0.12
t-stat		(-0.50)		(-0.54)
PS_VWF	–	0.49^{*}	–	0.46
t-stat		(1.84)		(1.69)

portfolio. Over a 20-year backtest, these two strategies exhibit significant outperformance to these benchmarks. During this period, $10,000 invested with the SSP at the end of 1999 would have yielded over $200,000 in 2018 compared to only $41,000 for the whole available universe of stocks ($64,820 for the equally-weighted) and $29,233 for the S&P 500. Additionally, $10,000 would have yielded over $140,000 for the NSP, compared to only $35,000 for the value-weighted raw-momentum strategy and $48,900 for the equally-weighted. These results further validate our models' measures performance in capturing changes in 10-K-s associated with future abnormal returns. Another thing to notice is that even though the SSP cumulative returns are larger than the NSP, the 3-factor and 5-factor alphas are higher for the NSP. This

is because NSP carries less risk with more consistent abnornmal returns.

5.4 Final remarks

Measuring modifications and semantic changes from the previous year 10-Ks is challenging because the disclosures are qualitative. Even though our measures are not perfect, they are a step forward in understanding and quantifying these hard to identify changes. However, this work highlight the importance and the real possibilities of knowledge extraction from neural embedding spaces.

We can assert, in light of our results, that neural network embedding techniques outperform in capturing semantic changes associated with future abnormal returns in year-on-year 10-Ks the more widely used state-of-the-art bag-of-words (BoW). This result is evident generating the embedding with Doc2Vec and also averaging Word2Vec embeddings. This was expected since treating words and phrases as discrete symbols fails to take into account the word order and the semantics of the words, while it also suffers from frequent nearorthogonality due to its high dimensional sparse representation.

Previous year returns proved to be a strong contributor to abnormal future returns associated with these changes in year-on-year 10-Ks. Specifically, a portfolio that selects companies whose cosine similarity of the year-on-year PV-DM Doc2Vec representations is higher than 0.95 and the previous year stock return is positive earns statistically significant three-factor and five-factor alphas up to 10% per year.

The main limitations of our approach are related to the computational time required to train these kind of models and to the occasional changes in the companies' executives that should be taken into account to better understand the nature of the changes in the financial reports. Both issues are objects for further analysis that we plan to present in our future works. Our measures are applicable to lots of other cases in which the disclosure is narrative, but the content is unrestricted, the timing is routine, such as CEO letters to shareholders, proxy statements, earnings press releases, and the prepared part of earnings conference calls.

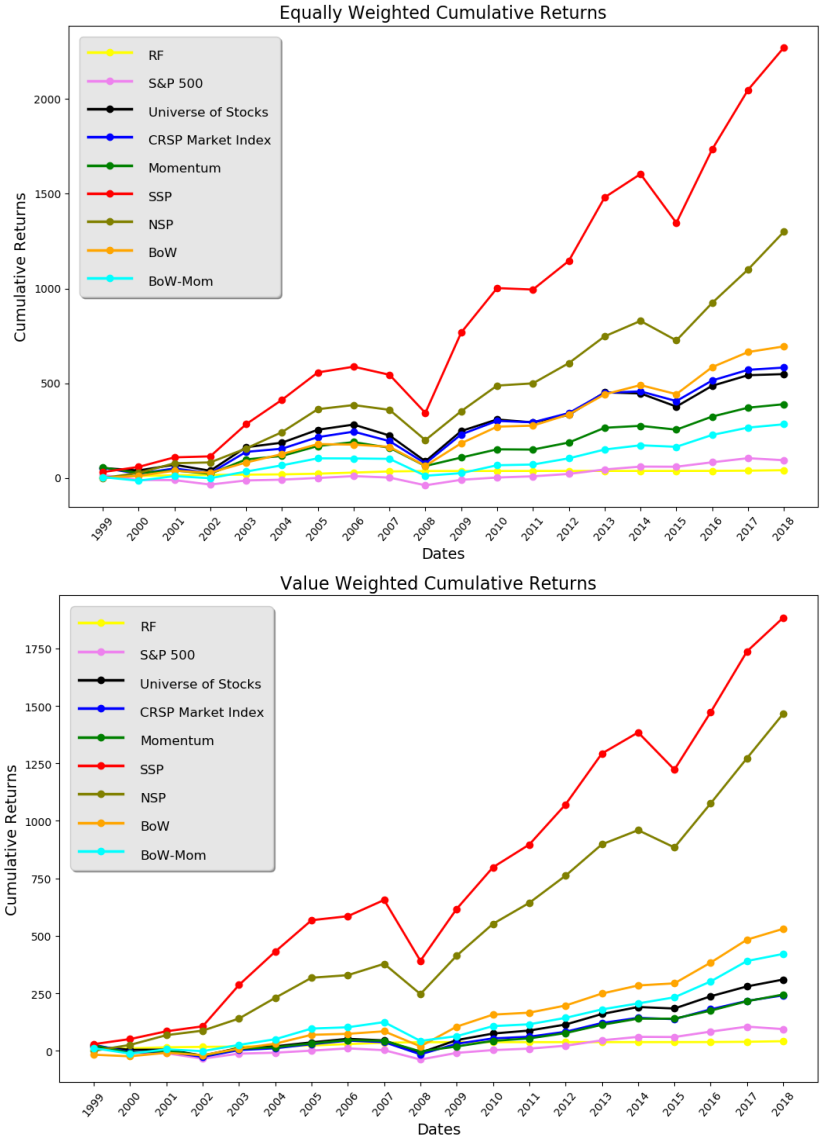


Figure 5.1: Value (UP) and equally (DOWN) weighted cumulative returns from 1999-2018 withs the "Semantic-similarity" (SSP) and the "Non-struggling" (NSP) portfolios. We compare the returns with the buy-and-hold cumulative return of S&P 500, the BoW-based with and without momentum respective portfolios, our universe of available stocks portfolio (Universe of Stocks) as well as the raw momentum (Momentum) portfolio and the CRSP market index. The momentum strategy in the portfolios selects stocks with positive past year return

Part III

Node embedding

Chapter 6

Enhance scalability of Node embedding on large networks

6.1 Proposed solution: The ActorNode2Vec algorithm

Although Node2Vec is presented as a scalable algorithm, the original formulation and implementation involve using a multi-threading solution only for the second phase of the random walk generation (workers parameter) to reduce the computational times. However, experimenting Node2Vec with large networks, it is immediately clear that the first phase of the probabilities computation is the most critical architecture point. This issue is due to the construction of the required probability distribution presented in equation 2.3. The use of a second-order Markov chain considering the second-order neighborhood of each node inevitably leads to a heavy dependency on the number of nodes and particularly on the network density. Increasing the dimension of the network (number of nodes and edges) leads to two main critical issues:

- **Memory requirements become computationally unfeasible**: Memory availability often represents a critical point in large networks mining. Requirements become infeasible on a single calculus node when the probability distribution construction requires to store a vector of probabilities on each edge depending on the second-order Markov chain.

- **The explosion of algorithm's time complexity**: Required time to construct the probability distribution increases depending on the dimension of the graph and on random walks parameters (e.g., number of walks and walk length)

In light of this, this phase of the algorithm represents the first considerable bottleneck of the algorithm. We propose to overcome this issue using an actor-based architecture that enables a reformulation of this step named ActorNode2Vec. The key idea is to distribute the computation of the probabilities distribution in equation 2.3 using several specialized actors that take the burden of this computation, each one for a subset of actors without the need of a preliminary ordering of nodes in the graph.

ActorNode2Vec implements two different strategies to distribute the computational bottleneck of the original algorithm depending on the need of the context. The first one aims to distribute with specialized actors only the probabilities computation part (**Distributed PC**), while the second one uses the same architecture to distribute probabilities computation but also random walks generation (**Distributed PC and RWC**) . How these two modes are efficiently implemented by actors is discussed further in the next sections but it is important to highlight that depending on the input parameters (e.g the walks length) the simple distribution of the probabilities bottleneck is not enough for large networks and it requires to work also on the random walk generation. For further details, see two publications related to this work [93] and [94].

We demonstrate the efficacy of this approach with a large network by analyzing the sensitivity of walk length and number of walk parameters and comparing with Deep walk and an Apache Spark distributed implementation of Node2Vec. Results show that with ActorNode2vec computational times are drastically reduced without losing embedding quality and overcoming memory issues.

6.2 ActoDeS

In order to implement a distributed solution for node embedding, we choose to use ActoDeS (Actor Development System) that is a Java software framework for the

development of concurrent and distributed applications that has been experimented in different application domains (see, for example, [95], [96], [97], [98] [99]). This software framework allows the definition of distributed applications exploiting concurrent objects (actors) [100]. These actors are implemented on the top of some pre-existed Java software libraries and solutions for supporting concurrency and distribution. These actors interact by exchanging asynchronous messages and, after their creation, can change several times their behaviors until they kill themselves. Each behavior has the main duty of processing the incoming messages that match some message patterns. Therefore, if an unexpected message arrives, then the actor maintains it in its mailbox until a behavior will be able to process it, or a behavior kills the actor. The processing of the incoming messages is performed through some message handlers. In response to a message, an actor uses the associated handler for: sending other messages, creating new actors, changing its local state or its behavior, setting a timeout within receiving a new message and finally killing itself. In particular, when a timeout fires, the actor sends automatically a timeout message to itself and the corresponding message handler is executed.

Depending on the complexity of the application and on the availability of computing and communication resources, an application can be distributed on one or more actor spaces. Each actor space corresponds to a Java virtual machine and so a system distributed on some actor spaces can be deployed on one or more computational nodes. An actor space acts as "container" for a set of actors and provides them the necessary services for their execution. An actor space performs its tasks through three main run-time components, (i.e., controller, dispatcher and registry) and through two run-time actors (i.e., the executor and the service provider).

The controller manages the execution of an actor space. It configures and starts the run-time components and the actors of the application, and manages its activities until the end of its execution by using different communication technologies (the current release of the software supports ActiveMQ, MINA RabbitMQ, RMI, and ZeroMQ). The registry is a run-time component that supports actor creation and message pass-

ing. In fact, it creates the references of new actors and supports the delivery of those messages that come from remote actors, by mapping each remote reference onto the local reference of the final destination of the message. The executor actor manages the execution of the actors of an actor space and in some cases creates its initial set of actors. Finally, the service provider actor provides an extensible set of services to the other actors of the application that allows them to perform new kinds of actions (e.g., to broadcast or multi-cast a message and to move from an actor space to another one).

