Graphs,Words, and Communities : converging paths to interpretability with a frugal embedding framework

Thibault Prouteau

COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Le Mans Université

THÈSE DE DOCTORAT DE

De
LE MANS UNIVERSITÉ
Sous le sceau de
LA COMUE ANGERS-LE MANS

ÉCOLE DOCTORALE N° 641
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Thibault PROUTEAU

Graphs, Words, and Communities:
Converging Paths to Interpretability with a Frugal Embedding Framework

Thèse présentée et soutenue au Mans, le 3 juillet 2024
Unité de recherche : Laboratoire d'Informatique de l'Université du Mans (LIUM)
Thèse N° : 2024LEMA1006

Rapporteurs avant soutenance :

Vincent LABATUT	Maître de Conférences HDR	Université d'Avignon, Avignon
Christine LARGERON	Professeur des Universités	Université Jean Monnet, Saint-Étienne

Composition du Jury :

Président :	Jean-Loup GUILLAUME	Professeur des Universités	Université de la Rochelle, La Rochelle
Examinateurs :	Cécile BOTHOREL	Professeur des Universités	IMT Atlantique, Brest
	Anaïs HALFTERMEYER	Maître de Conférences	Université d'Orléans, Orléans
	Vincent LABATUT	Maître de Conférences HDR	Université d'Avignon, Avignon
	Christine LARGERON	Professeur des Universités	Université Jean Monnet, Saint-Étienne
	Marie TAHON	Professeur des Universités	Université du Mans, Le Mans
Dir. de thèse :	Sylvain MEIGNIER	Professeur des Universités	Université du Mans, Le Mans
Encadrant :	Nicolas DUGUÉ	Maître de Conférences	Université du Mans, Le Mans

Invité :

Nathalie CAMELIN	Maître de Conférences	Université du Mans, Le Mans

Graphs, Words, and Communities: Converging Paths to Interpretability with a Frugal Embedding Framework

Thibault Prouteau

UNIVERSITÉ DU MANS

IMPRINT

Graphs, Words, and Communities:
Converging Paths to Interpretability
with a Frugal Embedding Framework

COLOPHON

This thesis was typeset using LaTeX and the `memoir` documentclass. It is based on Aaron Turon's thesis *Understanding and expressing scalable concurrency*[1], itself a mixture of `classicthesis`[2] by André Miede and `tufte-latex`[3], based on Edward Tufte's *Beautiful Evidence*.

The bibliography was processed by Biblatex. Most schemas and plots are made with PGF/Ti*k*Z.

The body text is set 10/14pt (long primer) on a 26pc measure. The margin text is set 8/9pt (brevier) on a 12pc measure. Matthew Carter's Charter acts as both the text and display typeface. Monospaced text uses Jim Lyles's `Bitstream Vera Mono` ("Bera Mono").

[1] https://people.mpi-sws.org/~turon/turon-thesis.pdf

[2] https://bitbucket.org/amiede/classicthesis/

[3] https://github.com/Tufte-LaTeX/tufte-latex

Acknowledgments

Here we are, almost ten years after starting my academic journey, I get to look back on everything I have accomplished and thank the people who helped me along the way.

I spent four years at LIUM, researching, teaching, learning. Teachers became supervisors and colleagues. I would like to thank Sylvain, my doctoral advisor, for his support, his long-term vision, and his wisdom in all things academia that were precious throughout this journey. When I started doing research at LIUM, two people were instrumental in my wish to pursue an academic career: Nathalie and Nicolas. We will start with Nathalie with whom I had the pleasure of teaching these last four years. Thank you for your inexhaustible joyfulness and passion for teaching. A person without whom this thesis would not have been possible is Nicolas. We started collaborating in 2019. Throughout this past half-decade, I had the pleasure of seeing become a real PI, overseeing the work of many young researchers with great benevolence. Thank you for your support and trust. We were soon joined by Simon, a brilliant linguist, and Anna, who brought a new look to our work. Thank you for your insights and your support!

Colleagues from the lab have also been of tremendous support throughout this work. We spent numerous hours sharing our academia struggles and socializing at BSD. Special thanks to Valentin, Martin, Théo, Thibault, Félix, Valentine and Albane for their friendship. LIUM is a lot of people with a passion for research and teaching. A special thanks to Jane, Marie, Meysam, Emmanuelle and Aghilas for their continuous encouragement. Administrative support was also key in this thesis with Étienne and Grégor for many things including (but not restricted to) the computational infrastructure. Thank you Anne-Cécile for your administrative support and making our life as PhD students easier when it comes to admin. Thank you, Clémence and more broadly the Doctoral administration for their work.

Farther from Le Mans, people have also contributed to this thesis. First, people from Blois, Maria, Gaëtan and Wissam. Thank you for your friendship these last few years. Some professors from Blois were also instrumental in me pursuing a PhD. Thank you Jean-Yves for sharing your passion of research which helped me realize what research could be. There are also research buddies, met in Santa Fe at CSSS, Meri, Sophia, and Lena. It's always a pleasure to catch up about which conferences you are attending and what you are up to. I also want to acknowledge the continuous and fundamental support of Rob and Manuel, even though you are far away you always manage to make me laugh!

Finally, family has been the greatest support throughout these last ten years of university. First my grandmas, although geographically far, they were always supportive and curious about my work. Thank you, mom and dad, for your financial and moral support. Thank you, Chloé and Oriane for making me a proud brother!

Thank you everyone!

Près de dix ans après le début de mon parcours universitaire, j'ai l'occasion de revenir sur tout ce que j'ai accompli et de remercier les personnes qui m'ont aidé tout au long de mon parcours.

J'ai passé quatre ans au LIUM, à chercher, à enseigner, à apprendre. Les enseignants sont devenus des encadrants et des collègues. J'aimerais remercier Sylvain, mon directeur de thèse, pour son soutien, sa vision à long terme et sa sagesse dans tous les domaines universitaires, qui m'ont été précieux tout au long de ce parcours. Lorsque j'ai commencé à faire de la recherche au LIUM, deux personnes ont joué un rôle déterminant dans mon désir de poursuivre une carrière universitaire : Nathalie et Nicolas. Commençons par Nathalie avec qui j'ai eu le plaisir d'enseigner ces quatre dernières années. Merci pour ton inépuisable joie de vivre et ta passion pour l'enseignement. Une personne sans laquelle cette thèse n'aurait pas été possible est Nicolas. Nous avons commencé à collaborer en 2019. Tout au long de cette demi-décennie, j'ai eu le plaisir de le voir devenir un véritable PI, encadrant avec beaucoup de bienveillance le travail de nombreux jeunes chercheurs. Merci ton soutien et ta confiance. Nous avons rapidement été rejoints par Simon, un brillant linguiste, et Anna, qui ont apporté un nouveau regard sur notre travail. Merci pour votre regard nouveau et votre soutien !

Les collègues du laboratoire ont également été d'un grand soutien tout au long de ce travail. Nous avons passé de nombreuses heures à partager nos difficultés et à boire des coups au BSD. Je remercie tout particulièrement Valentin, Martin, Théo, Thibault, Félix, Valentine et Albane pour leur amitié. Le LIUM grand groupe de passionnés de recherche et d'enseignement. Je souhaite remercier tout particulièrement Jane, Marie, Meysam, Emmanuelle et Aghilas pour leurs encouragements constants. Le soutien administratif a également été essentiel dans cette thèse avec Étienne et Grégor pour de nombreuses choses, y compris (mais pas seulement) l'infrastructure informatique. Merci à Anne-Cécile pour son soutien administratif et pour avoir rendu notre vie de doctorants plus facile concernant l'administratif. Merci à Clémence et plus largement au pôle doctoral pour leur travail.

Plus loin du Mans, des personnes ont également contribué à cette thèse. Tout d'abord, des personnes de Blois, Maria, Gaëtan et Wissam. Merci pour votre amitié durant ces dernières années. Certains professeurs de Blois ont également joué un rôle déterminant dans la poursuite de mon doctorat. Merci à Jean-Yves de m'avoir fait partager sa passion pour la recherche, ce qui m'a permis de prendre conscience de ce que pouvait être le métier d'enseignant-chercheur. Il y a aussi des camarades de recherche, rencontrés à Santa Fe a CSSS: Meri, Sophia, Lena. C'est toujours un plaisir de savoir à quelles conférences vous participez et ce qui occupe vos recherches. Je tiens également à souligner le soutien constant et fondamental de Rob et Manuel, même si vous êtes loin, vous arrivez toujours à me faire rire !

Enfin, la famille a été mon plus grand soutien tout au long de ces dix dernières années d'université. Tout d'abord, mes grands-mères, bien que géographiquement éloignées, m'ont toujours soutenue et se sont montrées curieuses de mon travail. Merci, papa et maman, pour votre soutien financier et moral. Merci à Chloé et Oriane de faire de moi un frère fier !

Contents

List of Figures

List of Tables

List of Definitions

Introduction

MOTIVATIONS

In all aspects of our daily lives, we interact with other individuals in our social circles, we use commodities such as public transport, the power grid to heat our homes and cook, social networks for leisure and information. They fall under what we call complex networks. Complex networks are structured complex systems in which items are interconnected if they entertain relations. These relations can, for example, be social between individuals, exchanges of information and messages in mail networks or in the form of energy between electrical power stations. Gaining insight into the topology of these complex networks can help us understand how they are structured, evolve and do not collapse. Furthermore, it might help us better understand how humans interact, communicate and collaborate. However, complex networks typically contain many interconnected elements, making it difficult to analyze their structure without algorithmic means.

Artificial Intelligence (AI), and particularly Machine Learning (ML) provide algorithmic solutions to exploiting large quantities of data. However, algorithms work with mathematical representations, and data is often not suited to be exploited as is. Representation learning solves this issue by providing summarized mathematical representations of data that can be exploited by machine learning algorithms. Representation learning spans many areas of science and can be used to extract representations of data for modalities as diverse as speech, image, text, temporal series. Usually, these vector representations of data are called embeddings. Embedding comes from the objective of learning the representation by compressing information in a latent space that conveys the similarity between items in the data source. In complex networks, graph embeddings are used to represent networks' nodes and edges. In Natural Language Processing (NLP) representations of words from texts are word embeddings.

How does an algorithm extract a representation of data? Algorithms rely on statistical methods to learn representations of data. They capture what items appear together, extract patterns from the data. Representation learning and, more generally, currently, most fields in AI rely heavily on neural networks. By ingesting large amounts of data, they can capture regularities and patterns of a dataset to solve tasks, like classifying texts, recommending content or goods, *etc.* Recently, chatbots have put NLP in the spotlight for the public. Users can interact with an AI that seems omniscient. However, chatbots like OpenAI's *ChatGPT* were only trained on massive text datasets to predict the most probable answer to a sequence of words. The underlying algorithm of these heavily publicized systems is a language model, learning a representation of language.

The first large architectures were *Pre-trained Language Models* (PLMS) whose size and training objectives evolved into today's *Large Language Models* (LLM). These architectures require large quantities of training data. At

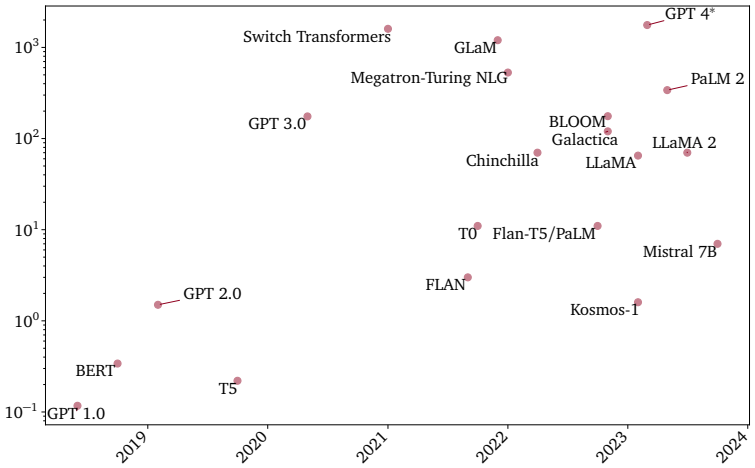


FIGURE 1: Sizes of some LLM (billions of parameters) according to their release dates. When multiple models are made available, only the largest is plotted. The list is not exhaustive and for illustration purposes. *Alleged size of GPT-4 not confirmed by OpenAI.

their beginnings, these architectures only amounted to around a hundred million trainable parameters. Since then, model sizes have grown, from GPT-3's 175 billion parameters in 2020 to models in the tens to hundreds of billion parameters nowadays (see Figure 1) [Bro+20; Ani+23].

[Bro+20] Brown et al., "Language Models are Few-Shot Learners"

[Ani+23] Anil et al., *PaLM 2 Technical Report*

This trend towards large neural architectures is observed in other areas of machine learning, including Complex Networks with the emergence of *Graph Neural Networks* (GNN) [Sca+09]. These large architectures have opened up new opportunities to involve AI in our everyday life: credit scores, assisted X-ray interpretations, automatic processing of CVs [Mos21; Bou+23].

[Sca+09] Scarselli et al., "The Graph Neural Network Model"

[Mos21] Moses, "Deep learning applied to automatic disease detection using chest X-rays"

[Bou+23] Bouhoun et al., "Information Retrieval Using Domain Adapted Language Models: Application to Resume Documents for HR Recruitment Assistance"

Despite the benefits brought by these new architectures, able to learn representations on large amounts of data, some concerns subsist. First and foremost, neural architectures are trained on large hardware systems using graphical processing units (GPU). The environmental impact of training and serving these large models to end users is a first challenge that begs the question of the long-term sustainability of these approaches. Their training consumes a lot of energy. In 2022, it was estimated that the cost of training the 176B parameters of the *BLOOM* LLM consumed 433,196 kWh of electricity, amounting to over 1B GPU hours for a training time of over 118 days. The total carbon cost is estimated to be around 24.7 tonnes of CO_2eq (however, the energy mix was mostly nuclear) [LVL23]. For reference, in [SGM19], it is estimated that the average yearly CO_2eq production of an American is around 16.4 tonnes. This measure is only considering training. Yet, the bulk of the energy consumption of large neural architectures is associated with inference, in production, when data is fed through the network to obtain an output. Inference in neural architectures like LLM comes with an increased power consumption: a standard *Google search* request consumes around 0.3 Wh of electricity, when *ChatGPT* consumes around 3 Wh per request, it represents a 1000% increase [De 23]. In such a scenario, with similar architectures, and with current hardware, the total consumption of *Google* AI operations could become equivalent to the yearly electricity con-

[LVL23] Luccioni et al., "Estimating the carbon footprint of bloom, a 176b parameter language model"

[SGM19] Strubell et al., "Energy and Policy Considerations for Deep Learning in NLP"

[De 23] De Vries, "The growing energy footprint of artificial intelligence"

sumption of Ireland (29.3 TWh per year)[De 23]. As a consequence, the long-term sustainability of AI architectures is a concern.