The development of a standalone or distributed application consists in the definition of the behaviors assumed by its actors and in the definition of few configuration parameters that allow the selection of the more appropriate implementation of the actors and of the other run-time components of the application. This solution allows the optimization of some or others execution attributes of an application (e.g., performance, reliability and scalability). Moreover, the deployment of an application on a distributed architecture is simplified because an actor can move to another computational node and can transparently communicate with remote actors.

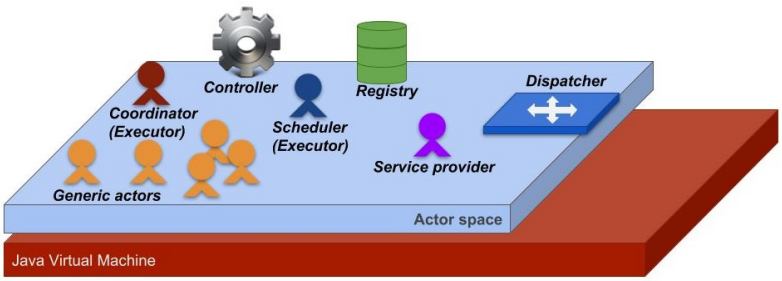


Figure 6.1: The actor space structure in ActoDeS.

6.3 Algorithm and distributed architecture

To implement an actor-based solution, we have defined different actor behaviors and elements that compose the software architecture:

- **Remote Launcher**: It represents the algorithm initialization point, it has the duty of deciding how many and which computational nodes have to be involved in the execution of the algorithm.
- **ActoDeS Broker**: It is a software component implemented in ActoDeS that has the duty of initializing communications among the actor spaces.
- **Node2Vec Initiator**: This behavior has the duty of initializing the algorithm. It requires some starting messages from the remote launcher to set the Node2Vec parameters and the input. network to embed. The actor space that contains the unique actor with this behavior is defined as the primary actor space.
- **Actorspace Initiator (AI)** : It has the duty of initializing the necessary actors for the algorithm in the actor space that do not contain any Node2Vec Initiator (secondary actor spaces). In detail it has the burden of creating the actors that are responsible for the probabilities computation.
- **Coordinator**: This behavior represents the core behavior of the algorithm. Once the initialization operations are terminated, it has to manage all of the architecture actors and manage the entire graph embedding process.
- **Probabilities Manager (PM)** This behavior represents the reformulation of Probabilities computation and random walks generation in the original Node2Vec algorithm. Each actor that assumes this behavior has the duty of computing probabilities only for a subset of nodes considering their second-order neighborhood (Distributed PC) and to build part of random walks that involve its nodes (Distributed PC and RWC).

6.3.1 Starting and initialization steps

The algorithm's execution is performed by the Remote Launcher (RL) introduced in the previous section. The RL represents a server application with the duty to read the input network from memory, choose the embedding parameters, and initialize different ActoDeS actor spaces in the distributed network. It initializes the primary

actor space, requiring the creation of the Node2Vec initiator and sending to this actor the graph and the parameters. At the same time, the RL initializes an ActoDeS broker on a secondary actor space to establish communication among the different actor spaces that build the architecture (Figure 6.2). After this step, the RL main duties are concluded, and it can wait for a notification of the algorithm's termination. Simultaneously, the Node2Vec Initiator actor requests the creation of an Actorspace Initiator on each secondary actor space and sends to each one the entire graph to be embedded and the Node2Vec parameters. The previous ActoDeS broker becomes another Actorspace initiator for its actor space. These different Initiators generate several Probabilities Manager actors (PM) to prepare the architecture for the next steps. The number of created actors is a parameter, but the default value is equal to the number of available logical CPUs on the network node. Each PM actor receives the entire graph and the other parameters. This design choice is motivated by the next steps of the algorithm and does not have to order the nodes of the graph, which would make computational complexity exponential. This choice is also due to taking advantage of the actors' shared memory on a single node because the graphs' operation is read-only. (See Figure 6.3).

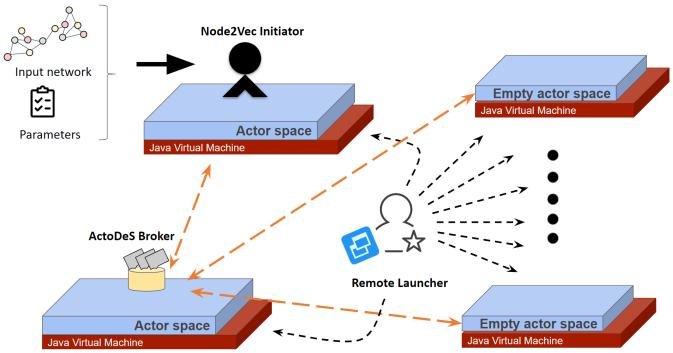


Figure 6.2: Starting step and creation of the actor spaces

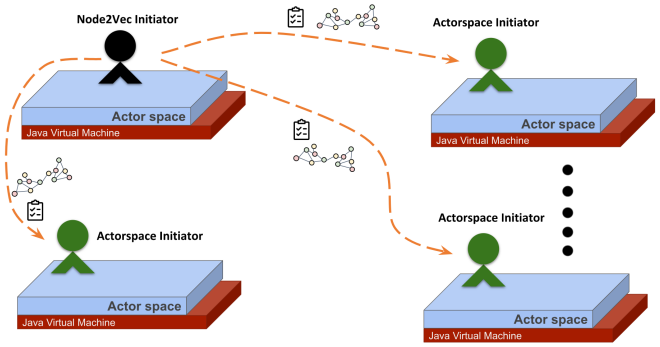


Figure 6.3: Initialization step of ActorNode2Vec

6.3.2 Distributed probabilities computation

This step represents the key difference between the original version of the algorithm and our proposal. In light of the limits presented previously, we propose a reformulation of this step where the required probability distribution is computed not on a single thread and neither with a multi-thread solution but by dividing and distributing the entire computation. The simple multi-thread solution on a single computational node has been excluded because it does not permit to overcome the memory issue presented in section A in the case of a large network. The two models implemented in the algorithm, PC and PC-RWC start to differ exactly from this point. In both cases, we defined a specialized typology of actor's behavior (Probabilities Manager or PM) with the duty of computing probabilities and walks for a subset of the graph nodes depending on the selected mode of operation. These specialized actors are created in the actor spaces on the request of the Coordinator actor. This last actor has to execute the following algorithm steps: It receives all of the PM's references from the Actorspace Initiators and kills the latter as they are no longer necessary. Secondly, the Coordinator sends actors references to each PM actor. This operation is necessary to enable the PMs to understand which subset of nodes is under their responsibilities for the probabilities computation. Figure 6.4 describes this phase of the algorithm.

Probabilities Managers have the duty of computing probabilities following the

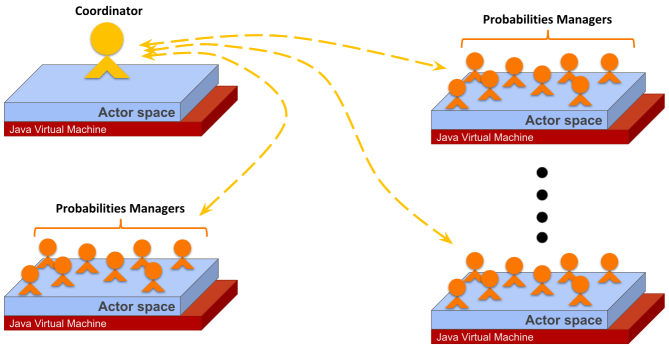


Figure 6.4: Distributed and actor-based probabilities computation

second-order Markov chain model presented in equations 2.3 and 2.4. Each one is responsible for a subset of the graph nodes. PMs know each other's references to identify the number of actors involved in that calculus and divide the nodes set. This choice permits to avoid node ordering, which grows exponentially with the number of nodes. In this way, each PM has the visibility of the entire graph to consider each node's neighborhood and it is computationally efficient using the advantages of the shared memory among the actors on a single working node. The nodes set is divided by each PM using an ordered list of the other involved PMs references, so each actor can detect the subset under its responsibility and who are responsible for the other nodes. Once probabilities have been computed by each actor using the first-order neighborhood, the weights of the edges and the second-order the Markov chain model also prepare the probability distribution to be sampled with the alias method. The output of each PM in fact, are two data structures, one for the probabilities and one for the related alias as the method requires [25]. At the end each PM sends these two structures to the Coordinator, which has the duty of collecting and joining them in a single probability distribution. To summarize, Probabilities Managers compute probabilities for a subset of nodes and edges but they compute also the alias probabilities to improve the following sampling process.

6.3.3 Random walks generation and embedding phase

Once the Coordinator received all of the required probabilities, it builds the probability distribution to be sampled for the random walks generation. It implements a biased coin system to sample from the probabilities and generates the random walks from each node with a fixed length. After that, the Coordinator collects the random walks in a corpus and feeds the Neural Network Model based on Word2Vec described previously. The embedding phase can represent the second bottleneck of the system in terms of time complexity, particularly depending on which technologies are used to implement the model. Some approaches to resolving this issue are presented in the future developments section.

6.3.4 Distributed computation of probabilities and random walks

In this case, Probabilities Managers have a different behavior after probabilities computation. Dealing with large networks can be problematic in terms of memory, although the distributed probabilities computation. This can happen when in a large network, the edge-density is very high and the average nodes degree. In this case, the computational node where the Coordinator lives could not have enough memory to receive walks from the probabilities Managers. In this case is it possible to use ActorNode2Vec in the PC and RWC operating mode. The first step remains the same as in the other mode, but once terminated with the probabilities, each PM starts to compute random walks for its nodes, sampling its partial probabilities distribution. When the next node to be visited is not in its responsibilities set, the PM looks for the target PM in the environment and ask it to continue the walk with its information. When the walk reaches the fixed-length determined by the algorithm parameters, it is sent to the Coordinator. This last one, free of the random walks generation duty, proceeds directly to the embedding phase once collected all of the walks.