The second shortcoming of modern AI systems is their *"black-box"* nature. A *"black-box"* model has inputs and outputs but does not provide any information about its internal mechanics, which could help understand how an output came to be. With algorithmic processing being increasingly used as means of analyzing large quantities of data to help decision-making in high-stake areas (e.g., medicine [Raj+22], justice [DF18]), the lack of transparency raises concerns. These concerns are related to algorithms being biased or inaccurate depending on the input provided. Examples of biased systems have been publicized regarding recidivism prediction, scoring the risk of recidivism of individuals with a criminal record [Har15]. *COMPAS*, a recidivism scoring system, has been widely criticized for being biased based on race and gender. In a 2016 *ProPublica* investigative report, journalists highlighted bias in the algorithm based on race, where black individuals were falsely flagged as high-risk at double the rate of white individuals. On the other hand, white individuals were more often labeled as low-risk than black individuals. Scores were inaccurate, such that it is estimated that 20% of defendants for which violent crimes were predicted actually committed them [Ang+16]. Biases can also be found in systems orienting health policies, negatively impacting parts of the population because they rely on cost rather than illness [Obe+19]. *Amazon* was also under the spotlight in 2018 for an automatic resume processing algorithm. It extracted keywords from resumes and ranked candidates. However, it was biased negatively toward women and was later scrapped by *Amazon*.

[Raj+22] Rajpurkar et al., "AI in health and medicine"

[DF18] Dressel and Farid, "The accuracy, fairness, and limits of predicting recidivism"

[Har15] Harcourt, "Risk as a Proxy for Race: The Dangers of Risk Assessment"

[Ang+16] Angwin et al., "Machine Bias"

[Obe+19] Obermeyer et al., "Dissecting racial bias in an algorithm used to manage the health of populations"

In all these instances, algorithms were biased. Whether the biases come from training data or the design of the system, it would have been beneficial to audit the system and interpret its internal mechanics to identify the origin of biases [Név+22]. That is where explainable and interpretable models are of interest, providing more ethical algorithms. In the case of representation learning, everything is based on data, extracting its patterns and structure. Subsequently, algorithms can, and do extract biased representations of the world conveyed by data. Yet, embeddings are frequently the first brick of AI systems, and if this first brick is biased, then the whole system might be. Biases stemming from representations may even be amplified in the following bricks of the pipeline [Zha+17]. Being able to audit the internal organization of an embedding model and understand its structure is beneficial in high-stake applications of AI.

[Név+22] Névéol et al., "French CrowS-Pairs: Extending a challenge dataset for measuring social bias in masked language models to a language other than English"

[Zha+17] Zhao et al., "Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints"

This work is motivated by the two problems we put forward. The first is the need for more sustainable systems, especially to learn graph and word embeddings. With more efficient algorithms, representations could be trained fast and without requiring large hardware architectures that consume large amounts of energy, and are expensive. The second issue faced by AI systems, and in our particular context of learning representations of data, is the lack of interpretability of models provided. Interpretability is a desirable feature when it comes to representations, to audit models and uncover biases in high-stakes contexts, but also to understand the structuration of information in such models.

The core of our work is focused on providing more sustainable represen-

tations of large networks and text corpora with frugal algorithms that can provide interpretable embeddings.

CONTRIBUTIONS

To fulfill our goal of interpretability in providing graph and word embeddings, we introduce a new graph-based theoretical framework. Our *Lower Dimension Bipartite Graph Framework* (`LDBGF`) can theoretically provide a compressed representation of a network without loss of information. However, `LDBGF` is not solvable in a reasonable time, and thus we introduce *Sparse Interpretable Node representations* (`SINr-NR`, `SINr-MF`), two implementations within the network that rely on community detection, clustering on networks that can be performed efficiently, to derive node representations frugally. We leverage the relation of nodes to this community structure to extract representations. Our approach is hospitable to all types of networks that have an underlying community structure, including large text corpora that can be represented as word co-occurrence networks. We demonstrate that the versatility of our model enables it to perform well on classical benchmarks with lower runtimes than most of its counterparts in graph embedding and word embedding. Later, we demonstrate the interpretability of our approach on the word embedding task that is enabled by the quality of our representations, their sparsity, and robustness.

THESIS OUTLINE

This memoir is organized in three parts.

The first part is dedicated to introducing and defining complex systems and motivating the need for representation learning to grasp their complexity. This first part is articulated in two chapters. Chapter 1 gives a guided tour of complex systems, of their main characteristics, before focusing on complex networks. Given the main characteristics of complex networks, we introduce phenomenons that emerge from the complex topology of networks, namely small world, growth and preferential attachment, and community structures, that are of particular interest in this thesis.

In Chapter 2 we motivate the need for embedding models to represent the complex topology of networks. We demonstrate that a kinship exists both philosophically and implementation-wise between graph embedding and word embedding. Philosophically because embedding methods are distributional and rely on the distributional hypothesis. In terms of implementation, graph and word embedding share common algorithmic approaches that can be classified in broad categories of methods. We then introduce the specifics of representation learning for language and complex networks.

Yet, these methods have shortcomings related to their complexity and lack of interpretability that motivate our work. In this context, we introduce our interpretable embedding framework *Lower Dimension Bipartite Graph Framework* (`LDBGF`). We illustrate that it is adapted to build interpretable and visualizable representations of a domestic flight network over the United States of America. Finally, we introduce two implementations within this framework that rely on communities to alleviate the complex-

ity associated with LDBGF. SINr-NR is based on an ad hoc measure of the connectivity between nodes and communities to extract representation. In SINr-MF, we rely on gradient descent to find the transition matrix between the graph adjacency matrix and the community membership matrix.

The second part of this memoir is dedicated to benchmarking the methods we introduced Chapter 2.

Chapter 3 focuses on graph embedding. We review how networks can be modeled before quantitatively evaluating our approach. We start by measuring the runtime of SINr-NR and SINr-MF relatively to other graph embedding frameworks and show that SINr-NR has a shorter runtime than most other popular approaches. We perform a *link prediction* evaluation that measures the ability of SINr-NR and SINr-MF to provide informative representation to predict missing links from a network with a classifier. Afterward, we measure the extent of information embedded in representations by trying to predict graph features: *degree centrality*, *clustering coefficient*, and *PageRank score*. This chapter concludes by measuring the extent of community information embedded in vectors by trying to reconstruct communities from embeddings in both a supervised and unsupervised setting.

Chapter 4 confronts SINr-NR to the word embedding task via word co-occurrence networks. We present the specifics of building co-occurrence networks to learn network-based word embeddings. We then introduce classical evaluation tasks to assess the quality of word embeddings: *pairwise word similarity* and *word categorization*. We show that SINr-NR is a good contender to produce word embeddings. They convey an accurate representation of language according to benchmarks, while maintaining runtime low.

Chapter 5, we dive into the interpretability of word embeddings. Interpretable models internal structure is supposed to make sense for humans. After exposing the main differences between explainability and interpretability, we motivate why they are of critical interest for high-stake applications of machine learning. Explainability and interpretability are gradually being incorporated into public policies. We focus on interpretability of vector spaces and lay the main requirements for an interpretable embedding space. With a human evaluation, we show that SINr-NR's dimensions are interpretable. We then investigate the stability of our model that guarantees the accuracy of interpretations drawn from a model. We investigate the contribution of sparsity to the interpretability of representations, and particularly to interpreting how sense is composed in an embedding vector. This allows defining a new kind of interpretability: vector-level interpretability. Finally, through visualizations, we explore the structure of our representations and visualize how sense is built through a combination of thematics.

In the last part of this manuscript, Chapter 6, we look back at the main contributions of this thesis and introduce the perspectives that stem from this work. The perspectives are multifold. The first one is related to experimenting with other approaches to sparsification to improve performances of models. The second one would aim at evaluating dimension-interpretability automatically and vector-level interpretability with a human evaluation. We finally discuss the possibility of integrating a time component in the models

to provide temporal graph and word embeddings that would allow studying the evolution of a network or language through time.

PUBLICATIONS

The research conducted during this PhD as well as the team work within the ANR-21-CE23-0010 *Dynamic and Interpretable Graph-based word embeddINGs* (DIGING) project resulted in the following publications:

- JOURNALS
  - **T. Prouteau**, N. Dugué, and S. Guillot. From Communities to Interpretable Network and Word Embedding: a Unified Approach. In: *Journal of Complex Networks (in press)*. (2024).
  - A. Béranger, N. Dugué, S. Guillot, and **T. Prouteau**. SINr-filtered: filtering communities to improve interpretability and performances. ***Submitted to PLOS Complex Systems***. (2024).

- INTERNATIONAL SYMPOSIUMS
  - A. Béranger, N. Dugué, S. Guillot, and **T. Prouteau**. Filtering communities in word co-occurrence networks to foster the emergence of meaning. In: *Conference on Complex Networks*. (2023).
  - S. Guillot, **T. Prouteau**, and N. Dugué. Sparser is better: one step closer to word embedding interpretability. In: *International Workshop on Computational Semantics* (IWCS). (2023).
  - **T. Prouteau**, N. Dugué, N. Camelin, and S. Meignier. Are Embedding Spaces Interpretable? Results of an Intrusion Detection Evaluation on a Large French Corpus. In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference* (LREC). (2022).
  - **T. Prouteau**, V. Connes, N. Dugué, A. Perez, J.-C. Lamirel, N. Camelin, and S. Meignier. SINr: Fast Computing of Sparse Interpretable Node Representations is not a Sin! In: *Advances in Intelligent Data Analysis XIX* (IDA). (2021)

- NATIONAL SYMPOSIUMS
  - A. Béranger, N. Dugué, S. Guillot, and **T. Prouteau**. SINr-filtered : filtrer les communautés pour mieux découvrir le sens. In: *Conférence Extraction et Gestion des Connaissances* (EGC). (2024).
  - **T. Prouteau**, N. Dugué, S. Guillot and A. Perez. SINr: a python package to train interpretable word and graph embeddings. In: *French Regional Conference on Complex Systems* (FRCCS). (2023).
  - S. Guillot, **T. Prouteau**, and N. Dugué. De l'interprétabilité des dimensions à l'interprétabilité du vecteur: parcimonie et stabilité. In: *Conférence sur le Traitement Automatique des Langues Naturelles* (TALN). (2023).

Additional publications outside the scope of this thesis

- V. Pelloin and **T. Prouteau**. Apprentissage de plongements de mots sur des corpus en langue de spécialité : Une étude d'impact. In: *RÉCITAL, 22e édition*. (2020).
- C. Reutaner, L. Lefeuvre, A. Fouqueray, **T. Prouteau**, V. Pelloin, C. Lopez, N. Camelin, F. Segond, N. Dugué, and D. Bourigault. Technologies sémantiques et accès à linformation dans le prescrit SNCF. In: *Congrès Lambda Mu 22*. (2020).

Part I

COMPLEX SYSTEMS AND THEIR REPRESENTATIONS

1

Complex systems and networks

Chapter contents

Synopsis

Complex systems exist in many parts of our daily lives, we, ourselves are complex biological systems composed of cells, tissues, proteins, evolving in social complex systems. When we commute to work using public transportation or have an online meeting through the internet, we are also interacting with complex systems. Their ubiquity in our daily lives makes complex systems an inexhaustible source of explorations to better understand how they form, grow and function. Due to their variety, systems are studied through the lens of many domains such as physics, ecology, neuroscience, or in our case computer science. We will focus on the subtype of complex networks in which items interact to form a greater network structure. This chapter is dedicated to introducing complex systems, methods to analyze them and define their main characteristics. In the second part of this chapter, we will focus our attention on complex networks, define their broad characteristics and how they reflect the properties of complex systems.

Résumé

Les systèmes complexes sont présents dans beaucoup d'aspects de nos vies quotidiennes. Nous sommes nous-mêmes des systèmes biologiques complexes composés de cellules, tissus et protéines. Nous évoluons également dans des systèmes sociaux. Lorsque nous utilisons les transports en commun, ou lorsque nous participons à une réunion en visioconférence, nous interagissons également avec et au travers de systèmes complexes. Leur omniprésence dans notre quotidien en fait une source inépuisable pour explorer et mieux comprendre comment ils se forment, se développent et fonctionnent. Par leur multiplicité, ces systèmes sont étudiés du point de vue et avec les méthodes de divers domaines tels que la physique, l'écologie, les neurosciences ou ici l'informatique. En ce qui concerne les travaux présentés dans ce manuscrit, nous allons nous concentrer sur le sous-domaine et les structures de réseaux complexes dans lesquels interagissent des objets pour donner une organisation à plus grande échelle. Ce chapitre a pour objectifs de faire un tour d'horizon des systèmes complexes et des diverses méthodes d'étude de ces derniers. Nous présentons alors leurs principales caractéristiques avant de porter notre attention sur les réseaux complexes. Nous définirons leurs principales caractéristiques et comment ils reflètent celles énoncées pour les systèmes complexes.

1.1 COMPLEX SYSTEMS, A COMPLEX DEFINITION

Complexity, the quality of being intricate, complicated. We, as humans, are complex systems, or to be more precise, a collection of complex systems that shape our form and behavior. Our body, for instance, has of many interacting complex systems, it is composed of organs that are part of systems (*e.g.*, circulatory, urinary) that can work together so that we may proceed with our life. Stephen J. Gould, American paleontologist, evolutionary biologist and historian of sciences, stated the following in *The Panda's Thumb: More Reflections in Natural History*:

> "Organisms are not billiard balls, propelled by simple and measurable external forces to predictable new positions on life's pool table. Sufficiently complex systems have greater richness." Stephen J. Gould [Gou10]

Organisms are complex as the result of their evolution and its impact on their function. So are other complex systems we take part in on an everyday basis: our social circles, our cities, our language. Complexity science is at the crossroads of many domains that deal with complex systems: physics, ecology, geography, sociology, computer science, etc. But what exactly are complex systems? And, would it be possible to give a universal definition?