6.4 Experiments setup

We experimented ActorNode2Vec with a well-known large network "Enron", to demonstrate the benefits of the actor-based solution and its efficiency in time complexity. The scalability of the actor-based solution overcomes memory issues. We experimented both operation mode of ActorNode2Vec: PC and PC-RWC. In our experimentation we compared our solution with the original Node2Vec implementation in Python and with the Scala-based implementation for Apache Spark, both of them are public and available on the Stanford SNAP laboratory web site. In particular, the Scala version comparison is interesting because it implements a distributed version of the algorithm. We compared our solution also with DeepWalk and, finally, with a Java version of Node2Vec. As previously mentioned, DeepWalk does not present memory issues because it does not compute a probabilities distribution. However to deal with large networks or high algorithm parameters, DeepWalk uses a streaming policy from the main memory to build the truncated random walks. The comparison with this algorithm is more interesting because with a less quality of the embedding, DeepWalk can be thought of as a lightweight version of Node2Vec.

6.4.1 The Enron Email dataset

The Enron Email dataset is a large collection of over 600,000 emails generated by the Enron Corporation's past employees and acquired by the Federal Energy Regulatory Commission during its investigation after the company bankruptcy in 2001. This collection has been processed for several scientific studies (e.g [101], [102]), analyzing discussion threads and the content of the email. These works have identified the most important component in about 37 thousands of email addresses. In particular in [102] they modeled this component as a graph by considering each email address as a node (36692 nodes), and the emails exchanges as undirected edges (183831 edges). Currently it is considered one of the most important standard network in Network Science.

6.4.2 Hardware specification

In order to compare the other algorithms with ActorNode2Vec with the Enron network we used a computer cluster with 4 linux nodes. The most performing one (Node 1) has been used to execute Node2Vec, DeepWalk, Java Node2Vec, the master in Apache Spark and The coordinator of ActorNode2Vec, the others to execute the Remote Launcher, the ActoDeS broker and the secondary actor spaces for the probabilities computation. Hardware specifications are the followings:

- **Node 1**: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz with 8 cores per socket, 2 threads for each core, 64 GB of RAM
- **Node 2**: Genuine Intel(R) CPU T1300 @ 1.66GHz, with 4 cores per socket, 1 threads for each core, 16 GB of RAM
- **Node 3**: Genuine Intel(R) CPU T1300 @ 1.66GHz, with 4 cores per socket, 1 threads for each core, 16 GB of RAM
- **Node 4**: Genuine Intel(R) CPU T1300 @ 1.66GHz, with 4 cores per socket, 1 threads for each core, 16 GB of RAM

6.5 Results

A first analysis concerned an overall comparison between Node2Vec and ActorNode2Vec operating in the PC mode. The results are focused on the time complexity because the proposed algorithm try to address the main bottlenecks of Node2Vec on large networks without changing the neural network at the end of the pipeline. For this reason, we did not make comparison in terms of embedding quality with a machine learning task. In light of this, in order to achieve this result, we choose to compute the embedding of the Enron graph with the following initial configuration of the parameters:

- Embedding dimensions : 100
- P : 1

Table 6.1: Run times with the initial parameters configuration

	Probabilities	Embedding	Total
Node2Vec	244.23 s	390.87 s	635.10 s
ActorNode2Vec (PC)	58.3 s	69.62 s	127.92 s

- Q : 1
- Walk length: 10
- Numbers of walks: 10

In particular with parameters p and q equal to one, we can assume that the DFS and BSF strategies are used uniformly in the random walks generation. This particular choice of P and Q parameters also enables us to have a next fair comparison with the Deep Walk algorithm, where the two searching strategies are balanced. However, p and q do not affect the algorithm's performance. Results of this first comparison are in Table 6.1. In this first experiment, ActorNode2vec required about 80 % less time to embed the Enron network. In this first case, the improvements seem to be related both to less time in probabilities computation and in the embedding phase. We have further analyzed this case by increasing the number of embedding dimension to 300. The embedding phase takes several minutes in both versions of the algorithm because it represents the number of neurons in the hidden layer of the adopted Multi-layer perceptron. It is also to report that the statistical model's training and inference could be more efficient in Python with the Gensim implementation of the neural network than in Java. The results of this case are in Table 6.2. In fact, in this case, ActorNode2Vec took about 28 % less than Node2Vec. However, the improvements are related only to a minus required time of the actor-based probabilities computation.

6.5.1 Sensitivity analysis of the parameters

In order to evaluate how the actor-based solution of ActorNode2Vec improves the time complexity in its two operating modes, we compared our solution also with

Table 6.2: Run times with embedding dimensions @300

	Probabilities	Embedding	Total
Node2Vec	481.22 s	745.16 s	1226.38 s
ActorNode2Vec (PC)	60.28 s	823.75 s	884.03 s

a distributed Scala implementation of Node2Vec available for Apache Spark, with DeepWalk and with a Java version of Node2Vec. These three additional implementations are further analyzed in a fair way with a sensitivity analysis of parameters. In light of the results presented in the previous section, we experimented ActorNode2Vec with the same initial configuration set of the parameters, but further analyzing the results by changing the number of walks and the walks length in a range between 1 and 30. These two analyses are necessary to understand how the actor-based solution of ActorNode2Vec improves the algorithm's time complexity. Walk length and the number of walks are the main parameters that affect the probabilities computation phase in terms of required time. Results concerning the walk length sensitivity are presented in Figure 6.5, while the others concerning the numbers of walks in Figure 6.6.

6.5.2 Comparison between PC mode and PC-RWC mode

The sensitivity analysis highlights a first result in the comparison between the different modes of ActorNode2Vec. As previously anticipated, the PC mode results to be the more efficient when probabilities computation can be the bottleneck in terms of memory. The computational node where the Coordinator is executed can manage the probability distribution. However, in case of a more dense and large network when the random walks generation has to be distributed, the PC-RWC mode requires only 3.9% 10.12% more of time than PC by varying the number of walks and Walks length, respectively.

6.5.3 Comparison with Node2Vec

The main important result concerns the comparison between ActorNode2Vec with Node2Vec on a single machine and its distributed version on Apache Spark. Is it possible to notice in Figure 6.5 and 6.6 both modes of ActorNode2Vec require less time than Node2Vec. In particular, ActorNode2Vec PC requires on average, 73% less time than Node2Vec and 72% less than the Spark version by varying the length of the walks. Considering the number of walks Actornode2vec requires respectively the 81.56 % less and the 74% less than Node2Vec and the Spark version. This results also highlight that the Node2Vec implementation on Spark offers better performance than the original one only when the value of the parameters is below a threshold.

6.5.4 Comparison with DeepWalk

DeepWalk differs from Node2Vec for the absence of a probabilities computation step and implements also a streaming strategy from the main memory to avoid memory issues. However varying the two parameters of interest as in the previous cases it is possible to see that below a very small threshold, its behavior is similar to ActorNode2Vec, but above that value both modes of our solution perform better. In particular, the PC mode requires on average the 19% less time in the number of walks case and the 10.5 % by varying the walks length.

6.5.5 Comparison with a Java version of Node2Vec

In order of further analyze if the better performance of ActorNode2Vec are due to the actor model or to the different Java platform, we implemented the original Node2Vec in Java. This version exploits common parts of ActorNode2Vec, but it does not provide distributed probabilities computation and random walking as the original Python implementation. In both cases (walk length and number of walks) ActorNode2Vec requires less computational time, especially with the grow of the parameter value. In particular, the PC mode requires on average, 35% less time in the number of walks case and 25.2% less by varying the walks length.