Mark Newman, a notorious physicist that extensively contributed to complex networks states that:

> "complex systems theory is not a monolithic body of knowledge" Mark Newman [New11]

The domain of complex systems is vast and involves a diversity of objects. Mark Newman is external faculty at the Santa Fe Institute (SFI), a multidisciplinary institute focusing on complex systems and happen to use an analogy by Doyne Farmer, also from SFI to describe complex systems theory:

> "[...] complex systems theory is not a novel, but a series of short stories. Whether it will one day become integrated to form a single coherent theory is a matter of current debate, although my belief is that it will not." Mark Newman, borrowed from Doyne Farmer [New11]

Therefore, it seems unlikely to find a definition that fits all the aspects of complex systems. However, we can outline the characteristics that together give rise to a complex system. We can then overview the areas and techniques used by complex systems scientists to model and better understand complexity. Finally, we can define what complexity means in the context of this work, what it entails, and how it motivates our quest of interpretability to better understand how word sense is formed.

1.1.1 *Characteristics of complex systems*

Complexity is frequently said to appear at the ridge between regularity and randomness. The epistemology of complex systems cites many characteristics that have an impact on complexity [Gel95; New11; LLW13; Est23]. However, depending on the domain, characteristics of complex systems seem to vary. The definition for a physicist might be different from that of a computer scientist or an economist. Therefore, I will focus on five aspects of complex systems that are necessary when dealing with complex networks and more specifically language oriented networks.

▸ MANY INTERACTING ELEMENTS

A complex system starts with elements: proteins in cells, bees in a swarm or people in a social network. In order for these systems to exchange information and interact, they need to be similar in some way, speak the same language. Cells need to have similar receptors to communicate chemically. People in social systems need to belong to the same circles, have similar behaviors or obey to the same interaction rules to meet. Ladyman et al. [LLW13] use the example of weather to illustrate the need for similarity in complex systems. Climate is composed of molecules that are part of the atmosphere. These molecules interact with one another and are influenced by radiation from the sun, by geology, etc. Put together, these molecules interacting with their environment give rise to the weather, that itself is a system.

[LLW13] Ladyman et al. "What is a complex system?"

We will later introduce word co-occurrence networks (words that appear together in a sentence are said to co-occur and can be linked together in a network where they are represented by nodes), whose topology is rooted in the rules of language, and in turn how words interact in sentences. Each word in the language has a meaning or at least a function. The interaction of words in sentences and texts give rise to sense. We will also study other types of real-world networks, such as email exchanges, co-publication of articles, co-citation of publications or friendships on social networks that all form systems of many interacting elements.

▸ INTERACTIONS

Interactions is the second element to consider when attempting to define a complex system. Interaction may take the form of energy, collisions, or communication. Morin [Mor92] states that interactions allow "organization" in a complex system. Without interaction, it is just an ensemble of independent and unrelated elements. Thus, in non-interacting systems, if they can be called as such, no chance is left for items to organize into a complex topology. Interactions can be local: authors collaborating on a scientific publication, ants following pheromone trails left by other members of their colony. They can also be more distant: suboceanic fiber connections connecting servers on different continents. Interactions can be defined by simple rules, yet, their complexity lies in the fact that it is complicated to predict interactions of many elements. However, some tasks aim at predicting links, for example the *link prediction* for complex networks aims at predicting whether a link is likely to appear or exist in a network for a given

[Mor92] Morin "From the concept of system to the paradigm of complexity"

pair of nodes. In this particular case, topology of the network can be used in heuristics (based on node similarity, centrality in the network, etc.) to predict missing or future links. Predictions are not restricted to links, we may wish to predict the centrality of a node in a network, how many neighbors it has, etc. We will experiment with predicting links in networks and features of these networks Section 3.3.

▸ HIERARCHY

Hierarchy is an important feature of complex systems. By hierarchy, we mean that a complex system is composed of multiple subsystems that are in turn also the result of a combination of subsystems down to an elementary level. An example of hierarchy could be the universe. At the highest macroscopic level, super clusters of galaxies, followed by galaxies and solar systems that in turn contain planets. At the microscopic-scale, these planets are composed of molecules themselves composed of atoms which are in themselves composed of protons, neutrons, and electrons. Each and every level in this hierarchy is dependent on its previous level. Simon [Sim62] described in his seminal paper many hierarchical complex systems across various domains. In biological systems, cells combine to form tissues and organs that in turn constitute functional systems. Furthermore, H. Simon also points out that even though some systems might display subordination between hierarchical levels, it is not always the case. To be more precise, a higher level in the hierarchy is not necessarily the "boss" of its subsystems. This further complexifies interactions between systems.

[Sim62] Simon "The Architecture of Complexity"

Let us try to expose the hierarchy of language that will be one of the systems we model Chapters 2 and 4. Herbert uses the example of a book composed of chapters which are divided in sections, paragraphs, sentences, clauses, and phrases that are in turn composed of words. If we take this reflection further, a book might be part of a collection that constitutes a corpus. Corpus have different genres, which constitute another hierarchical level of linguistic resources of a language. If we take this reflection further, language can be described hierarchically in many ways. If we take words as elementary units of the language, they are organized together in sentences, paragraphs, and texts following the rules of grammar to provide and higher representation that is sense. Furthermore, words are part of an ontological system that organizes terms according to the concepts they relate to. We will dive more in depth Sections 1.2.4 and 2.3 into the hierarchical organization of complex networks and especially their underlying community structures.

▸ EMERGENCE

Emergence is another important characteristic of complex systems. As Aristotle [Ari24] famously conjectured: "The whole is something besides the parts." Combining the elements contributes to a greater purpose than a strict summation of their individual properties. The combination of these elements gives rise to emergent phenomena. Emergence is related to self-organization of a system, where organization arises from local interactions. Examples of self-organizing systems include bird flocks where birds fly in the same direction based on the behavior of their neighbors. In linguistics, multiple instances of emergence have been formulated. Hopper [Hop87]

[Ari24] Aristotle *Metaphysics*

[Hop87] Hopper "Emergent Grammar"

talks about "*Emergent Grammar*", suggesting that discourse self-defines its structure, shape, and regularity. Thus, language change is likely shaped by its use. If enough speakers use language in the same manner, then it is changed [Kel94]. Another phenomenon that emerges from text is *Zipf*'s law [Zip35]. The frequency of a word seems to be approximately inversely proportional to its rank (i.e. $1/rank$, vocabulary ordered from most to least frequent, see Figure 1.6). The *Zipf* distribution seems to hold regardless of the size of the text at hand. We cover Zipf's law and related distributions Section 1.2.3.

[Kel94] Keller, *On language change: The invisible hand in language*

[Zip35] Zipf, *The psycho-biology of language.*

Zipf's law and monkeys.
George Miller [Mil57] observed that the distribution of words typed randomly by monkeys on a keyboard also follows *Zipf*'s law.

▸ ROBUSTNESS AND LACK OF CENTRAL CONTROL

Robustness relates to the fact that a system does not collapse if one of its elements disappears or changes behavior. Since organization is distributed over the whole system and not liable to a unique central control, a small perturbation in a localized area of the system likely has little impact on its macroscopic structure. Let us go back to the bird flock example. A couple of birds having unusual movements do not jeopardize the movement of the whole. This robustness holds until perturbation is so strong that the topology of the system is changed significantly. It is beneficial to study the robustness of complex systems to analyze the resilience of banking systems, computing infrastructures, or ecosystems following the extinction of species.

1.1.2 *General theories and tools of complex systems*

We have thus far exposed some features displayed by complex systems. As complex systems span many domains, theories to model their phenomenons and dissect complexity are also diverse. Our work focuses mainly on complex networks and their topology. However, many frameworks are employed throughout the domain to understand the underlying dynamics and topology of complex systems. Mitchell [Mit09] offers a guided tour in complexity territory, providing keys to better grasp the richness that comes from intricacy.

[Mit09] Mitchell *Complexity: a guided tour*

These frameworks include non-linear dynamical systems which are governed and can be described with differential equations that capture their evolution based on initial conditions. They can exhibit chaotic behaviors, such that a small change in initial conditions has a considerable impact on the evolution of the system. Examples of systems involving non-linear dynamics include the weather, where small changes in temperature, humidity, or wind can have a considerable influence. A double pendulum is also a non-linear dynamical system which, depending on the initial setup and speed, can display chaotic motion. Population models such as the Logistic Map are another set of systems which can be described with non-linear dynamics. Chaotic behaviors make these systems difficult to predict and control over extended time periods.

Adaptation and game theory are other useful frameworks used across multiple communities. Many systems adapt through time and evolve based on interactions between the elements they include. Natural evolution through selection is a characteristic example of adaptation and emergent behavior.

Complex adaptive systems span many areas and often define a fitness function that measures how well an element, group, or strategy is doing in comparison with competition. This includes evolutionary theory, especially in biology; genetic algorithms in computer science that compete against each other through multiple generations to maximize a fitness function and solve difficult problems; game theory is another approach where "player" aims to choose the best strategy to maximize payoffs. Game theory can be especially useful to study foraging and mating strategies in species or human careers and financial decisions.

Information theory also informs on the complexity of a system. According to information theory, frequent patterns in complex systems have a low information content. Thus, being able to quantify information content and identify rare patterns can help put forward intricate organization in complex systems. This allows finding recurrent patterns in DNA, networks or the internet. Claude Shannon is one of the fathers of information theory and contributed to theorized how it can inform on the complexity of a system. We will later introduce measures relying on information content of text widely used in Natural Language Processing (NLP).

Computational complexity is an area of complex systems interested in the difficulty of computational tasks. Computational complexity is mainly considered in computer science with the objective of estimating space (memory) and time required to complete computation. Among classes of complexity, nondeterministic polynomial (NP) problems are a particular class of complexity where correctness of a solution can be checked rapidly but finding the solution itself takes significant time. The traveling salesman problem is a notorious NP-hard problem (problems of the NP class) where the objective is to find the shortest itinerary while visiting multiple cities. Verifying whether the itinerary found is shorter than a given distance or whether all the cities have been visited is rather easy but finding an optimal solution can prove difficult. Apart from purely computer-based considerations of complexity, Aaronson [Aar13] argues that computational complexity is also applicable in natural systems to quantify complexity, such as protein-folding, quantum computing and relativity.

[Aar13] Aaronson "Why Philosophers Should Care about Computational Complexity"

Agent-based modeling has the goal of simultaneously and autonomously simulating the behavior of agents in a complex system and their interactions. These interactions can then give rise to emergent phenomenons. Agent-based models (ABM) are useful across many domains such as epidemiology to study the spread of diseases, biology to model plant-animal interactions, language choice dynamics, urban planning via digital twins of cities or to simulate forest fires. ABM rely on a set of rules based on which agents interact in a virtual space. It is critical for an ABM to be as accurate as possible in mimicking the system it models so that conclusions are relevant.

Graph theory and complex networks are ubiquitous in the study of complex systems. Networks can map the topology of a system. Who is interacting with whom, and how these interactions evolve. Like most frameworks in complex systems, networks are used across many domains. To represent a system with a graph, one first has to answer two questions. Who or what is interacting? With whom is it interacting? This allows defining two things in a network: nodes and edges. Nodes represent each element of the

system, the network of elements is connected by edges which model their interaction. The remainder of this work relies on networks, so we will dive more in depth into what shapes a network and what its topology tells us about emergent phenomenons in complex systems. For now, let us focus on some applications of complex networks. Biology is at the forefront of complex systems. Network scientists in biology are, for example, interested in understanding diseases' propagation or interactions between proteins in organisms. The World Wide Web itself emerges from an interconnected network of computers. It is a goldmine to extract and study large-scale networks such as friendships on social media or co-authorship of authors in scientific publications. This manuscript will not suffice to present all the fascinating applications of complex networks. However, the next section is dedicated to introducing key characteristics and vocabulary of networks and describing emerging phenomenons and organization that span many scientific domains.

1.2 COMPLEX NETWORKS: TOPOLOGY OF INTERACTION

1.2.1 *Preliminary definitions*

Networks model complex systems and help extracting knowledge about their structure and interactions of elements that are part of these systems. In their simplest form, networks are a collection of points representing elements joined in pairs by lines representing their interactions. The nomenclature of complex networks names these points *vertices* or *nodes* and the lines that join them *edges* or *links*[1]. Figure 1.1 is an example of a network with six nodes and nine edges.

[1]These terms will be used indistinctly throughout this manuscript such that *node≡vertex* and *edge≡link*.

Definition 1 (Network)**.** Let $G = (V, E)$ be a network such that V is the set of vertices and $E \subset V \times V$ the set of edges connecting nodes.

The direct consequence of Definition 1 is that each node has a neighborhood N which may be empty if a node is not connected to any other node.

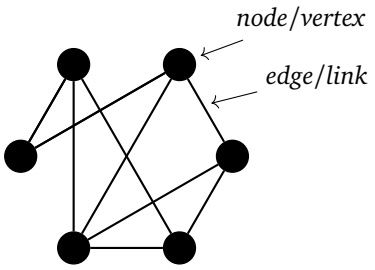


FIGURE 1.1: A simple graph consisting of 6 nodes and 9 edges.

Definition 2 (Neighborhood of a node)**.** Let $G = (V, E)$ be a network and $u \in V$ a node of G. The neighborhood of u, $N(u)$ is $N(u) = \{v \in V | (u, v) \in E\}$.

If a set of nodes are all neighbors of each other, we have a particular structure called a clique.

Definition 3 (Clique)**.** Let $G = (V, E)$ be an undirected network, a clique is a subset of vertices, $\mathcal{C} \subseteq V$, such that all nodes of the clique are adjacent, $\forall u, v \in \mathcal{C}, u \neq v \Rightarrow (u, v) \in E$

The size of the neighborhood, commonly called the degree of a node, may inform us on the amount of interaction a node has.

Definition 4 (Degree of a node)**.** Let $G = (V, E)$ be a graph and $u \in V$ be a node of G. The degree of node u, written $d(u)$ is $d(u) = |N(u)|$, the number of neighbors $N(u)$ of u.

Not all interactions are bidirectional, for example, in food chains, a mouse will rarely be the predator of a cat. That is where direction comes into play. It allows representing unidirectional interactions: who eats whom, who follows whom on a social media platform? This particular category is called directed networks, where interactions have a direction [2].

[2] Although better at modeling non-symmetric interactions, direction is often forgotten when applying algorithms to directed networks. We do not employ directed networks in our work and subsequently do not present the specifics of directed networks beyond a main definition.

Definition 5 (Directed Network)**.** Let $G = (V, A)$ be a directed network with V the set of nodes and A the set of arcs which are ordered pairs of nodes. Contrary to undirected networks and their edges, for two nodes $u, v \in E$, arc $(u, v) \neq (v, u)$. In arc $(u, v) \in A$, u is the *source* node and v the *target* node.

Edges may be weighted to quantify the relation between nodes. In this case, we talk about weighted networks. Weighted edges are for example, used in roadmap networks where street intersections represent nodes and edges represent streets. In such a network, each portion of road can be weighted according to the maximum flow of cars that can, for example, simultaneously transit on a particular section. Such a graph will see an additional set Ω added to its definition.

Definition 6 (Weighted Network)**.** Let $G = (V, E, \Omega)$ be a triplet (V, E, Ω) such that V is the set of nodes, E the set of edges and Ω, $\Omega : E \to \mathbb{R}$ associates a weight to each edge in E.

For weighted networks, the definition of degree can be extended to take this weight in consideration. We can define a weighted degree.

Definition 7 (Weighted Degree)**.** Let $G = (V, E, \Omega)$ be a weighted graph and $u \in V$ be a node of G. The weighted degree $d_w(u)$ of u is the sum of the weights of edges connecting u to its neighbors in $N(u) : d_w(u) = \sum_{v \in N(u)} \Omega(u, v)$.

Weighted degree can inform further than solely the number of connections, it provides another depth that quantifies the overall richness of these connections. The number of neighbors and degree of two nodes can be the same, but their weighted degrees can be immensely different.

Characterizing a network by its degree distribution is a good start. But many more features can be used to describe graph topology. Path length is an indicator of how far two nodes are in a graph. Let us first define a path and then path length in the context of unweighted and weighted graph.

Definition 8 (Path)**.** Let $G = (V, E)$ be a graph. A path of length k in G is an ordered list of edges $(e_1, \ldots, e_k) | \forall i \in \{1, \ldots, k\}, e_i =$

$(e_{i,s}, e_{i,t} \in E)$, s being the source node, t the target node, and $\forall i \in \{1, \ldots, k\}$, $e_{i,c} = e_{i+1,t}$. The path thus connects $e_{1,s}$ and $e_{k,t}$ with k edges.

- In an unweighted graph, the length of a path is the count of edges a path takes.
- In a weighted graph, the length of a path is the sum of the weights of edges a path takes.

We have defined the bare minimum to get started with the phenomenons that result from and contribute to the topology of networks. Further definitions will be distilled when necessary. Networks are essentially graphs, but not exactly. The next paragraph is dedicated to this question.

▸ NETWORK OR GRAPH?

The terms Network and Graph are used interchangeably in the literature. Yet, there is a subtle distinction between a network that refers to a real system and a graph that refers to the mathematical representation of such a system [BP16]. We may talk in terms of social network, connecting friends or family members, we also talk about metabolic networks that determine the physiology and biochemistry in a cell. However, when talking about the representation of such networks, we employ the terms social graph and metabolic graph. We in no way disregard this distinction, still, we will often employ network for graph and vice versa.

[BP16] Barabási and Pósfai, *Network science*

1.2.2 *It's a small world*

At first glance, the idea that two people on opposite sides of the globe can be connected by a few acquaintances seems unlikely. Yet, it is a common property to many networks, so-called *small-world* networks. Stanley Milgram, professor at Harvard, aimed to demonstrate this phenomenon with a life-scale experiment [TM69]. The setting was relatively straightforward. The goal of Milgram was to observe experimentally how many people are needed to transmit a folder from a randomly selected subject to a designated person for which some details like name, town, or occupation are given. Participants were encouraged to hand over or send by post the folder to the person they thought was more likely to help this quest—as in closest to the target individual—and that they knew on a first name basis. The source participants were chosen in Omaha, Nebraska. The target was a stockbroker in Boston, Massachusetts. The average number of intermediaries to reach the stockbroker in Boston was found to be 5.2 which can seem quite short. The fact that participants had access to specific information regarding the target individual was undoubtedly a contributing factor in shortening the chains. Still, this experiment is a good example of the small-world phenomenon in real-world networks.

Small world at the Santa Fe Institute. In 2022, I attended a complex system summer school at the Santa Fe institute in New Mexico, and dinners were a good occasion to share about our culture and our backgrounds. One of the Ph.D. candidates from Arizona State University shared her background, moving from the south of Spain to the US to pursue a Master's degree and a Ph.D. To our surprise, I happen to know a childhood friend of hers that I met while studying in Germany. Naturally, we exclaimed, "It's a small world!".

[TM69] Travers and Milgram, "An Experimental Study of the Small World Problem"

Definition 9 (Small-world network [WS98])**.** A small-world network $G = (V, E)$ is a graph where the majority of nodes are not di-

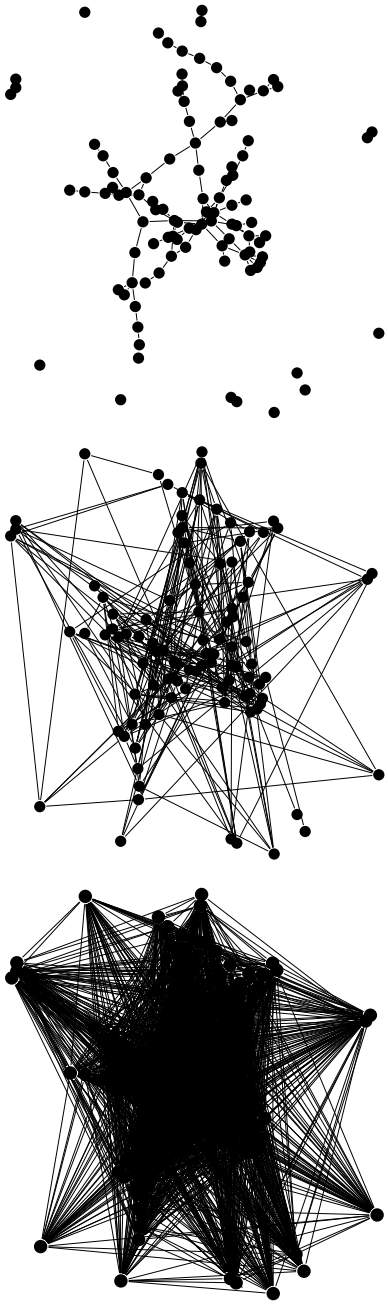


FIGURE 1.2: Three Erdős-Rényi random graphs (top to bottom): $\mathcal{G}(50, 0.02)$, $\mathcal{G}(50, 0.05)$, and $\mathcal{G}(50, 0.5)$. The higher the probability p in $\mathcal{G}(n, p)$ the more edges in the graph.

rectly linked, yet the neighbors of any given node tend to be connected to one another. This characteristic allows for most neighboring nodes to be reached from any starting point in just a few steps or hops. The distance L_{uv} between two nodes $u, v \in V$ chosen at random grows logarithmically with the number of nodes $n = |V|$ in the network such that:

$$L_{uv} \propto n \tag{1.1}$$

In 1959, Paul Erdős and Alfréd Rényi introduce a random graph model with the goal of easily generating random graphs. The Erdős-Rényi model (ER) [ER58], named after its initiators, generates random graphs that can be for example used to investigate connectivity, clustering, and degree distributions in a controlled and tractable manner. Random graphs also serve as a comparison point for real systems. Indeed, random graphs patterns of connectivity can be compared with those observed on real-world networks. The ER model ($\mathcal{G}(n, p)$), generates a random graph with n nodes. Without edges, the random graph will be composed of n connected components of size one. At most, $\frac{n(n-1)}{2}$ edges can connect n edges to form a fully connected graph. Parameter p is the probability for each of these $\frac{n(n-1)}{2}$ edges to exist independently of each other. On average, this will result in $\frac{n(n-1)}{2} \cdot p$ edges. Three ER graphs with fifty nodes are presented Figure 1.2 with a probability p set to 0.02 where we can see multiple components, 0.05 where connectivity is denser and a single giant component is formed, and with $p = 0.5$ a much denser connectivity.

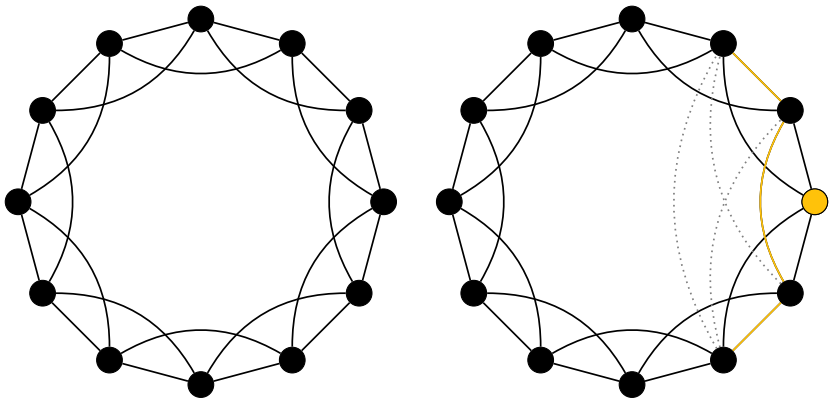
The ER model produces random graphs with unrealistic degree distributions that converge to a Poisson distribution rather than a power law degree distribution that is often observed in real graphs. Furthermore, unlike in real graphs, clustering coefficients tend to be small. It is unfortunate, as high clustering coefficients are distinctive of real life networks and especially those exhibiting small-world properties. But what exactly is the clustering coefficient?

It measures how clustered the neighborhood of a node is, put more simply, how intertwined the neighbors of a node are.

Definition 10 (Clustering Coefficient of a node [WS98])**.** Let $u \in V$ be a node and $N(u)$ its neighborhood and $d(u)$ its degree (k for simplicity in random graph models). The clustering coefficient of node u is:

$$c(u) = \frac{|\{(v, w) \in E | v, w \in N(u)|}{C^2_{d(u)}} \tag{1.2}$$

An illustration of the clustering coefficient of a node is presented Figure 1.3. It characterizes the notion of transitivity, that is, for two nodes u and v neighbor of each other and a third node w neighbor of v, whether u and w are connected. In that case, if all relations are transitive, all nodes are part of the same clique.

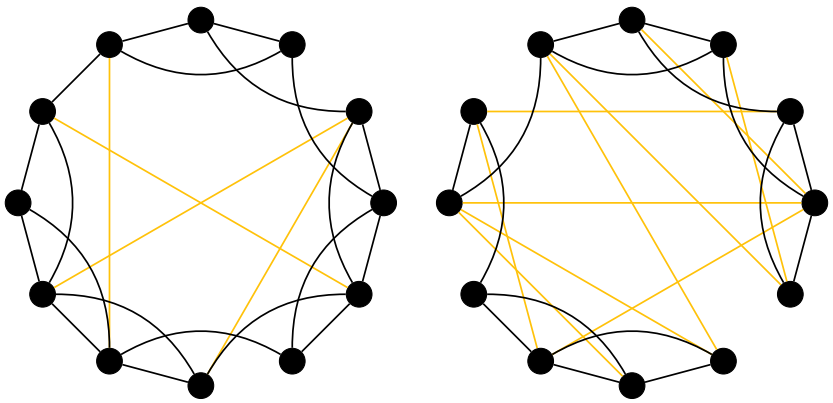


(a) A regular graph with 12 nodes, 23 edges, each node has degre $k = 4$.

(b) The node in red has 4 neighbors connected by 3 edges (red) out of the 6 (3 grey do not exist) if they were all connected. Thus, the clustering coefficient of node A is $c(A) = \frac{3}{6} = 0.5$

FIGURE 1.3: Illustration of the clustering coefficient of a node in a regular graph.

Watts and Strogatz in 1998, in their Nature article *Collective dynamics of small world networks* [WS98] introduce an alternative model to generate random graphs that can have high clustering coefficients. Their model relies on regular lattices like the one presented Figure 1.3(a). The Watts–Strogatz (WS) model $\mathcal{G}(n, k, \beta)$, starts with a regular graph with n nodes, where node degree is k and whose clustering coefficient is relatively high. In an iterative process, it rewires each edge of the regular lattice with a probability β under the constraint of avoiding self-loops and duplicating links. This rewiring process constrained by probability β leads to new links, thus shortening the average path length between nodes Figure 1.4. This rewiring procedure has an impact on clustering coefficient by removing, to some extent, the regularity in node neighborhood. Furthermore, like the ER model, the WS model does not generate graphs with a natural degree distribution as that of networks in nature.



(a) A regular graph with 12 nodes, $k = 4$ after applying the Watts–Strogatz rewiring procedure with probability $\beta = 0.2$, $\mathcal{G}(12, 4, 0.2)$.

(b) A regular graph with 12 nodes, $k = 4$ after applying the Watts–Strogatz rewiring procedure with probability $\beta = 0.5$, $\mathcal{G}(12, 4, 0.5)$.

FIGURE 1.4: Illustration of the rewiring procedure in the Watts–Strogatz model.

Let us go back to small-world networks. In their seminal article, Watts and Strogatz present three small-world networks. The network of co-starring actors in movies listed in the Internet Movie Database (IMDB), the power grid of the western states of the US with generators, transformers, and substations linked by high-voltage lines. The third network is the brain network of the worm C. elegans. They all exhibit small-world behavior with a diameter between two and four for the networks of actors and C. Elegans, and close to nineteen for the power grid network. WS' model can approximate the diameter of natural networks of a similar size. However, when it comes to clustering coefficient, the average value is at least one order of magnitude smaller than in each actual network. What about language? Does it exhibit small-world properties?

[CS01] Cancho and Solé "The small world of human language"

Cancho and Solé [CS01] investigate this property in word co-occurrence networks. Word co-occurrence networks are constructed from large collections of texts called corpus. Words are represented by nodes, and their co-occurrence together in a sentence or at a set distance leads to the creation of an edge in the graph linking two words. Cancho and Solé construct two co-occurrence networks respectively called Unrestricted Word Network (UWN) and Restricted Word Network (RWN) from three quarters of the 10^7 words in the British National Corpus (BNC)[3]. UWN contains all words from the corpus, whereas RWN is limited to words that co-occur more than what could be expected by chance if words were randomly ordered in sentences. That is, for two words i and j in the lexicon, $p_{ij} > p_i p_j$. Their observation is similar to that of Watts and Strogatz, networks of the language have a small diameter ($L_{UWN} = 2.63$, $L_{RWN} = 2.67$) and a high clustering coefficient ($c_{UWN} = 0.687$,$c_{RWN} = 0.437$). Subsequently, graphs stemming from the language are small world. On average, words are at most at a distance of three from each other in the underlying structure of the language. This is significant, as the closer words are to each other, the easier it is for sentences to be constructed by a speaker.