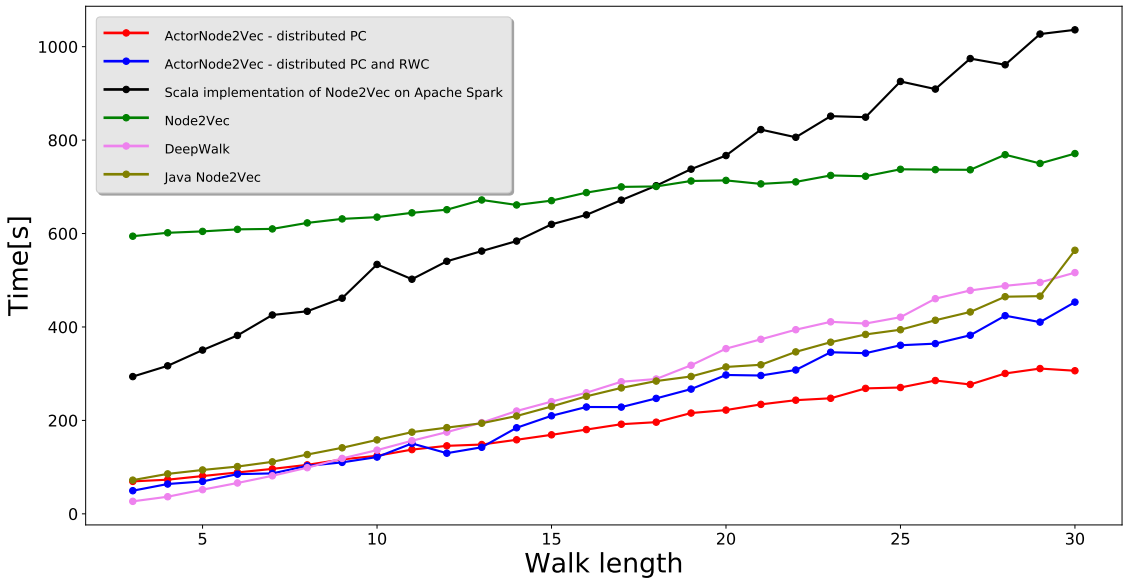


Figure 6.5: Sensitivity analysis of the Walk length parameter

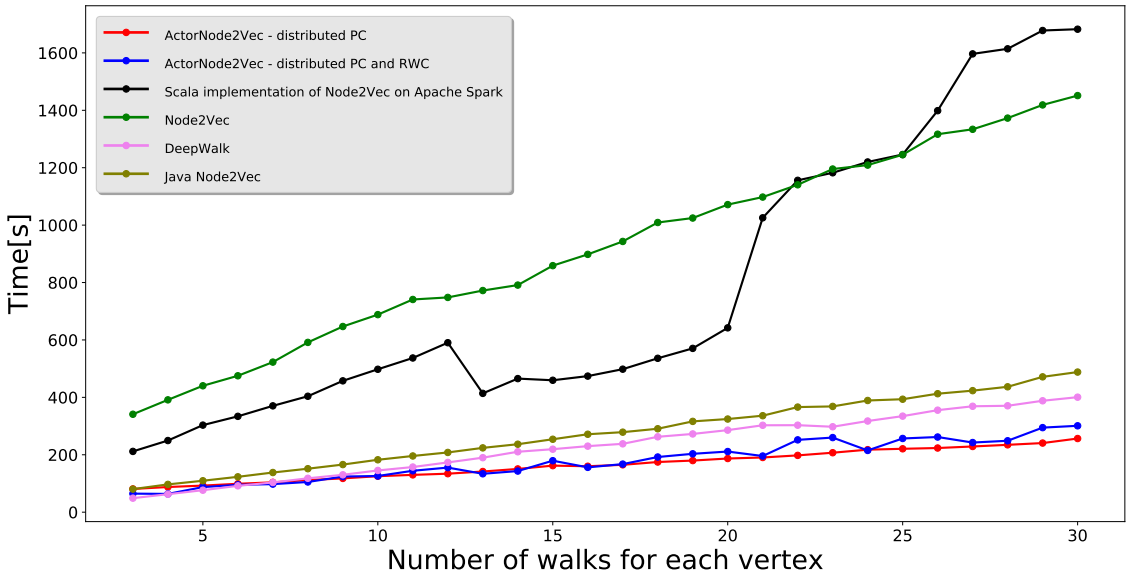


Figure 6.6: Sensitivity analysis of the Number of walks parameter

6.6 Final remarks

This work presents a scalable and distributed version of the Node2Vec algorithm called ActorNode2vec, developed on an actor-based architecture that allows to overcome some limitations in terms of memory requirements and time complexity when applied to large networks. Results show important reduction with respect to other available algorithms in terms of the required computational times, while memory issues are overcome with the scalability of the solution. Some future developments are related to the second bottleneck of the algorithm, that concerns with the time complexity of the Neural Network adopted for the embedding phase. For example an implementation with DeepLearning4j could improve required training times, as the choice of distributing also the training phase. Some other future developments are related to the use of the actors also for a distributed generation of the random walks and the use of different strategies for the network exploration.

Chapter 7

Node embedding in dynamic graphs

The recent advent of random walks-based embedding techniques enabled a more efficient application of machine learning techniques on graphs.

These techniques allow each node of a network to be encoded into an arbitrary low-dimensional vector representation, which can be exploited by statistical learning models. However, the main limitation of these approaches is that the embedding task is solved as an optimization problem on a static snapshot of the graph. In a real scenario, temporal dynamics should be considered with some consequences: new nodes might join the network and get a representation of only these new ones. As a consequence, a new training step over the entire graph is required. Even more, training models with static approaches can have resource-intensive requirements, especially when dealing with large networks

In light of this, an incremental feature learning that builds on top of previously already learned knowledge (previous partial embedding of the network) and well-known properties can be a solution to address both limitations efficiently in real scenarios. Our approach is suitable for graphs whose degree distribution resembles a

power-law function. However, this is not a severe limitation because several works have largely demonstrated that real systems tend to present this node degree distribution. Moreover, the choice of power-law graphs is motivated by the presence of the Hub nodes (nodes with the highest degree). These particular nodes are selected as the more suitable and rich of information to approximate the entire network when performing feature learning of new nodes. However, a different criterion to select the first nodes is possible and it will be further investigate in future works.

This research work presents two main scientific contributions: (a) an incremental feature learning algorithm for node embedding, which exploits properties of power-law distribution and spaces alignment techniques named WalkHub2vec; (b) we demonstrate empirically, by performing node labeling tasks, that a lightweight solution to encode new nodes, based on limited knowledge of the embeddings of the network hub-nodes, can provide comparable or better performances, with respect to static approaches. Experiments have been performed on three well-known graphs, Wikipedia, Blogcatalog, and Protein-Protein interactions, that are considered a benchmark in graph embedding literature.

7.1 Context

In the last two years, several works have tried to address the issue of node embedding in a dynamic context, and this increasing production of different solutions highlights the importance of the topic both in academia and industry. The most used approach in the literature is based on modeling dynamics as a sequence of static snapshots.

The first algorithm, based on network snapshots, is the Continuous-time Dynamic Network Embedding (**CTDNE**) [103], proposed for the link prediction task. It relies on the concept of temporal random walks, using timestamps to preserve the time order of edges and thus their temporal dependencies. It can preserve temporal structural properties for link prediction, but it requires retraining at every change in the network.

Other successful snapshots-based works, relying on a different approach, are **DynGEM** [104], the one proposed in [105], **Dynnode2vec** [106] and **Dyngraph2vec** [107]. Dyngem and Dyngraph2vec estimate node representation with a deep autoencoder and an LSTM, respectively, by initializing network weights with the previously computed embedding at time $t-1$. Both algorithms address the problem of graph visualization and link prediction without node classification. Their main limitation is their use of their own learning model (PropSize), so they only work with this embedding method.

The algorithm proposed in [105] is an extension of the Skip-gram model and, in particular of the LINE algorithm. Its interesting contribution is that when a new node joins the network, changes in the representations of other nodes are limited. In light of this, it computes a new representation with new walks only for a selected set of vertices. It offers impressive performance in several tasks, including a simple task of node classification, independently to the static algorithm used for random walk generation.

Finally, Dynnode2vec can be seen as a combination that takes the advantages of CTDNE and the algorithm in [105]. It uses the concept of evolving walk generation and computes new representation only for new nodes that join the graph in a particular timeframe by initializing the hidden weights of the neural network with the ones used to embed the snapshot at time $t-1$.

More recently, in [108], the authors proposed an algorithm, named **Credit probing network embedding**, that aims to compute node embedding for evolving networks with a partial monitoring of the network. The basic idea is to update each node vector dealing with it as a timestamps vector and to exploit a probing system to decide which node and when it should be updated. However, in this case, the evaluation task is represented by the link prediction task, and the source code is not currently available.

Finally, in [109], the authors propose **tNodeEmbed**, an algorithm based on Node2Vec that learns new nodes representation by aligning snapshots using Orthogonal Procustes [110] and then training an LSTM for a given task to optimize feature learning. It is not easy to assert which one is the best, because they are experimented on different datasets and with some gaps for what concerns other more difficult tasks like Node classification. The only tNodeEmbed takes into consideration the task of multi-class node classification, i.e., the benchmark task allowing a comparison with static approaches.

This last algorithm reaches very interesting performance, but a complete comparison with the static case remains difficult. The authors experimented with tNodeEmbed on different datasets and the used parameters for embedding are unclear. From direct experiments and the analysis of recent literature, we can assert that the idea of dynamic network embedding based on snapshot does not solve scalability issues and retraining is time-consuming, although less than the static case under certain conditions. In particular, these are important limitations when dealing with real networks that can often be large and dense. We have to be able to maintain previous embeddings and sometimes the set of previous walks whenever there is a change.

It is possible to conclude that several issues and open problems are present in dynamic network embedding. However, considering the recent growth of this research line, all of the works reviewed in this section propose interesting approaches with important progress that trace a way to more generalized solutions. In this work, we take into account specifically the multi-class node labeling task in light of *(i)* the reported lacks in state of the art, and *(ii)* the different issues which this case presents, with respect to the more studied case of link prediction.

7.1.1 Properties of power-law graphs

WalkHub2Vec founds its theoretical justifications in several properties of power-law graphs (Scale-free networks). The importance of this particular category of networks has been widely discussed in several research fields (E.g. [111],[112],[113]). The

main reason is related to the fact that it fits the relationships among entities in many real systems, in various domains. The difference between a random network and a power law one lies in the probability distribution, which better describes a node's choice with a particular degree. In random networks, the randomness of links generation brings to a Poisson distribution and a lack of a component composed of nodes with a higher degree than other nodes, i.e., the hub nodes [114]. On the other hand, real systems are often better described by a power-law because only a few nodes present a high degree, while most of the remaining nodes have a small degree. When a system presents a power-law degree distribution, it means that the probability of randomly selecting a node with degree K is well approximated by Equation 7.1

$$P_k \sim k^{-\gamma} \tag{7.1}$$

This difference has an important consequence on how networks tend to grow. In power-law graphs, the evolution is well described by the Barabási-Albert Model [115]. This model assumes that in power-law graphs (or Scale-free graphs) the growth is described by a property called **Preferential Attachment**. This property claims that the probability $\Pi(k)$ of a new node to be connected with node i depends on the degree k_i, as in Equation 7.2. That means that the growth in this kind of graph follows a probabilistic mechanism where a new node is free to connect to any node in the network, whether it is a hub or has a single link. However, if a new node has a choice between a degree-two and a degree-four node, it is twice as likely that it connects to the degree-four node [114]. This behavior related to the power-law distribution is well known also as the Pareto principle, or 80/20 rule [116]. This rule can be summarized, saying that roughly 80% of the effects come from 20% of the causes. For example, as Pareto noticed in the 19th century, 20% of the population earned most of the money, while most of the population (80%) earned rather small amounts. Thus, it is likely that new incomes are destined to that 20% of people. Similarly, if a new node joins a network, it will likely be connected with the graph's 20% of hub nodes. This property relies on the long-tail of the power law distribution: in this kind of graphs, it is possible to observe that most nodes have a small degree while few nodes have a big degree (the so called long-tail). For this reason, as illustrated in [114] in power-law

graphs is present a giant component that is composed by hub nodes (nodes with a high degree) that connect most of the normal nodes with a small degree.

$$\prod(k_i) = \frac{k_i}{\sum_j k_j} \tag{7.2}$$

7.2 WalkHub2Vec: A meta-algorithm for node embedding in dynamic networks

In this section we describe the incremental feature learning algorithm for node embedding. In light of the properties summarized in Section 7.1.1, we take into account power-law graphs. First of all because WalkHub2Vec aims to be a solution to embed real systems graphs, thus this choice does not represent a limitation. Secondarily, to the best of our knowledge, these properties have not been taken into account by previous works.

Finally, as we demonstrate in the result section, these properties represent a useful tool to address the dynamical case of a new node joining the network and in need for a vector representation. Also, we believe that an important part of the information required by the node classification can be directly found in the hubs component of each graph. The algorithm that we present is based on DeepWalk for simplicity, but it can be used with other embedding solution (e.g. Node2Vec or LINE) and also combining them in different moment. Additionally, being it based on a static embedding of a network, it is possible to retrieve past trained embeddings of a network to obtain the representation of new nodes without retraining.

Every time a new node has to be embedded with respect to the previous static embedding, we seek to optimize a variant of the problem in Equation 2.2 specifically for incremental feature learning. This optimization maximizes the probability of observing a second-order neighborhood among the hubs component for the new node i, conditioned on an aligned feature representation between the target original space and a lightweight embedding of the hubs, or in other words the embedding of node i

with respect to the original embedding space:

$$\max Pr(N_H(i)|R \cdot f_H(i)) \tag{7.3}$$

Where R is the rotational matrix necessary to align the drifted embedding space and is later discussed in detail. Being A defined as the static algorithm chosen for the unique snapshot of the graph, WalkHub2Vec has the following steps:

1. It computes the static embedding of the network G with A following the specific parameters of the considered algorithm and solving the general optimization problem in Equation 2.2. Given a graph G we define also $E_{G_{\{G\}}}$ as the embedding of G made with respect to G itself.

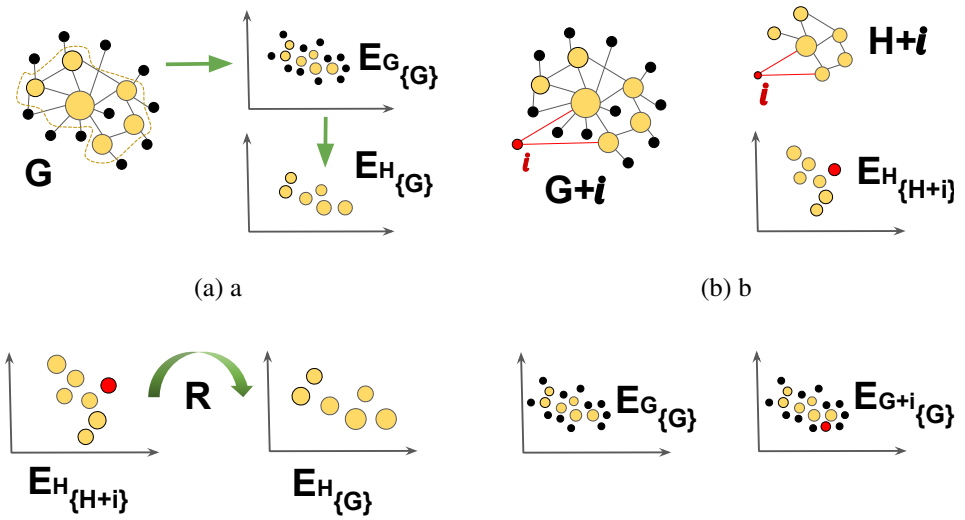
2. The algorithm extracts the hub component of the network by selecting the sub-graph H induced by the percentage λ of nodes with the highest degree. The parameter λ is set as default equal to 20 in light of the Pareto principle. For construction $E_{H_{\{G\}}} \subset E_{G_{\{G\}}}$ without any other computation (Figure 7.1a).

3. When a new node i joins the network, it is considered a graph composed by the sub-graph H plus the node i: $H+i$. In the rare case when the new node in the power-law graph has not links with the hubs, a single random walk from i to one of the hubs is considered in the construction of $H+i$. Given the smaller network $H+i$ a lightweight embedding $E_{H_{\{H+i\}}}$ is computed with A or an arbitrary algorithm (Figure 7.1b).

4. In order to get the embedding of node i with respect to the original embedding space, the algorithm proceed to align $E_{H_{\{H+i\}}}$ with $E_{H_{\{G\}}}$ by learning the optimal rotation matrix. The alignment involves only nodes' representations who belong to the second-order neighborhood of i in the hub component (Figure 7.1c). Alignment details are discussed later in 7.2.1.

5. Once the rotational matrix R is computed, the embedding of i with respect of

the original embedding space is the dot product between R and the embedding of i with respect to $H + i$ space (Figure 7.1d).



(a) a (b) b

(c) c (d) d

Figure 7.1: (a) First and second steps of the algorithm; The embedding of the hub component is extracted as a subset of the entire embedding E_G(b) Third step: A new node i joins the network and a new embedding of only the H sub-graph is computed; (c) The two embedding spaces have all nodes in common expect for the new node i. They have to be aligned to get i coordinates with respect to the original space. Embedding alignments by learning the optimal rotation matrix; (d) Node i with its resulting embedding in the original space

7.2.1 The orthogonal Procrustes problem

The orthogonal Procrustes problem [110] is a matrix approximation technique that aims to learn the orthogonal rotational matrix R which closely maps a matrix A to a matrix B (Equation 7.4). It is based on the Frobenius norm and a closed-form solution

is provided with the Single Value Decomposition (SVD).

$$R = argmin_{\Omega}||\Omega A - B||_F \ \ subject\ to\ \ \Omega\Omega^T \tag{7.4}$$

In WalkHubs2Vec, matrix A is represented by the embedding of the second-order neighborhood of node i in $E_{H_{\{H+i\}}}$ and respectively the correspondents in $E_{H_{\{G\}}}$. In order to increase the quality of the alignment the two matrix are previously pre-processed following the full Procrustes superimposition [117]. It requires that matrices, that can be seen as geometric shapes to be aligned, are previously uniformly scaled and translated to have the average value in the origin.

7.3 Experimental results

In this section we provide an overview of the datasets and methods which we will use in our experiments. Code and data to reproduce our results are available on GitHub (see footnote 1). In order to evaluate WalkHubs2Vec and the related methodology, we performed two different kind of node labelling tasks: multi-label classification and multi-class classification. The difference is that in the first one each node can have more than one label at the same time and each one has to be correctly predicted. For both tasks, with the aim of a fair comparison between our method and the static techniques, we used the exact experimental procedure as in [118, 119], DeepWalk [120] and Node2Vec [23] and their common datasets. Specifically, we randomly selected a percentage of the labeled nodes between 10 and 90%, and use them as training data. The rest of the nodes are used as test set. We repeated this process 10 times, and we report the average performance in terms of both Macro-F1 and Micro-F1. The machine learning model used for both tasks is a One-Vs-Rest logistic regression. [1]

7.3.1 Datasets

For the multi-label classification we considered three networks: PPI and Wikipedia. We selected these networks as a benchmark because have also been used to evaluate Node2Vec and DeepWalk in their respective papers.

[1] https://github.com/gfl-datascience/walkHub2vec

- **PPI**: Protein-Protein interactions for Homo Sapiens, is a network with 3,890 nodes, 76,584 edges and 50 different labels, where labels represent the biological states.

- **Wikipedia**: This network is composed by 4,777 nodes with 40 different labels and 184,812 edges. It is a co-occurrence network of word appearing in the first million bytes of the Wikipedia dump. Labels are Part-of-Speech (POS) tags associated to each word.

- **Blogcatalog**: It is a social network of relationships among blogger authors on the BlogCatalog website. The network has 10,312 nodes, 333,983 edges and 39 different labels that represent blogger interest inferred through metadata.

For the multi-class classification task we used Cora, that is one of the most famous in literature for this task (E.g. [19],[121]) with its various versions. It is a citation network composed by papers about different research areas of Computer Science and their citations. We used the largest version available with all of the nodes and the labels [122]. The network has 23,567 nodes, 93965 edges and 10 classes, which represent the topic of the articles.

7.3.2 Experimental setup

In light of the reasons explained in Section 7.1 about a necessary fair comparison between static methods and dynamic ones, we used the same dataset used to evaluate Node2vec and DeepWalk. Unfortunately, these datasets have not temporal information. This has not been a significant limitation for our evaluation, since we do not aim to solve a temporal problem but an incremental one, when a new generic node joins the network.

Under this hypothesis, we selected the 10% of the nodes with the lowest degree in each dataset to simulate their appearance (I set). This choice is also motivated by the idea that the leaf nodes with few links are often under evaluated by the sampling edges techniques used in the Skip-gram model. In order to prove the effectiveness

of our methodology, we also treated the moments at which they join the network as totally independent events. In other words, we do not consider the possibly existing edges in the sub-graph induced by the I set. The rest of the network represents the first static snapshot used as input for WalkHubs2Vec.

As static embedding we used predominantly DeepWalk and in one case also Node2vec as demonstration that our algorithm and the methodology can deal with any node embedding technique. We also made a comparison with these two algorithms when the entire networks are embedded statically. We used the same hyper-parameters proposed in the respective papers of DeepWalk and Node2Vec, without any optimization of these parameters.

- Dimension = 128
- Walk lenght = 10
- Number of walk = 80
- P and Q (Node2vec) both equal to zero to compare with DeepWalk

We also used in each case the parameter $\lambda = 20$, in this way the static network is divided into two groups: 20% of hubs and the remaining 80% unconsidered for the embedding.

7.3.3 Multi-label node classification

In this section we report the results of the multi-label classification on PPI, Blogcatalog and Wikipedia. We experimented our algorithm with the previously introduced methodology by selecting different portions of nodes as training set, in the range between 10% and 90%. However, we report only results for the case of 50%, because we are interested in evaluating how features, computed incrementally on the 10% of the nodes of I set, can affect the prediction results. This condition became more evident when we randomly selected large portions of the network in the 10 runs. Results are presented in Figure 7.2.