[3] more details on BNC can be found Section 4.3.2

Small-world behavior does not emerge randomly, it is the consequence of the evolution of the system that is modeled by the network. If we go back to the properties of complex systems mentioned Section 1.1.1, small-world organization is an emergent phenomenon that has roots in how the network grows. We will now focus our interest towards preferential attachment that plays a major role in how networks form, grow and organize.

1.2.3 *Growth and Preferential attachment*

Growth and preferential attachment are tied together. They influence how underlying complex systems modeled by networks form and evolve. Throughout their life, networks grow, new information is added to them, some can be removed. Let us take the example of the internet. At its birth, it contained a single page[4], built in 1991 by Tim Berners-Lee at CERN. From there, it grew to be a globalized network of pages linked to one another with hyperlinks. At its beginnings, web directories such as *Yahoo!* gathered links to many web pages, which was later supplanted by search engines like *Google*. The web grew from one page to billions today. This growth process is not random but rather organic. Preferential attachment plays a major

[4] http://info.cern.ch/hypertext/WWW/TheProject.html

role in how most real-world networks grow.

Definition 11 (Preferential Attachment [BA99])**.** Preferential Attachment is a growth process in which new nodes added to a network tend to connect preferentially to high degree nodes. This process is sometimes deemed a "rich gets richer" phenomenon, where nodes that already have many neighbors are more subject to see their degree increase with the addition of new nodes.

Preferential attachment has been illustrated in numerous domains. First in 1923, in mathematics as the Pólya process [EP23] where black or white balls are randomly drawn from an urn. This ball is then returned to the urn accompanied by a ball of similar color, thus increasing the probability of drawing a ball of that color in the future. In statistics, this process is named after Yule that modeled the growth of populations. Zipf [Zip35; Zip49] presented a similar process to demonstrate the distribution of wealth in society. Simon [Sim55] used it to describe the size of cities, the number of scientific publications and the distribution of word frequencies. Price [Pri76] theorized cumulative advantage to explain papers' citations. Merton [Mer88] later presented the *Matthew effect* where wealth is distributed among individuals based on what they already have.

[EP23] Eggenberger and Pólya, "Über die Statistik verketteter Vorgänge"

[Zip35] Zipf, *The psycho-biology of language.*

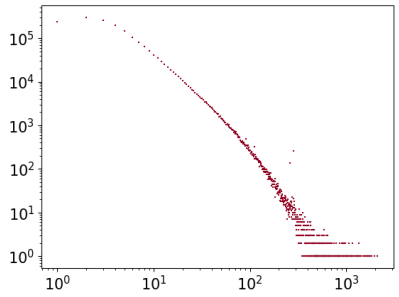
[Zip49] Zipf, *Human behavior and the principle of least effort.*

[Sim55] Simon, "On a Class of Skew Distribution Functions"

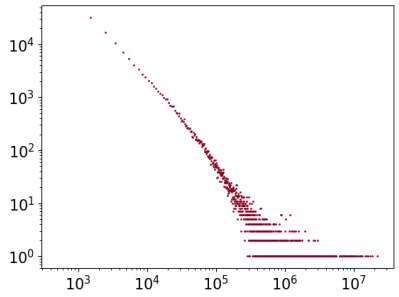
[Pri76] Price, "A general theory of bibliometric and other cumulative advantage processes"

[Mer88] Merton, "The Matthew Effect in Science, II: Cumulative Advantage and the Symbolism of Intellectual Property"

All these processes are explained by preferential attachment. But what exactly does preferential attachment entails? Complex networks that grow according to the preferential attachment process tend to have particular distributions: power-law or Pareto distributions. A simple illustration of this distribution is for example wealth: in a given population, few people are extremely wealthy, and a large proportion is of modest wealth. This observation can be made in many real-world scenarios and lead to power-law distributions. In such distributions, the relative change in a quantity, for example wealth, results in a relative change in the number of individuals in a wealth bracket that may be proportional to a power of the change: one quantity varies as the power of another. How does that translate into data?



(a) Distribution of the number of collaborations (degree of each node is the number of publications) in the graph of the DBLP computer science bibliography available from KONECT [Kun13]. In abscissa, number of publications (degree), in ordinate, number of authors with such a number of publications.



(b) Distribution of the population of 118k cities with more than 1,000 inhabitants from GeoNames [Geo05]. Population in abscissa and number of cities with such a population in ordinate.

FIGURE 1.5: Example of two systems exhibiting power-law-like distributions.

Figures 1.5(a) and 1.5(b) are two examples of systems exhibiting power-law-like distributions. The first distribution is extracted from a network available on the KONECT [Kun13] project website. This network models authors' collaborations in the computer science DBLP bibliography ($n = 1,824,701, m = 29,487,744$). This graph is multi-edge, with each edge representing a paper co-published by two authors. The degree of a node thus accounts for every single time an author has collaborated with another scientist.

[Kun13] Kunegis, "KONECT – The Koblenz Network Collection"

What we can see when plotting this distribution on a log-log plot is that few authors have many collaborations, and many have fewer papers and thus collaborations. Power-laws are frequently indicative of a hierarchy. For example, in the DBLP network, a senior Principal Investigator (PI) with years of experience in the field and a tight-knit network of collaborators will most probably have many collaborations. Influential PIs are part of teams and frequently oversee the work of more junior researchers with whom they publish. They may collaborate simultaneously with several junior researchers. That is a prime example of preferential attachment in a way, new researchers to the field will most probably be introduced to a community by a member with more collaborations. Our second example Figure 1.5(b) is that of 118,451 cities' population from GeoNames [Geo05]. Similarly to the collaboration example we previously plotted, the distribution of city sizes seems to exhibit a similar distribution. Large cities such as Beijing with over 21 million inhabitants are rare on earth. Smaller cities and villages are far more common. Yet, again, if we consider metropolitan areas, they sometimes exhibit a sense of hierarchy between a city concentrating the majority of the population and smaller suburbs.

[Geo05] GeoNames *All Cities with a population > 1000*

Let us circle back to language. Does language also exhibit a power-law distribution for its words? Zipf [Zip35] shed light on the observation that word frequency seems to decrease inversely to rank–assuming words are ordered from most frequent to least frequent. Thus, word frequency seems to follow a power-law. We have plotted the word distribution in the `BNC` corpora (described later, Chapter 4) Figure 1.6. According to Zipf's law the frequency distribution resembles what we would expect from power-law distributions. Both Cancho and Solé [CS01] and Dorogovtsev and Mendes [DM01] treat human language as a word web of co-occurring words. We have seen previously how Cancho and Solé [CS01] demonstrate that human language is small-world. Dorogovtsev and Mendes [DM01] theorize the evolution of language as an evolving web of words that lead to a two-slope power-law distribution.

[Zip35] Zipf *The psycho-biology of language.*

[CS01] Cancho and Solé "The small world of human language"

[DM01] Dorogovtsev and Mendes "Language as an evolving word web"

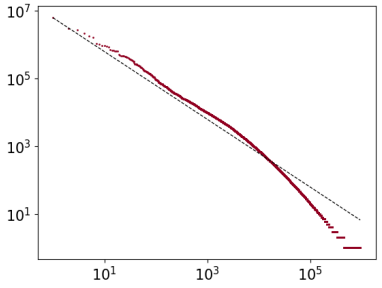


Figure 1.6: Zipf's curve of frequency according to rank in the BNC. Dashed line is ideal Zipf curve if law holds for rank 1.

This growth leads to an organization of the language into a core kernel of the language whose size does not evolve much through the addition of new words. The interesting finding is that the core of language does not have the same degree distribution as the more peripheral, rarer words. This results in a two-slope degree distribution, much like what we can see Figure 1.6 around the 10^4 mark.

As with previous phenomenons, a new random graph model is introduced to provide networks with preferential attachment phenomenons. In Barabási and Albert's model [BA99], a new node in the random graph connects with a higher probability with nodes of large degrees. Subsequently,

[BA99] Barabási and Albert "Emergence of scaling in random networks"

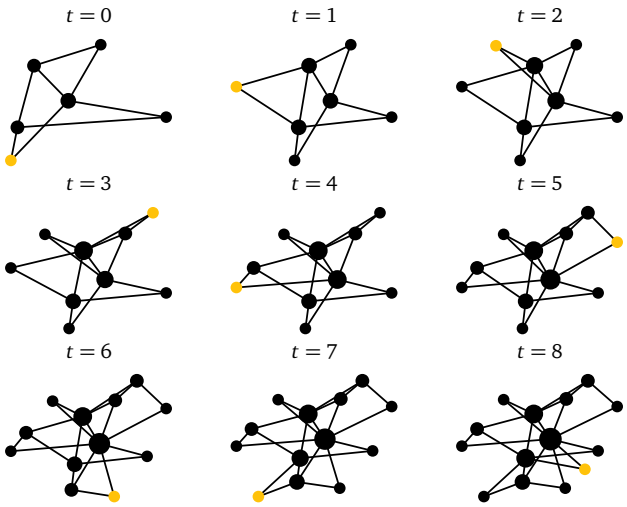


FIGURE 1.7: Barabási-Albert preferential attachment model in a growing network. Initial network has $m_0 = 6$, at each addition of a node (in yellow), it is linked to 2 nodes based on their degree. Node size is proportional to their degree.

high-degree nodes tend to gain edges and thus neighbors at a rapid rate when lower-degree nodes are less susceptible to being drawn randomly and see their degree increase. New nodes preferentially link to high-degree existing nodes, as illustrated Figure 1.7.

So far, we have observed that interactions of many elements at the microscopic level, namely node-node interactions, give rise to macroscale properties. That is the emergence phenomena that we described Section 1.1.1. For example, power-law distributions of degrees and small-world topology emerge from the rules that govern the way a network grows. Networks also lack central control. Their topology may be influenced by nodes with relative importance in the network (*e.g.*, large degree, acting as a hub), but most networks do not collapse upon the removal of some of their nodes or vertices. However, there is still an element of hierarchy among nodes, some may hold more critical positions in the network, influencing the average distance between any two nodes. For example, in language, if we construct a network of words occurring together in sentences, the removal of the article "*the*" has a considerable influence on the topology since it bridges between many words as it is very frequent[5]. Furthermore, hierarchy is not solely restricted to the node, we will see in the next section that communities, dense groups of connected nodes, are at an intermediary level of organization between the microscopic (node-node interactions) and the macroscopic scales. We will employ the term mesoscopic to refer to this level.

[5] Despite "*the*" being a frequent term, it will almost systematically be removed from co-occurrence networks as its benefit semantic content/frequency is considered low. Refer to Section 4.3.1 for more information on stopwords.

1.2.4 *Communities*

A property exhibited by many networks is community structure. Most networks are not a monolithic mass of nodes linked by edges but are rather organized in dense groups of nodes that are more connected with one another than with the rest of the network. That is what we call a community,

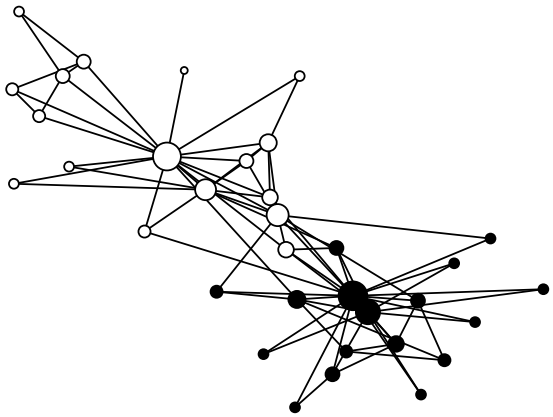


FIGURE 1.8: Visualization of the communities in Zachary's Karate Club.

part of a community structure. Naturally, the average clustering coefficient of a network informs on the potential of a network to have an underlying community structure.

Definition 12 (Community)**.** According to Girvan and Newman [GN02], communities are: "subsets of vertices within which vertex-vertex connections are dense, but between which connections are less dense".

Communities in social networks are quite straightforward: a family, a company, an athletic club, *etc*. However, they can be found in a wide variety of areas, ranging from biological to technological networks. A notorious example is Zachary's karate club network. This network of a university karate club is frequently used to illustrate the notion of communities in social networks. The fission into two communities results from a conflict between the karate instructor and the president of the club. We can see Figure 1.8 the two communities of the network. Naturally, most networks can be divided in many more than two communities. For example, your relationships are probably split between work, social activities, family, *etc*.

Detecting communities is a task that has gathered much interest and led to the development of many approaches that we detail Section 2.3: hierarchical clustering, edge betweenness, spectral clustering, statistical inference, network dynamics, and optimization. Given a graph, the objective of community detection algorithms is to find a partition of nodes, usually in disjoint communities so that links between communities are scarce and links within communities are dense.

Definition 13 (Partition)**.** An ensemble $P = \{P_1, P_2, \ldots, P_k\}$ is a partition of the nodes V of a graph if and only if it is a set of disjoint

subsets of V whose union equals to V, mathematically:

- $\forall i \in \{1, \dots, k\}, P_i \subseteq V$
- $\bigcup_{i \in \{1,\dots,k\}} P_i = V$
- $\forall i, j \in \{1, \dots, k\}^2, i \neq j \Rightarrow P_i \cap P_j = \varnothing$

However, it is an arduous task when most networks do not have ground-truth community structure to compare against the detected partition. Even then, so-called "*ground-truth*" communities based on node metadata are discussed [DBL17; CCP18]. If a node comprises multiple types of metadata, many partitions are possible to gather nodes in communities. Furthermore, depending on the chosen "*ground-truth*" partition, communities might not be reflected structurally.

[DBL17] Dao et al., "Community detection methods can discover better structural clusters than ground-truth communities"

[CCP18] Chakraborty et al., "Metadata vs. Ground-truth"

Still, if communities are detected automatically, it is of the utmost importance to estimate the strength of division of a network into communities. Subsequently, Newman [New06] introduces Modularity.

[New06] Newman "Modularity and community structure in networks"

Definition 14 (Modularity)**.** The modularity $Q_{\mathcal{C}}$ of a partition $\mathcal{C}$ described in Definition 13 measures the strength of the connection between two vertices in a network compared to the probability that these two vertices would be randomly connected based on their degree.

$$Q_{\mathcal{C}} = \frac{1}{2m} \cdot \sum_{i,j \in V \times V} \left[A_{ij} - \gamma \frac{d(i) \cdot d(j)}{2m} \right] \cdot \delta(C_i, C_j) \tag{1.3}$$

A_{ij} denotes whether edge $ij \in E$ (the weight of edge ij in a weighted graph), $\delta(u, v)$ is 1 if $u = v$ (are in the same community), 0 otherwise, γ is the multi-resolution parameter. This parameter is useful to alleviate the resolution limit, which impedes uncovering small communities and might lead to internally disconnected communities [FB07] [a].

[a] see Section 2.3.1 for more details

Communities can be essential in understanding the macroscale structure of a network. Section 2.3 of Chapter 2 is dedicated to community detection algorithms, since it is at the heart of this work of representation learning. I will touch on applications of community detection and methods and a wide panel of algorithms to partition nodes.

1.3 GRASPING COMPLEXITY

With this introductory chapter, we have taken a stroll through the world of complex systems. Our key outtakes with this tour of complex systems are the following:

(i) Drawing a clear definition of complexity is hard. It spans many domains, even beyond fundamental sciences. We can, however, highlight the main characteristics of complex systems that are relevant to our work on complex networks: many interacting elements, emergence, hierarchy, and lack of central control.

(ii) Tools to study complex vary across domains. The main domains of complex systems are non-linear dynamics, game and information theory, simulation with agent-based models, complex networks.

(iii) Complex networks, that will be our focus in this work, exhibit characteristics of complex systems: small-world, preferential attachment, and community structures are key phenomenons that emerge in complex networks.

(iv) Communities form a substructure of some networks in which nodes are more connected than with the rest of the network. Detecting them can inform us on the topology of the network.

Because of the sheer number of connections within a network, extracting a representation of a graph's topology and the position of a node within a network can prove useful. A latent space extracted from the network can most probably grasp local information but also more distant topology characteristics, such as the position within a community or the network. Modeling networks with a latent space summarizes their structure and can provide essential information for tasks in which vector spaces are better suited, like classification or regression tasks. Moreover, part of the benefit of relying on a graph structure and extracting a vector representation is the ability to come back to the network and observe the local structure around a node. Even stronger, the ability with some representation learning approaches to provide interpretable spaces without the need to come back to the network.

As we will see Chapter 2, there is a kinship between representation learning of networks and text. Latent spaces which we will call embedding spaces (or models) are meant to derive representations encompassing characteristics of structure, whether it is from an image, a text, or a network. After demonstrating the kinship between word and graph embedding approaches and the limitations of current methods, we will present a joint approach to extract graph embeddings that can also work as a word embedding method when applied to co-occurrence networks.

In the next chapter, we present why representation learning is of interest to grasp the structure of graphs and language, and uncover the underlying complex systems they model. We will see that, although, at first glance, graphs, and texts are quite different, embedding methods for both types of structures bear resemblance, and so do their limits. Finally, we introduce our unified approach to graph and word embeddings.

2

From network to vector: learning representations

Chapter contents

Synopsis

Networks are intricate to exploit without third-party representations that are able to grasp their complex topology in a vector of reduced size. Commonly called embeddings, they are useful to provide a compact representation of networks encompassing their organization in input of machine learning algorithms. Network embedding has the objective of extracting such representation. However, embeddings are not limited to networks, they are widely used to represent data and particularly from large textual corpora with word embedding. Network embedding and word embedding have the same goal: providing a compact vector representation of data while preserving and summarizing local and broader similarities between items. Naturally, because network and word embedding share a common goal, they also share a kinship with regard to the philosophy behind representation learning (distributional hypothesis) and the methods involved to embed data. However current methods still suffer from limitations related to their compute complexity and lack of intepretability. The race towards ever-larger and complex architectures results in more energy consumption and harder to implement solutions. The lack of interpretability with "black box" models means that it is hard to audit and comprehend how sense is built in an embedding model. In order to address these limitations, we developed an embedding method able to derive interpretable representations both from networks and text with low compute. We introduce the theoretical framework `LDBGF` involving cliques to compress information from networks in the form of bipartite networks and two implementations inspired by this framework: `SINr-NR`, relying on community detection, and `SINr-MF` that performs a matrix factorization of the adjacency matrix of a network.

Résumé

Exploiter la richesse des connaissances offertes par les réseaux et leur topologie est complexe sans faire appel à des méthodes tierces pour la représentation de leur contenu. Ces méthodes, communément appelées méthodes de plongements (*embeddings*) permettent d'obtenir des représentations compactes pouvant être utilisées en entrée d'algorithmes d'apprentissage automatique. Les plongements de graphes ont pour objectif de fournir ces représentations sous forme de vecteurs. Cependant, les méthodes de plongements ne sont pas restreintes aux réseaux, elles sont légion lorsqu'il s'agit de représenter des données, notamment du texte au travers de ce que l'on appelle plongements de mots. Les plongements de graphes et de mots partagent un objectif commun : fournir une représentation compacte des données tout en préservant les similarités et dissimilarités entre ces données dans l'espace d'origine. Puisque cet objectif est commun aux méthodes de plongements, ces dernières reposent aussi sur une hypothèse commune, l'hypothèse distributionnelle. Aussi, les méthodes employées présentent de grandes similitudes qui témoignent d'une parenté entre plongements de graphes et de mots que nous allons mettre en lumière dans ce chapitre. Néanmoins, les méthodes de plongements ne sont pas sans défauts, notamment en ce qui concerne l'interprétabilité des représentations et leur complexité de calcul. La course vers des modèles de plongements toujours plus grands résulte en une croissance du besoin en ressources de calcul et de stockage ainsi qu'en une complexification de leur apprentissage. De plus, le manque d'interprétabilité associé aux modèles "boîte noire" signifie qu'il est difficile d'inspecter leur structure et de comprendre comment la représentation des données est formée. Afin de répondre à ces problèmes, nous avons développé une méthode de plongements capable de dériver des représentations interprétables, à la fois à partir de réseaux et de textes, en peu de temps et avec peu de ressources de calcul. Nous introduisons le cadre théorique `LDBGF` utilisant les cliques d'un réseau pour découvrir une projection bipartie permettant de compresser le graphe. Nous décrivons ensuite deux implémentations inspirées par `LDBGF`: `SINr-NR`, qui repose sur la détection de communautés, et `SINr-MF` qui effectue une factorisation de la matrice d'adjacence d'un réseau.

2.1 THE WORD-GRAPH EMBEDDING KINSHIP

2.1.1 *From the distributional hypothesis to embedding models*

▸ DISTRIBUTIONAL HYPOTHESIS FROM THE LINGUISTIC STANDPOINT

Collections of data are mostly structured. In a graph, the fact that two nodes are connected by a link means something in the system it models. The same goes for the co-occurrence of words in sentences, they are not randomly put next to one another to form sense. These assumptions seem logical for anyone speaking a language or evolving in social circles. However, elaborating a mathematical representation of data that encompasses these characteristics is less obvious. In the 1950s, linguists Zellig Harris and John Firth devised the distributional hypothesis, which is still to this day relevant to word embedding.

Distributional models come with the idea that a meaning is contextual, that it depends on surrounding lexical items and emerges from use. The combination of the uses for a word could thus help grasp its meaning.

> "The placing of a text as a constituent in a context of situation contributes to the statement of meaning since situations are set up to recognize use. As Wittgenstein says, 'the meaning of words lies in their use.' The day to day practice of playing language games recognizes customs and rules. It follows that a text in such established usage may contain sentences such as 'Don't be such an ass!', 'You silly ass!', 'What an ass he is !' In these examples, the word ass is in familiar and habitual company, commonly collocated with you silly—, he is a silly—, don't be such an—. **You shall know a word by the company it keeps!** One of the meanings of ass is its habitual collocation with such other words as those above quoted."
>
> Firth [Fir57]

Firth [Fir57] illustrates here, not without humor, the concept of representation from context. The famous "You shall know a word by the company it keeps!" is often used to summarize the distributional hypothesis. The idea that a word's meaning can be represented using its varied contexts. So much so that a word's representation can be a composite of its different meanings pertaining to the contexts in which it appears. By using context and co-occurrence of items, we not only grasp the lexical structure, but also the syntactic structure to form meaning [SS91].

[Fir57] Firth "A synopsis of linguistic theory 1930-55."

[SS91] Sinclair and Sinclair, *Corpus, Concordance, Collocation*

> "A phrase can be defined for the moment as a co-occurrence of words which creates a sense that is not the simple combination of the sense of each of the words.[...] it is much more fruitful to start by suppos-

> ing that lexical and syntactic choices correlate, than that they vary independently of each other."
>
> Sinclair [SS91]

Sentences are structured sequences of words from which sense emerges. However, collocation in a sentence is not all, language is not meaning, and conversely, meaning is not restricted to language. Firth [Fir57] indicates that meaning is forged by human communication and does not conform to the structure of the physical world that is ruled for example by differential equations. Meaning goes beyond language, feeding from subjective experiences of speakers, and is thus difficult to grasp.

> "[...] the distinction between distributional structure and meaning is not yet always clear. Meaning is not a unique property of language, but a general characteristic of human activity. It is true that language has a special relation to meaning, both in the sense of the classification of aspects of experience, and in the sense of communication. But the relation is not simple. For example, we can compare the structures of languages with the structure of the physical world (*e.g.* the kind of phenomena that are expressed by differentiation and integration in calculus), or with what we know about the structure of human response (*e.g.*, association, transference). In either case, it would be clear that the structure of one language or another does not conform in many respects to the structure of physical nature or of human response–*i.e.* to the structure of objective experience from which we presumably draw our meanings"
>
> Harris [Har54]

Meaning is complex, and grasping it fully from language seems unlikely if it is influenced by one's experiences. We will later study representations based on the distributional hypothesis for language. Grounding these representations so that they are aware of language use is an ongoing research problem [BK20; Pav23]. Not all meaning may be accessible through language, but we can still grasp some meaning of lexical items through their distribution. With a high-enough number of contexts, we might tend toward capturing an approximate meaning of words and also identify similarities and dissimilarities between them.

[BK20] Bender and Koller, "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data"

[Pav23] Pavlick, "Symbols and grounding in large language models"

> "Though we cannot list all the co-occurrents (selection) of a particular morpheme, or define its meaning fully on the basis of these [co-occurents], we can measure roughly the difference in selection between elements, say something about their difference in meaning,

> and also [...] derive certain structural information."
>
> Firth [Fir57]

At the time, it was difficult to capture many contexts for a word and apply probabilistic methods as we do today with word embedding models. Yet, the idea of sampling contexts and using them to define meaning had already emerged. So did the idea that if two words frequently appear in similar contexts, there might be underlying information about their meaning or position in a sentence: they might be related (meaning-wise) or assume a similar position within a sentence (structure-wise). A parallel can be made for networks. When sentences are structured sequences of words, networks are nodes which form a structure due to the edges they share. Two neighboring nodes are thus not connected by chance. The fact that they appear in the vicinity of each other carries meaning regarding the network structure. The same goes for two nodes having the same neighborhoods, meaning that they interact with the same set of nodes. There can thus be a transposition of the distributional hypothesis to networks and a kinship that we will see does stem from the distributional hypothesis but also stems in the methods employed to learn representations from networks and texts.

▸ IMPLEMENTATION OF DISTRIBUTIONAL HYPOTHESIS IN DISTRIBUTIONAL MODELS

J. R. Firth passed away in 1960, but he might have wished to know what emerged from the distributional hypothesis. With the increase in computation capabilities and data availability, distributional models, rooted in the distributional hypothesis, have allowed extracting large quantities of contexts and using them to represent items. Distributional models can derive vector representations of data based on context. What is called word embedding in NLP, vertex embedding in network theory, has the same goal. Providing a vector representation for the lexicon of a corpus or a node in a network based on their context. The context in a sentence is composed of the words that co-occur with the word considered. The context of a vertex can be its direct neighborhood, but may also be grasped further away with random walks. Random walks (Definition 15), can explore the local neighborhood of a root node step by step by transiting along edges. We can view these sequences as some kinds of sentences of walks on networks, but with nodes in place of words. And thus, context would be extended to a node's neighbors of neighbors. In this context, we might be tempted to adapt Firth's famous claim to nodes: *You shall know a node by the company it keeps*. We will see Section 2.1.3, that in some cases, similar algorithms can be applied to sentences and random walks.

From these contexts, the goal of distributional methods, regardless if they work with words or nodes, is to embed close to one another, nodes or words that have related meanings or positions in their structure (network or language).

Since the distributional hypothesis is central in representing nodes and words, there is a clear kinship at this level between graph embedding and word embedding. Yet, the kinship between graph and vertex embedding

does not stop at the distributional hypothesis. Algorithms employed in word and graph embedding share common grounds. We will see Sections 2.1.2 and 2.1.3 that it is mostly NLP's word embeddings that influence graph embeddings, and we will highlight the extent of this kinship in implementations. We start by introducing word embedding that extracts representations of words before introducing graph embedding that build representations from graphs.

2.1.2 *Representing language*

Processing large collections of documents to automatize document classification, information extraction or sentiment analysis requires a representation of its content. The simplest and rawest approach to representing words is one-hot encoding. Let's say we have a corpus of documents with a vocabulary of size 1,000, we first build a registry of the vocabulary, ordered alphabetically. Then, for each word in the corpus, we build a vector of size 1,000 with a single value 1 at the index of the word in the vector. What happens when the collections increase to a vocabulary of 10,000, or 100,000 words? The size of the vector needed to represent each word increases linearly with vocabulary size. That is the curse of dimensionality, increasing vocabulary size mechanically increases memory required to store representations. Furthermore, the representations provided by one-hot encoding do not inform on the meaning of words as described in the distributional hypothesis, we cannot compare representations to establish similarities between items since all dimensions are orthogonal. To alleviate the curse of dimensionality, and to provide vector representations embedding meaning, new models were engineered, to keep the footprint of representations manageable despite the growing sizes of corpus, and also include more semantic content.

Word embedding methods have multiplied from the first models in the 1980s and 1990s to the large pretrained models we know today. However, they still rely on the distributional hypothesis. We will retrace the path followed by embedding models back to the late 1980s and discover broad families of methods.