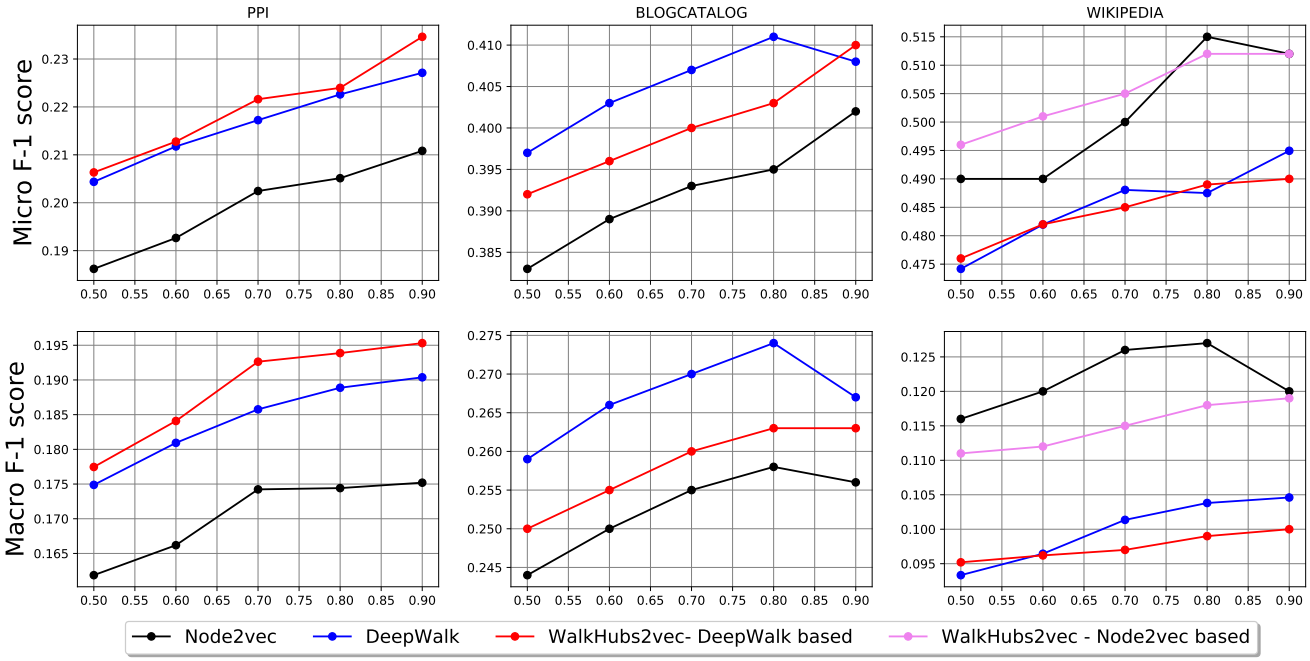


Figure 7.2: Performance evaluation of different benchmarks on varying amount of labeled data used for training. The x axis denotes the fraction of labeled data, whereas the y axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores respectively.

7.3.3.1 Protein-Protein Interactions

On this dataset, WalkHubs2Vec performs slightly better than Node2Vec and Deepwalk both with Micro and Macro F-1 score. In particular it is interesting the result when the selected percentage of training is the 90% and probably most of the nodes of I set have been considered in the 10 runs.

7.3.3.2 BlogDatalog

On this dataset, WalkHubs2Vec performs in a comparable way with DeepWalk with an average divergence of -0.004 and -0.009 in terms of respectively micro and macro

F-1 score.

7.3.3.3 Wikipedia

Unlike the two previous networks, on Wikipedia Node2Vec performs in general better than DeepWalk. In light of this, we present results obtained with WalkHubs2Vec based on DeepWalk but also based on Node2Vec. In both cases WalkHub2Vec performs in a comparable way than the static techniques. The average divergence between Node2vec and the incremental case is 0.003 and -0.006 for micro and macro F-1 respectively. As expected, the version of WalkHubs2Vec based on Node2Vec performs bettern than the one based on DeepWalk as reflection of the static performance of the algorithms.

7.3.4 Comparative analysis for multi-label classification

Several features of the selected datasets should be considered, to get an overall evaluation of our algorithm in the multi-label classification task. First of all, each network presents a power-law distribution for node degree, but Wikipedia graph has a more significant cut-off than the others. Practically, on this dataset, the power-law distribution is more acceptable after a certain degree (around 10), while before this value, a log-normal distribution is more evident. In our opinion, this fact is what makes the difference with results we obtained with the PPI dataset since the density of the two graphs is quite similar (PPI: 0.005, Wikipedia: 0.003) and the average node degree (PPI: 19, Wikipedia: 15).

Another critical factor is how the topological structure affects the results and how the number of labels that have to be predicted. We analyzed this aspect more by comparing the three networks in terms of modularity for community detection and connected components. In PPI, where we have better results than the other algorithms, the number of labels is 50, modularity is 0.337, and it is possible to find 43 different clusters (or communities) and 35 connected components. That means that nodes tend to arrange in groups with the same labels, and thus, the classification task results to

be more simple. In the Wikipedia network, with a Modularity equal to 0.2, we found only 12 clusters, a single connected component with 40 different labels that have to be predicted. Finally, Blogcatalog resulted in a more difficult case: with a Modularity equal to 0.24 we found 7 different clusters, one single connected component when the number of labels is 39.

In light of this, we hypothesize that exploiting the power-law properties can make a significant difference when the labels reflect also the topological structure of the network, and on the other hand, have comparable results in the other cases. However, in these last cases, having comparable results by using only the 20 percent of nodes to compute the embedding is an interesting result that should be more analyzed in future works.

7.3.5 Multi-class node classification

In this section, we report the results of the multi-class classification on Cora with WalkHubs2Vec based on DeepWalk with the static case where the entire network is embedded with Node2Vec and DeepWalk. In this case, WalkHubs2Vec outperforms the baseline methods both in micro and macro F-1 score. The average gain of WalkHubs2Vec is about 0.01 in both metrics. Results are presented in Figure 7.3.

7.4 Final remarks

In this research work we have addressed the problem of incremental feature learning for node embedding in power-law graphs. The work is focused on the idea of taking advantages of the scale-free property of this category of graphs, to compute embedding of new nodes without retraining the learning model in the node classification context. We propose also an implementation of this methodology, WalkHubs2Vec, that can deal with any embedding techniques. Results demonstrate how the combination of partial embeddings based on the hubs component and embedding alignment can solve the problem with good results in terms of features quality for the new nodes. WalkHubs2Vec reaches equal or slightly better results when compared to well-known

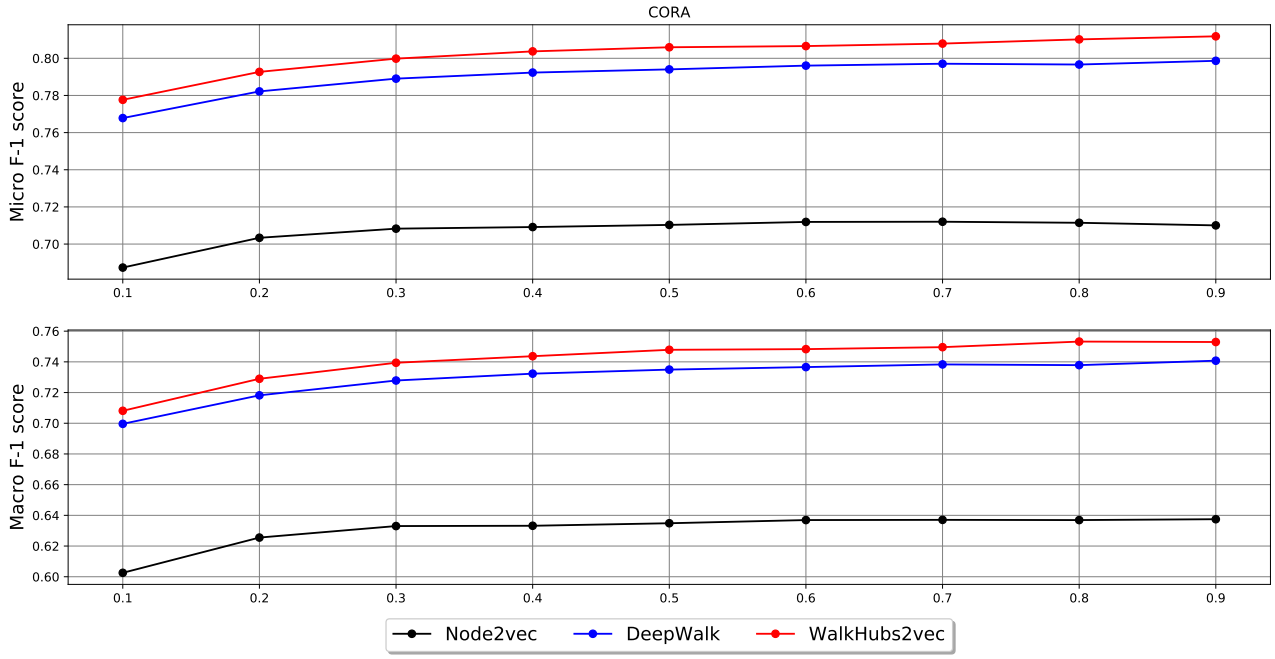


Figure 7.3: Performance evaluation on Cora of different benchmarks on varying the amount of labeled data used for training. The x axis denotes the fraction of labeled data, whereas the y axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores respectively.

static methods, as Node2vec and DeepWalk. Future developments are related to performing different tasks with this methodology (e.g. link prediction) and tests with temporal data. Moreover, it would be interesting to use a similar approach also in word embedding and other embedding contexts.

Chapter 8

Conclusions

All of the works described in this thesis are related to the representation learning, as a preliminary task achieved using neural network embeddings. This thesis has two main goals.

The first one is to analyze the opportunity of designing representation learning tasks on unstructured data to extract new knowledge. This first goal is achieved with a novel methodology that extracts and highlights information by exploiting human mobility metrics and similarity metrics over the embedding spaces. In chapter 3, we presented two contributions as use cases of knowledge extraction from embedding spaces.

In particular, the one related to the analysis of scholars represents a first attempt to combine mobility metrics for the first time in a non-geographical context. As shown by the results, it is possible to extract useful knowledge by building a semantic space and exploiting it with the Radius of Gyration. Future works are related to applying this methodology to new contexts to make also quantitative analysis other than qualitative. For example, it is possible to use the same methods to understand how a single change its way of writing (for example, in arts, music, but also in the financial domain).

The second contribution is exploiting neural network embedding to understand American companies' changes while reporting their 10Ks statements. As shown by

the results, neural embedding techniques outperform in both cases (based on word2vec and in doc2vec) other methods presented in the state of the art (See Lazy prices and bag-of-word in section 3.2). Indeed, by building a novel portfolio selection algorithm by measuring cosine similarity over the semantic space, we can outperform the market and the state of the art in this field.

The second goal of this thesis is to overcome the main limits of representation learning on graphs. In chapter 4, we presented two more contributions in this direction.

The first one, named ActorNode2Vec is an actor-based version of the Node2vec algorithm that distributes random walks and probabilities computation exploiting different agents that organize themselves and enable them to perform representation learning over large-scale networks. The results demonstrate that distributing all of the steps before the neural network training permits to save more than 80% of computational time in a normal context.

The second one, named WalkHub2vec, is an algorithm for incremental feature learning to deal with dynamic graphs. In particular, this work aims to learn the representation of a single new node by using only the 20% of hub nodes of the graph and not the entire network. Experiments show that the algorithm has comparable performance with static approaches. It means that it is not necessary to compute a new static snapshot of the graph and a new embedding just to encode the new entry. Indeed, this work wants to demonstrate that exploiting power-law properties can be a good approach when the graph's changes are related to a few nodes, rather than computing the embedding again for the entire graph. Future works are related to more experimentation of this algorithm but also novel approaches for dynamic graph embedding.

Candidate's publications related with this thesis

Extended description of the work described in this thesis can be found in:

- ActorNode2Vec: An Actor-based solution for Node Embedding over large networks. G Lombardo, A Poggi. Intelligenza Artificiale 14 (1), 77-88.
- Neural network embeddings on corporate annual filings for portfolio selection. G Adosoglou, G Lombardo, PM Pardalos. Expert Systems with Applications, 114053.
- A multi-agent architecture for data analysis. G Lombardo, P Fornacciari, M Mordonini, M Tomaiuolo, A Poggi. Future Internet 11 (2), 49.
- A combined approach for the analysis of support groups on Facebook-the case of patients of hidradenitis suppurativa. G Lombardo, P Fornacciari, M Mordonini, L Sani, M Tomaiuolo. Multimedia Tools and Applications 78 (3), 3321-3339
- A Scalable and Distributed Actor-Based Version of the Node2Vec Algorithm. G Lombardo, A Poggi. WOA 2019.

Bibliography

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[4] Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A Persson, Gerbrand Ceder, and Anubhav Jain. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95–98, 2019.

[5] David Reinsel, John Gantz, and John Rydning. The Digitization of the World From Edge to Core. Technical report, International Data Corporation, 11 2018.

[6] Jeff Desjardins. What happens in an internet minute in 2019. Technical report, World Economic Forum - Visual Capitalist, 2019.

[7] Henry Leung and Simon Haykin. The complex backpropagation algorithm. *IEEE Transactions on signal processing*, 39(9):2101–2104, 1991.

[8] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[9] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[11] Yann Le Cun and Yoshua Bengio. Word-level training of a handwritten word recognizer based on convolutional neural networks. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, volume 2, pages 88–92. IEEE, 1994.

[12] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[13] Geoffrey E Hinton. Distributed representations. 1984.

[14] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[15] Predrag Radivojac, Wyatt T Clark, Tal Ronnen Oron, Alexandra M Schnoes, Tobias Wittkop, Artem Sokolov, Kiley Graim, Christopher Funk, Karin Verspoor, Asa Ben-Hur, et al. A large-scale evaluation of computational protein function prediction. *Nature methods*, 10(3):221, 2013.

[16] Anton Kocheturov, Mikhail Batsyn, and Panos M Pardalos. Dynamics of cluster structures in a financial market network. *Physica A: Statistical Mechanics and its Applications*, 413:523–533, 2014.

[17] Gianfranco Lombardo, Paolo Fornacciari, Monica Mordonini, Laura Sani, and Michele Tomaiuolo. A combined approach for the analysis of support groups on facebook-the case of patients of hidradenitis suppurativa. *Multimedia Tools and Applications*, 78(3):3321–3339, 2019.

[18] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[19] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[20] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[21] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[23] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[24] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[25] Alastair J Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):253–256, 1977.

[26] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[27] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[28] Fatemeh Soleimani-Roozbahani, Ali Rajabzadeh Ghatari, and Reza Radfar. Knowledge discovery from a more than a decade studies on healthcare big data systems: a scientometrics study. *Journal of Big Data*, 6(1):1–15, 2019.

[29] Jipeng Qiang, Ping Chen, Tong Wang, and Xindong Wu. Topic modeling over short texts by incorporating word embeddings. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 363–374. Springer, 2017.

[30] Alexander Veremyev, Alexander Semenov, Eduardo L Pasiliao, and Vladimir Boginski. Graph-based exploration and clustering analysis of semantic spaces. *Applied Network Science*, 4(1):104, 2019.

[31] Carol Friedman, Pauline Kra, Hong Yu, Michael Krauthammer, and Andrey Rzhetsky. Genies: a natural-language processing system for the extraction of molecular pathways from journal articles. In *ISMB (supplement of bioinformatics)*, pages 74–82, 2001.

[32] Matthew C Swain and Jacqueline M Cole. Chemdataextractor: a toolkit for automated extraction of chemical information from the scientific literature. *Journal of chemical information and modeling*, 56(10):1894–1904, 2016.

[33] Edward Kim, Kevin Huang, Adam Saunders, Andrew McCallum, Gerbrand Ceder, and Elsa Olivetti. Materials synthesis insights from scientific literature via text extraction and machine learning. *Chemistry of Materials*, 29(21):9436–9444, 2017.

[34] Scott Spangler, Angela D Wilkins, Benjamin J Bachman, Meena Nagarajan, Tajhal Dayaram, Peter Haas, Sam Regenbogen, Curtis R Pickering, Austin Comer, Jeffrey N Myers, et al. Automated hypothesis generation based on

mining scientific literature. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1877–1886, 2014.

[35] Liang Wang, Zhiwen Yu, Bin Guo, Tao Ku, and Fei Yi. Moving destination prediction using sparse dataset: A mobility gradient descent approach. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(3):37, 2017.

[36] Tian Qin, Wufan Shangguan, Guojie Song, and Jie Tang. Spatio-temporal routine mining on mobile phone data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5):56, 2018.

[37] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[38] Luca Pappalardo, Salvatore Rinzivillo, Zehui Qu, Dino Pedreschi, and Fosca Giannotti. Understanding the patterns of car travel. *The European Physical Journal Special Topics*, 215(1):61–73, 2013.

[39] Luca Pappalardo, Dino Pedreschi, Zbigniew Smoreda, and Fosca Giannotti. Using big data to study the link between human mobility and socio-economic development. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 871–878. IEEE, 2015.

[40] Fang Yao and Yan Wang. Tracking urban geo-topics based on dynamic topic model. *Computers, Environment and Urban Systems*, 79:101419, 2020.

[41] Lauren Cohen, Ulf Axelson, Nick Barberis, John Chalmers, Karl Diether, Irem Dimerci, Joey Engelberg, Umit Gurun, Gerald Hoberg, Xing Huang, Hunter Jones, Bryan Kelly, Jonathan Karpoff, Dana Kiku, Patricia Ledesma, Dong Lou, Asaf Manela, Ernst Maug, Craig Merrill, Mike Minnis, Toby Moskowitz, Peter Nyberg, Cesar Orosco, Frank Partnoy, Mitchell Petersen, Christopher Polk, Taylor Nadauld, Krishna Ramaswamy, Samantha Ross, Alexandra Niessen-ruenzi, Stefan Ruenzi, Dick Thaler, Pietro Veronesi, Sunil Wahal, Daniel Weagley, and Lauren Cohen. Lazy Prices 2019. 2019.

[42] Feng Li. Annual report readability, current earnings, and earnings persistence. *Journal of Accounting and Economics*, 45(2-3):221–247, 2008.

[43] Min Peng, Jiahui Zhu, Hua Wang, Xuhui Li, Yanchun Zhang, Xiuzhen Zhang, and Gang Tian. Mining event-oriented topics in microblog stream with unsupervised multi-view hierarchical embedding. *ACM Trans. Knowl. Discov. Data*, 20(3):38:1–38:26, April 2018.

[44] Aminul Islam and Diana Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10, 2008.

[45] Mark EJ Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the national academy of sciences*, 101(suppl 1):5200–5205, 2004.

[46] Jie Ren and Richard N Taylor. Automatic and versatile publications ranking for research institutions and scholars. *Communications of the ACM*, 50(6):81–85, 2007.

[47] Éric Archambault, David Campbell, Yves Gingras, and Vincent Larivière. Comparing bibliometric statistics obtained from the web of science and scopus. *Journal of the Association for Information Science and Technology*, 60(7):1320–1326, 2009.

[48] Judy F Burnham. Scopus database: a review. *Biomedical digital libraries*, 3(1):1, 2006.

[49] P. Fornacciari, M. Mordonini, M. Nonelli, L. Sani, and M. Tomaiuolo. Knowledge discovery on scopus. *CEUR Workshop Proceedings*, 1959:1–12, 2017. Conference of 3rd International Workshop on Knowledge Discovery on the WEB, KDWEB 2017.

[50] Alfredo Yegros-Yegros, Ismael Rafols, and Pablo D'Este. Does interdisciplinary research lead to higher citation impact? the different effect of proximal and distal interdisciplinarity. *PloS one*, 10(8):e0135095, 2015.

[51] Qiu Fang Ying, Srinivasan Venkatramanan, and Dah Ming Chiu. Modeling and analysis of scholar mobility on scientific landscape. In *Proceedings of the 24th International Conference on World Wide Web*, pages 609–614. ACM, 2015.

[52] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.

[53] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[54] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247, 2014.

[55] A. Harzing, S. Alakangas, and D. Adams. hia: an individual annual h-index to accommodate disciplinary and career length differences. *Scientometrics*, (99):811–821, 2014.