▸ From document to word representation

In the late 1980s, Dumais et al. [Dum+88] introduced *Latent Semantic Analysis* (`LSA`). Known today as a topic modeling or document vectorization method, `LSA` uses a document-term matrix in combination with Singular Value Decomposition (`SVD`) to represent documents and words. `SVD` relies on singular values of a matrix to provide a decomposition in a product of matrices that can help reduce its dimension.

[Dum+88] Dumais et al. "Using latent semantic analysis to improve access to textual information"

In the case of `LSA`, the sole context considered is the appearance of words in documents, which may be altered with a weighting scheme such as `tf-idf`. Because the order of words within documents does not make a difference in the output representation, it is called a *bag-of-words* model. Representation methods considering local context, closer to the distributional hypothesis, were later developed by Schütze [Sch92] that applied `SVD` to co-occurrence matrices (matrices tallying the number of times words co-

[Sch92] Schütze "Dimensions of meaning"

occur in sentences), and predicted, with the growth in capacity and availability of computation resources, the emergence of more co-occurrence-based methods. The *Generalized Latent Semantic Analysis* (GLSA) introduced by Matveeva et al. [Mat+05] build on the LSA approach but with a co-occurrence matrix. GLSA uses PMI (Definition 17) as a reweighting scheme for the co-occurrence matrix before reducing dimension, using SVD to keep only relevant co-occurrences. This pipeline is again studied years later in Levy et al. [LGD15] showing the relevance of this combination even against state-of-the-art methods at the time. Methods affiliated with LSA constitute the first family of co-occurrence-based methods to represent words from large corpora. As we will see Chapter 3, SVD is also useful as a dimension reduction approach in some graph embedding methods.

[Mat+05] Matveeva et al. "Term representation with generalized latent semantic analysis"

[LGD15] Levy et al. "Improving Distributional Similarity with Lessons Learned from Word Embeddings"

GloVe

In keeping with methods of the LSA family, other methods factorize a co-occurrence matrix. *Global Vectors for word representation* (GloVe) was popularized by Pennington et al. [PSM14] in 2014, at a time when the community was ready to use pretrained vectors as input to many downstream tasks. As its name states, GloVe has the objective of using global and local information to build word representations. Global information is captured by considering the whole corpus while constructing the co-occurrence matrix. It looks at the distribution of words across the entire dataset, capturing how often words co-occur with each other irrespective of specific local contexts. This global perspective helps in capturing broader semantic relationships between words. Local information is captured with a sliding context window used to retrieve co-occurrences in the corpus. If two words appear within the same sliding window, then they co-occur.

[PSM14] Pennington et al. "GloVe: Global vectors for word representation"

GloVe is a global log bilinear model. Put more simply, GloVe's objective is to factorize the log co-occurrences of words by the product of two embedding matrices learned jointly at training time: U for words and V for contexts. Each word i is represented by a vector in $u_i \in U$, which captures semantic information and is supposed to encode linguistic properties of the word, such as its meaning. V represents the embedding matrix for the context words j surrounding a target word within the context window with an embedding $v_j \in V$. These vectors are used to model the co-occurrence of two words within the same context. GloVe is philosophically close to SVD and NMF which decompose a matrix into the product of two or more matrices. The logarithm of co-occurrence of two words i and j , X_{ij}, in the corpus, is modeled using the following equation:

$$\log(X_{ij}) \approx u_i^T v_j + b_i^U + b_j^V \tag{2.1}$$

with $b_i^U, b_j^V \in \mathbb{R}$ being biases to dampen the influence of frequency on representations.

The higher the co-occurrence frequency between words is, the more similar their embeddings should be. Factorization with GloVe is performed using a stochastic gradient descent to simultaneously optimize U and V. Put together, the GloVe model can be summed up in the following equation:

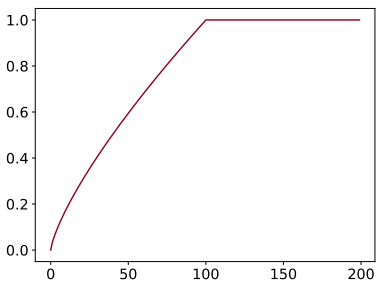


FIGURE 2.1: Effect of GloVe normalization applied to co-occurrences, $x_{\max} = 100$.

$$\mathcal{L} = \underset{U,V,b^U,b^V}{\operatorname{argmin}} \sum_{i=1}^{n}\sum_{j=1}^{n} f\left(X_{ij}\right)\left(u_i^\top v_j + b_i^U + b_j^V - \log\left(X_{ij}\right)\right)^2 \tag{2.2}$$

with:

$$f\left(X_{ij}\right) = \begin{cases} \left(\frac{X_{ij}}{x_{\max}}\right)^{\frac{3}{4}} & \text{if } X_{ij} < x_{\max} \\ 1 & \text{otherwise.} \end{cases} \tag{2.3}$$

$f\left(X_{ij}\right)$ has multiple purposes: it reduces the impact of large co-occurrences between terms by thresholding them at 1 (x_{max} is commonly set to 100), reduces the impact of low co-occurrences and also the time complexity to estimate parameters, particularly because the logarithm is undefined for zero.

`GloVe` was praised for its implementation made available by the authors and the simplicity with which embeddings could be extracted. Furthermore, they also provided pretrained models for *Wikipedia*, *Commoncrawl* and *Twitter* corpora [1].

[1] https://nlp.stanford.edu/projects/glove/

Just before `GloVe` was popularized, `Word2vec`, a word embedding model changed the landscape and really initiated the frenzy around distributional models.

▸ WORD2VEC

Mikolov et al. [Mik+13] introduced `Word2vec`, which soon became an essential method in NLP research but also in the industry. `Word2vec` is a clever blend of ideas in an easy-to-use framework, which allowed its rise as the most popular word embedding framework at the time. So much so that it sustainably influenced representation learning frameworks and lead to many adaptations such as `fastText` [Boj+17] or `Dict2Vec` [TGH17], and even beyond NLP as we present in Section 2.1.3.

[Mik+13] Mikolov et al. "Efficient estimation of word representations in vector space"

[Boj+17] Bojanowski et al., "Enriching Word Vectors with Subword Information"

[TGH17] Tissier et al., "Dict2vec: Learning word embeddings using lexical dictionaries"

`Word2vec` has two architectures : *Continuous Bags Of Words* (`CBOW`) and *Skip-Gram with Negative Sampling* (`SGNS`) which try to approximate word co-occurrence by the product of embeddings. `CBOW` models a word w_i given its context c ($p(w_i|c)$), on the other hand, `SGNS` models a word w_i given a word w_j ($p(w_i|w_j)$). To estimate these probabilities accurately, `Word2vec` uses large text corpora.

`CBOW`. The `CBOW` architecture of `Word2vec` models the probability of observing a word i given its context $c = w_{j_1}, w_{j_2}, \ldots$. This probability is parameterized by the vector representations of the context words c:

$$p\left(w_i|c;U,V\right) = \frac{\exp\left(u_i^\intercal \alpha_c\right)}{\sum_{i'=1}^{n}\exp\left(u_{i'}^\top \alpha_c\right)} \tag{2.4}$$

Here, $\alpha_c \in \mathbb{V}$ is the real-valued vector representing the context c, which is the average of the vectors representing context words c. Thus, this probability is recognized as a softmax function, which is the exponential of the target word representation by its context representation, normalized over the entire vocabulary.

`CBOW` requires a dataset D constructed by a sliding window approach over the corpus $\mathcal{C}$. This step is carried out by applying a sliding window

of dimension at most $2\ell + 1$ on corpus $\mathcal{C}$. This process yields both left and right contexts relative to the target word. The problem thus arises as a maximization problem of the likelihood of the dataset D under the assumption Equation (2.4).

$$\underset{U,V}{\operatorname{argmax}} \left(\prod_{(w_i,c)\in D} \frac{e^{u_i^\intercal \alpha_c}}{\sum_{i'=1}^{n} e^{u_{i'}^\intercal \alpha_c}} \right) \tag{2.5}$$

After taking the logarithm, this is equivalent to maximizing the following log-likelihood:

$$\begin{aligned} \mathcal{L} &= \underset{U,V}{\operatorname{argmax}} \left(\log \left(\prod_{(w_j,c)\in D} \frac{e^{u_j^\intercal \alpha_c}}{\sum_{i=1}^{n} e^{u_i^\intercal \alpha_c}} \right) \right) \\ &= \underset{U,V}{\operatorname{argmax}} \left(\sum_{(w_j,c)\in D} \log \left(\frac{e^{u_j^\intercal \alpha_c}}{\sum_{i=1}^{n} e^{u_i^\intercal \alpha_c}} \right) \right) \end{aligned} \tag{2.6}$$

`SGNS`. The goal of architecture `SGNS` is to model the conditional probability of observing a word j within a distance of at most ℓ from a word i. This probability is parameterized by the vector representations of the target (U) and context words (V):

$$p\left(w_j | w_i; U, V\right) = \frac{1}{1 + \exp\left(-u_i^\intercal v_j\right)} \tag{2.7}$$

This conditional probability is actually a sigmoid function σ applied to the dot product between the target vector of word i and the context vector of word j. Thus, the higher the dot product between the representations u_i and v_j of the words, *i.e.*, the more similar the two words are, the higher the conditional probability.

Matrices U and V are estimated through several steps. First, a dataset D is constructed with a sliding window like in `CBOW`. The next step involves maximizing the likelihood of the dataset D under the assumption $p(w_j|w_i; U, V) = \sigma\left(u_i' v_j\right)$. Negative sampling is introduced to make the model converge. Indeed, the absence of negative sampling leads to matrices U and V with high values that maximize the Cartesian product at the expense of words representations. This issue can thus be formulated as binary classification, *i.e.*, logistic regression between positive word pairs and negative word pairs. A dataset D' is constructed, consisting of positive samples $(w_i, w_j, \gamma_{i,j})$ with $\gamma_{i,j} \in \{-1, 1\}$ such that $\forall (w_i, w_j) \in D, (w_i, w_j, 1) \in D'$, and k negative samples $(w_i, w_j', -1) \in D\prime$ with w_j' randomly drawn from the vocabulary W. The log-likelihood of this dataset is then expressed as follows:

$$\underset{U,V}{\operatorname{argmax}} \sum_{(w_i,w_j)\in D'} \log\left(\sigma\left(\gamma_{ij} u_i^\top v_j\right)\right) \tag{2.8}$$

`Word2vec` is commonly implemented using a shallow two-layer neural network. However, it has been shown that `SGNS` is implicitly factorizing a word-context matrix [LG14]. Subsequently, there seems to be a kinship between all static models that we described, related to matrix factorization, even beyond the evident link created by the distributional hypothesis.

[LG14] Levy and Goldberg, "Neural Word Embedding as Implicit Matrix Factorization"

Dense discrete models (that provide one dense representation per word) share features.

Limits of discrete models

One limitation with discrete models is that there is only one vector per word, regardless of its polysemy. Let's consider, for example, the noun "*bank*", in some contexts it will be related to the institution with whom money is deposited, in other contexts it might indicate the side of a river. Soon, new word embedding models took this aspect of word sense into consideration with contextualized representations, which up to this day remain the preferred approach.

Towards large contextual models

Polysemous words exist and do not have the same sense depending on the context in which they appear. Providing a contextualized vector per word, allowed to discriminate between multiple senses of a word. We previously mentioned the polysemy of "*bank*" but words might have senses that are more subtle. Let us mention another example of words' sense influenced by context. In the two following sentences: "*She thanked mother nature.*", "*She thanked the Queen mother.*", although most of the context is the same, the sole change of "*nature*" to "*Queen*" gives an all other meaning to "*mother*". Contextual models appeared as an alternative to discrete models that provide one representation regardless of the context in which a word appears. The increasing popularity of neural networks also benefited to the rise of contextual neural methods. Since then, most methods have relied on shallow or deep neural networks to learn vectorized representations. `Elmo` [Pet+18] came first with a bi-LSTM architecture, followed by `BERT` [Dev+19] that kick-started the *Pre-Trained Language Models* (`PLMS`) frenzy. `BERT` is a self-supervised transformer architecture implementing what is called self-attention and consists of 340 million parameters. Transformers have first benefited from attention mechanisms [Vas+17]. Attention is a mechanism used in neural networks, particularly in models like Transformers, to selectively focus on different parts of input data. Regarding text, Attention allows the model to weigh the importance of each word or token in a sequence extracted from a corpus relatively to others. This is achieved by computing attention scores, which quantify the relevance of each token to every other token in the sequence. These scores are then used to compute a weighted representation of each token, with higher weights assigned to more relevant tokens. Self-attention thus allows contextualizing the representation of a word using its context. These models are trained by masking words in sentences extracted from large corpora, and trying to predict them from their contexts. `BERT` spawned a whole family of `PLMS` models, modifying hyperparameters and training objectives such as predicting the next sentence `RoBERTa` [Liu+19], specialized in a language like French with `CamemBERT` [Mar+20] or to a field like `clinicalBERT` [Als+19].

[Pet+18] Peters et al., "Deep Contextualized Word Representations"

[Dev+19] Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

[Vas+17] Vaswani et al., "Attention is all you need"

At the beginning of 2017, `PLMS` sizes were in the order of a hundred million parameters, which already required large quantities of training data [OSR19]. Subsequently, the major players in the field, namely *Google*, *Facebook*, *OpenAI*, *NVIDIA*, and *Microsoft* had a definite advantage in the race

[OSR19] Ortiz Suárez et al., "Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures"

towards large models with computing resources that could be allotted to training PLMS. Quickly, model sizes and amounts of data involved in training architectures increased. We moved towards what we call *Large Language Models* (LLM). The number of trainable parameters grew model after model, with GPT-3 counting 175 billion parameters in 2020 and most models in the order of tens to hundreds of billions of parameters[2]. However, large architectures allow drastic improvements in many areas of NLP, greater visibility, and industrial opportunities.

[2]It is worth noting that at the time this manuscript was written, a trend towards "*smaller*" LLM is underway. Models with fewer than ten billion parameters are being released (*Mistral 7B, Phi-2, Eagle 7B*) which are supposedly more manageable sizes.