[56] Hugo Barbosa, Marc Barthelemy, Gourab Ghoshal, Charlotte R James, Maxime Lenormand, Thomas Louail, Ronaldo Menezes, José J Ramasco, Filippo Simini, and Marcello Tomasini. Human mobility: Models and applications. *Physics Reports*, 734:1–74, 2018.

[57] Kai Zhao, Mirco Musolesi, Pan Hui, Weixiong Rao, and Sasu Tarkoma. Explaining the power-law distribution of human mobility through transportation modality decomposition. *Scientific reports*, 5:9136, 2015.

[58] D.R. Cox and O.E. Barndorff-Nielsen. *Inference and Asymptotics*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1994.

[59] L. Wasserman and L.A. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer, 2004.

[60] Quang H Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica: Journal of the Econometric Society*, pages 307–333, 1989.

[61] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[62] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. Numerical recipes in c++. *The art of scientific computing*, 2:1002, 1992.

[63] G. Salton, A. Wong, and C. S. Yang. Vector Space Model for Automatic Indexing. Information Retrieval and Language Processing. *Communications of the ACM*, 18(11):613–620, 1975.

[64] George Adosoglou, Gianfranco Lombardo, and Panos M Pardalos. Neural network embeddings on corporate annual filings for portfolio selection. *Expert Systems with Applications*, page 114053, 2020.

[65] Tim Laughran and Bill Mcdonald. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65, 2011.

[66] Travis Dyer, Mark Lang, and Lorien Stice-Lawrence. The evolution of 10-K textual disclosure: Evidence from Latent Dirichlet Allocation. *Journal of Accounting and Economics*, 64(2-3):221–245, 2017.

[67] Arman Khadjeh Nassirtoussi, Saeed Aghabozorgi, Teh Ying Wah, and David Chek Ling Ngo. Text mining of news-headlines for FOREX market prediction: A Multi-layer Dimension Reduction Algorithm with semantics and sentiment. *Expert Systems with Applications*, 42(1):306–324, 2015.

[68] Marjan Van De Kauter, Diane Breesch, and Véronique Hoste. Fine-grained analysis of explicit and implicit sentiment in financial news articles. *Expert Systems with Applications*, 42(11):4999–5010, 2015.

[69] Hakan Gunduz and Zehra Cataltepe. Borsa Istanbul (BIST) daily prediction using financial news and balanced feature selection. *Expert Systems with Applications*, 42(22):9001–9011, 2015.

[70] P. S.M. Nizer and J. C. Nievola. Predicting published news effect in the Brazilian stock market. *Expert Systems with Applications*, 39(12):10674–10680, 2012.

[71] Constantin Zopounidis, Michael Doumpos, and Panos M Pardalos. *Handbook of financial engineering*. Springer Science & Business Media, 2010.

[72] Shanshan Wang, Kaiquan Xu, Long Liu, Bing Fang, Shaoyi Liao, and Huaiqing Wang. An ontology based framework for mining dependence relationships between news and financial instruments. *Expert Systems with Applications*, 38(10):12044–12050, 2011.

[73] Eduardo Lupiani-Ruiz, Ignacio García-Manotas, Rafael Valencia-García, Francisco García-Sánchez, Dagoberto Castellanos-Nieves, Jesualdo Tomás Fernández-Breis, and Juan Bosco Camón-Herrero. Financial news semantic search engine. *Expert Systems with Applications*, 38(12):15565–15572, 2011.

[74] Min Yuh Day and Chia Chou Lee. Deep learning for financial sentiment analysis on finance news providers. *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016*, (1):1127–1134, 2016.

[75] Paul C. Tetlock. Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62(3):1139–1168, 2007.

[76] Timotej Jagric, Stefan Bojnec, and Vita Jagric. Optimized spiral spherical self-organizing map approach to sector analysis–the case of banking. *Expert Systems with Applications*, 42(13):5531–5540, 2015.

[77] Yangtuo Peng and Hui Jiang. Leverage financial news to predict stock price movements using word embeddings and deep neural networks. *2016 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference, pages 374–379, 2016.

[78] Sahar Sohangir, Dingding Wang, Anna Pomeranets, and Taghi M. Khoshgoftaar. Big Data: Deep Learning for financial sentiment analysis. *Journal of Big Data*, 5(1), 2018.

[79] Paola Cerchiello, Giancarlo Nicola, Samuel Ronnqvist, and Peter Sarlin. Deep learning bank distress from news and numerical financial data. *SSRN Electronic Journal*, 06 2017.

[80] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Doc2Vec. *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*, 32:29–30, 2015.

[81] M. Doumpos, C. Zopounidis, and P.M. Pardalos. *Financial Decision Making Using Computational Intelligence*. Springer US, 2012.

[82] Dimitris Andriosopoulos, Michalis Doumpos, Panos M. Pardalos, and Constantin Zopounidis. Computational approaches and data analytics in financial services: A literature review. *Journal of the Operational Research Society*, 70(10):1581–1599, 2019.

[83] Stephen V. Brown and Jennifer Wu Tucker. Large-Sample Evidence on Firms' Year-over-Year MD&A Modifications. *Journal of Accounting Research*, 49(2):309–346, 2011.

[84] Feng Li. Do Stock Market Investors Understand the Risk Sentiment of Corporate Annual Reports? *SSRN Electronic Journal*, 2011.

[85] Oriana Bandiera, Andrea Prat, Stephen Hansen, and Raffaella Sadun. Ceo behavior and firm performance. *Journal of Political Economy*, 128(4):1325–1369, 2020.

[86] Vipula Rawte, Aparna Gupta, and Mohammed J. Zaki. Analysis of year-over-year changes in risk factors disclosure in 10-K filings. *Proceedings of the 4th International Workshop on Data Science for Macro-Modeling, DSMM 2018 - In conjunction with the ACM SIGMOD/PODS Conference*, pages 2–5, 2018.

[87] Feng Mai, Shaonan Tian, Chihoon Lee, and Ling Ma. Deep learning models for bankruptcy prediction using textual disclosures. *European Journal of Operational Research*, 274(2):743–758, 2019.

[88] Ming Feng Tsai, Chuan Ju Wang, and Po Chuan Chien. Discovering finance keywords via continuous-space language models. *ACM Transactions on Management Information Systems*, 7(3), 2016.

[89] [dataset] Bill McDonald. Stage one 10-x parse files, 2019. data retrieved from World Development Indicators, `https://sraf.nd.edu/data/`.

[90] Eugene F. Fama and Kenneth R. French. Dissecting anomalies. *Journal of Finance*, 63(4):1653–1678, 2008.

[91] Jey Han Lau and Timothy Baldwin. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. pages 78–86, 2016.

[92] Eugene F. Fama and Kenneth R. French. Multifactor explanations of asset pricing anomalies. *The Journal of Finance*, 51(1):55–84, 1996.

[93] Gianfranco Lombardo and Poggi Agostino. Actornode2vec: An actor-based solution for node embedding over large networks. *Intelligenza Artificiale, vol. 14, no. 1*, pages 103–114, 2020.

[94] Gianfranco Lombardo and Agostino Poggi. A scalable and distributed actor-based version of the node2vec algorithm.

[95] Enrico Franchi, Agostino Poggi, and Michele Tomaiuolo. Blogracy: A peer-to-peer social network. In *Censorship, Surveillance, and Privacy: Concepts, Methodologies, Tools, and Applications*, pages 675–696. IGI Global, 2019.

[96] Giulio Angiani, Paolo Fornacciari, Gianfranco Lombardo, Agostino Poggi, and Michele Tomaiuolo. Actors based agent modelling and simulation. In *Highlights of Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*, pages 443–455, Cham, 2018. Springer International Publishing.

[97] Paolo Fornacciari, Monica Mordonini, Agostino Poggi, Laura Sani, and Michele Tomaiuolo. A holistic system for troll detection on twitter. *Computers in Human Behavior*, 89:258–268, 2018.

[98] Gianfranco Lombardo, Paolo Fornacciari, Monica Mordonini, Michele Tomaiuolo, and Agostino Poggi. A multi-agent architecture for data analysis. *Future Internet*, 11(2):49, 2019.

[99] Michele Tomaiuolo, Gianfranco Lombardo, Monica Mordonini, Stefano Cagnoni, and Agostino Poggi. A survey on troll detection. *Future Internet*, 12(2):31, 2020.

[100] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, USA, 1986.

[101] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *CEAS*, 2004.

[102] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[103] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976. International World Wide Web Conferences Steering Committee, 2018.

[104] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.

[105] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. Dynamic network embedding: An extended approach for skip-gram based network embedding. In *IJCAI*, pages 2086–2092, 2018.

[106] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3762–3765. IEEE, 2018.

[107] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, page 104816, 2019.

[108] Yu Han, Jie Tang, and Qian Chen. Network embedding under partial monitoring for evolving networks. In *IJCAI*, pages 2463–2469, 2019.

[109] Uriel Singer, Ido Guy, and Kira Radinsky. Node embedding over temporal graphs. *arXiv preprint arXiv:1903.08889*, 2019.

[110] John R Hurley and Raymond B Cattell. The procrustes program: Producing direct rotation to test a hypothesized factor structure. *Behavioral science*, 7(2):258–262, 1962.

[111] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *nature*, 453(7196):779, 2008.

[112] Reka Albert. Scale-free networks in cell biology. *Journal of cell science*, 118(21):4947–4957, 2005.

[113] Vladimir Boginski, Sergiy Butenko, and Panos M Pardalos. Statistical analysis of financial networks. *Computational statistics & data analysis*, 48(2):431–443, 2005.

[114] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, 2016.

[115] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5):60–69, 2003.

[116] Rosie Dunford, Quanrong Su, and Ekraj Tamang. The pareto principle. 2014.

[117] IL—MARDIA DRYDEN. Kv: Statistical shape analysis. *John Willey, New York*, 1999.

[118] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 817–826. ACM, 2009.

[119] Lei Tang and Huan Liu. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1107–1116. ACM, 2009.

[120] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[121] Sharad Nandanwar and M Narasimha Murty. Structural neighborhood based classification of nodes in a network. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1085–1094. ACM, 2016.

[122] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.