In a world of LLM, traditional discrete models seem long gone, however, the quantity of training data required and the energy consumption of these architectures both at training and inference time remain a concern [SGM19; Sam+23]. First, in terms of sustainability regarding global warming. Second, because LLMs require enormous GPU resources, making them less suitable for small to medium-sized institutions [SPS20]. And third, neural networks, alike most dense embedding techniques are black boxes with whom it is complex to understand and explain an output solely from the model. We will develop on the limits of existing approaches Section 2.1.4, and specifically on the interpretability of word embeddings Chapter 5.

[SGM19] Strubell et al., "Energy and Policy Considerations for Deep Learning in NLP"

[Sam+23] Samsi et al., *From Words to Watts*

[SPS20] Sharir et al., "The Cost of Training NLP Models: A Concise Overview"

2.1.3 *Representing networks*

Networks are not exempt from the need for meaningful representations encompassing their topology. Indeed, understanding and exploring the data they model involves downstream tasks related to machine learning, *e.g.*, link prediction, feature prediction or node classification. To that end, in parallel to word embeddings in NLP, graph embedding methods have been developed to provide a representation tailored to data structured into graphs. We will mostly focus on representations of nodes, although, graphs, subgraphs, and edges can also be represented by embeddings [SCV19]. One such method is introduced for *link prediction* Section 3.2.1 with the goal of providing a representation of edges.

[SCV19] Sinha et al., "Systematic Biases in Link Prediction"

▸ FACTORIZATION WITH EIGENDECOMPOSITION

Graph embedding is similar to word embedding when it comes to the philosophy behind techniques to learn representations. Factorization methods were the first ones to be employed. Among popular factorization methods, many rely on eigenvectors or eigenvalues to reduce dimensions of a matrix capturing similarity between nodes. `IsoMap` [TSL00] leverages a neighborhood graph based on the geodesic distance between nodes. This geodesic distance is usually the sum of weights along the shortest path between nodes that can be computed using Dijkstra's algorithm. The embedding matrix is then obtained by taking the top n eigenvectors of the geodesic distance matrix. `Laplacian Eigenmap` [BN03] relies on the graph Laplacian matrix :

[TSL00] Tenenbaum et al., "A global geometric framework for nonlinear dimensionality reduction"

[BN03] Belkin and Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation"

$$\mathcal{L} = \mathcal{D} - \mathcal{A} \tag{2.9}$$

With $\mathcal{A}$ the adjacency matrix of the graph and $\mathcal{D} = \text{diag}(d_1, \cdots, d_n)$ the diagonal degree matrix. This intermediary representation is supposed to capture the topology around a node. The `Laplacian Eigenmap` embeddings are the eigenvectors of the Laplacian $\mathcal{L}$ matrix. Both `IsoMap` and `Lapla-`

cian Eigenmap have the goal of preserving *first-order* proximities, which are direct connections between nodes in a network.

Second-order proximity extends beyond direct connections to capture similarities between nodes based on their shared neighborhood structures. Two nodes may be similar if they share similar neighbors or neighbor structures even though they are not directly connected. Several factorization methods are aimed at preserving first and second-order proximities. LINE defines two objective functions that are optimized separately to keep close to one another in the latent space nodes that are connected and nodes that have similar neighborhoods. They optimize these two objectives using a stochastic gradient descent, and then concatenate the two representations for first and second-order proximities. GraRep goes a step further by trying to capture proximities up to a distance four. To that end, they build what they call *k-step* transition matrices, which are essentially modeling the probability of transitioning from a node u to a node v in at most k steps. Then, after applying *Singular Value Decomposition* (SVD) to each *k-step* transition matrix, they concatenate the representations obtained at these different orders that will form an embedding. *High-order Proximity preserved embedding* (HOPE) is also relying on SVD to factorize a similarity matrix with the goal of preserving high-order proximities in the output representation. The framework can be applied to many different similarity metrics that capture high-order proximities. The similarity metrics include: *Katz Index* (Definition 26), *Rooted PageRank*, *Common Neighbors* (Definition 21), and *Adamic Adar* (Definition 22). The methods we have described share a kinship with those described in the beginning of Section 2.1.2 in the sense that they factorize matrices using singular values and vectors. Yet, the kinship does not stop to eigendecomposition methods.

▸ RANDOM WALKS

Random walks are a useful tool for grasping and understanding the structure of a graph. Unsurprisingly, random walks have also been found useful to extract nodes' representations. Let us first define what a random walk is.

Definition 15 (Random walk)**.** A random walk $\mathcal{W}_{u_i}$ rooted at node u_i is a stochastic process resulting in a sequence $\mathcal{W}^0_{u_i}, \mathcal{W}^1_{u_i}, \cdots, \mathcal{W}^t_{u_i}$ such that each next time step $\mathcal{W}^{t+1}_{u_i}$ is chosen at random among the neighbors of node u_t.

From Definition 15, we understand that random walks record paths within a network and are thus useful to describe their topology. Interestingly, these random walks, that are sequences are quite similar to what sentences would be in texts. Except words in random walks are replaced by nodes, and the context of a node is its neighbors and the neighbors of its neighbors. Naturally, graph embedding using random walks imported methods first developed for sentences. Deepwalk, Walklets and node2vec rely on similar principles, they first extract random walks that act as sentences, but with nodes in place of words. Then, Word2vec's SGNS method can be applied to extract dense representations. Deepwalk [PAS14] proceeds to generate multiple truncated random walks of size t for each node. Walk-

[PAS14] Perozzi et al., "DeepWalk: Online Learning of Social Representations"

[Per+17] Perozzi et al., "Don't Walk, Skip! Online Learning of Multi-scale Network Embeddings"

[GL16] Grover and Leskovec, "node2vec: Scalable Feature Learning for Networks"

lets [Per+17] alters the way random walks are sampled. It introduces a *skip* step that skips nodes in random walks. This key alteration is supposed to allow capturing higher order relationships between nodes. node2vec is also leveraging random walks and SGNS but the sampling of random walks is different from Deepwalk and Walklets. node2vec [GL16] introduces philosophy from *Breadth First Search* (BFS) and *Depth First Search* (DFS) to sample sequences. Two parameters are introduced: p, the return parameter sets the probability to revisit a node. The higher it is, the farther from the root node the walk will likely explore. The lower it is, the more locally the walk will explore. The in-out parameter q can approximate the behavior of BFS if its value surpasses 1 and thus explore locally. If its value is lower than 1, the exploration will venture further away from the root node. The combination of these two parameters is supposed to interpolate between BFS and DFS to sample random walks and capture graph topology.

The similarity between node embedding and word embedding methods when it comes to random walk is straightforward, as the factorization algorithm to compute embeddings remains the same as Word2vec.

[BGV19] Brochier et al., "Global Vectors for Node Representations"

GloVe also has a graph counterpart under the name of *Global Vectors for Node Representation* (GVNR) [BGV19]. GloVe factorizes a word co-occurrence matrix, GVNR relies on a node co-occurrence matrix extracted from random walks. Its particularity is that distant co-occurrences are weighted down the further from each other nodes are. This allows discarding rare co-occurrences that are taken in consideration in GloVe. Furthermore, GVNR also includes negative samples similarly to SGNS to better consider non-co-occurrence.

With Deepwalk, Walklets, node2vec and GVNR, we see clearly the filiation between word and graph embedding methods. Algorithms are adapted to function with the data provided but, in essence, the techniques remain similar.

▸ NEURAL APPROACHES TO NETWORK EMBEDDING.
Similarity between models in NLP and for network embedding does not stop with Word2vec and GloVe. Graphs have also been subject to new architectures involving neural networks. Graph Neural Networks (GNN) methods have developed. As with LLM, attention mechanisms have also been introduced in GNN. To explain what attention brings to GNN, we first need to review what preceded its integration. Most GNN architectures have one objective, each node is aggregating the representations of its neighbors. By iteratively aggregating the representation from neighbors, representations are supposed to embed higher-order information than just their local neighborhood. *Graph Sample and Aggregation* (GraphSAGE) introduced by Hamilton et al. [HYL17] does just that. Although, aggregating all the representations from all neighboring nodes is not necessarily relevant and can be complex depending on the degree of a node. GraphSAGE thus introduces a sampling strategy before aggregation. GraphSAGE sampling can simply consist of randomly selecting a predefined number of neighbors and aggregating their representation, it can be biased based on degree or any other node property or similarity measure. If we summarize GraphSAGE, each node u aggregates information from its sampled neighborhood $N_S(u)$ to compute its own

[HYL17] Hamilton et al. "Inductive representation learning on large graphs"

representation e_u. This representation is generated by aggregating the embeddings of neighboring nodes using an aggregation function (aggregate), followed by an update function (σ) applied element-wise. The node representation e_u is computed as:

$$e_u = \sigma(\text{aggregate}(\{e_v, \forall v \in N_S(v)\})) \tag{2.10}$$

where e_v denotes the embedding of neighboring node v. `GraphSAGE` operates through multiple layers of aggregation to capture increasingly abstract representations of nodes in the graph.

The sampling strategy of `GraphSAGE` allowed speeding up embedding extraction. However, attention can bring more to `GNN` than a random sampling of neighbors representations. As attention allowed focusing on parts of the context in building word embeddings in NLP, in *Graph Attention Networks* (`GAT`), attention is used to focus on more informative neighbors and weighing down less relevant ones. `GAT` computes attention with a self-attention mechanism weighing the similarity of the updated node to its neighbors, thus sampling representations from of the most relevant neighbors.

Attention introduced in `GAT`, a `GNN` model is another evidence that there is a permeability of graph embedding methods to advances made in other areas of machine learning and particularly NLP. Algorithms and architectures shared across NLP and complex networks demonstrate the kinship between these domains, the incremental aspect of representation learning methods, and of artificial intelligence in general. However, similarly to `LLM`, `GNN` methods require GPU resources and fine-tuning to obtain a well-performing model. The addition of attention parameters complexifies training and increases time complexity.

The fact that attention has made its way into `GNN` further reinforces the evidence of a permeability of network methods to advances made in other areas of machine learning. The race towards large neural architectures means that compute time, and energy consumption are also on the rise. Moreover, representation learning algorithms do not provide interpretable, auditable models from which interpretations related to the structure of the latent space can be made. If we look further, decisions made down the line in a downstream task setting are hard to explain if the first element of the pipeline, *i.e.*, the representation is not interpretable. We will touch on the limits of these models in Section 2.1.4 and specifics of interpretability Chapter 5.

2.1.4 *Shortcomings of network and word embedding*

As we have seen, embedding methods share a kinship even across domains (see Table 2.1). Algorithms remain relevant for different types of data with small adaptations to the input structures provided. Subsequently, their main shortcomings are also shared. The first and most obvious inconvenience is the complexity of methods with regard to compute time, resources required, and amounts of data required to extract a representation. This comes evidently with the emergence of `LLM` and their large architectures (see Figure 1). Foremost, because training large architectures requires numerous GPUs for extended periods of time. Such hardware resources consume large

		Word Embedding	Node Embedding
Static	Eigendecomposition	• LSA • GLSA	• IsoMap • Laplacian Eigenmap • LINE • GraRep • HOPE
	SGNS	• Word2vec • fastText	• Deepwalk • Walklets • node2vec
	GloVe objective	• GloVe	• GVNR
Contextual	Attention	• BERT • RoBERTa • CamemBERT • GPT-3	• GAT

TABLE 2.1: Summary of kinship relations between word and graph embedding methods.

[SGM19] Strubell et al. "Energy and Policy Considerations for Deep Learning in NLP"

quantities of electricity. Strubell et al. [SGM19] back in 2019 drew attention towards the CO_2 cost of transformer architectures, before LLM became legion. Computing the carbon footprint of AI became central [LGI21] resulting in guidelines to gaining efficiency in implementations and compute infrastructure [Pat+21]. This is partly due to the terabytes of training data ingested by LLM.

Although training consumes significant amounts of energy, inference also comes at a great energy cost. In some cases, this can be equivalent to the electricity consumption of small European countries such as Croatia or Ireland [Sam+23; De 23]. Orders of magnitude in compute time are not comparable with Word2vec or HOPE, we will see Chapters 3 and 4 that theirs is still not negligible, and should be considered.

[Sam+23] Samsi et al., *From Words to Watts*

[De 23] De Vries, "The growing energy footprint of artificial intelligence"

What is common to all approaches we have so far presented, regardless whether the goal is to obtain node or word embeddings, is the lack of interpretability in the latent spaces derived. That is, being able to understand how an embedding space is structured, what contributes to the representation of each item and derive it solely from the latent space extracted. We dive into interpretability Chapter 5 and touch on the difference between interpretable and explainable models. Interpretability is desirable for word embedding as a way to understand how word sense is composed. Dense methods such as Word2vec and GloVe provide vector spaces in the order of 300 to 500 dimensions, their dimensions are entangled. By entangled, we mean that dimensions contribute to multiple aspects of the representation. As a consequence, it is hard, if not impossible, to interpret the contribution of each dimension of the embedding space to the representation of an item. LLMs can provide contextual representations that change for the same word appearing in different contexts. Subsequently, most attempts at explaining such models rely on applying post-hoc models that try to explain the contribution of features to a representation or to the decision of a classifier [RKR20; SMF18]. Post-hoc interpretability is a step in the right direction, but the base model remains intrinsically a black box. Furthermore, if a post-hoc explainable model needs to be trained on top of the base model, it further contributes to the carbon footprint of approaches.

[RKR20] Rogers et al., "A Primer in BERTology: What We Know About How BERT Works"

[SMF18] Shin et al., "Interpreting word embeddings with eigenvector analysis"

Explainability and interpretability have been integrated into policies at the European level (GDPR) and in the French law and are desirable for algorithmic processing in high-stake domains (medicine, law, credit scoring). We will see Chapter 5 that a sparse embedding space, with a few activated dimensions per item can provide disentangled dimensions and interpretability.

Neural architectures like LLM in NLP and GNN in complex networks represent a leap forward for both domains. They allow progress on downstream tasks but also raise concerns regarding their sustainability and "black box" nature. To address these concerns, we work towards more sustainable and interpretable representations while preserving the jump in performances enabled by large neural architectures. As a matter of fact, a motivating example is that input embeddings are the first layer of LLM. If we work towards efficiently obtaining interpretable representations, it is a step forward towards better LLM that are in tune with contemporary concerns.

2.2 BUILDING INTERPRETABLE REPRESENTATIONS

Most embedding methods lack interpretability. Yet, it is essential for sensitive application of AI. Biased representations might lead to biased downstream predictions that cause harm based on protected characteristics of individuals such as age, gender, race including color, nationality, ethnic or national origin, *etc.* With the rise of AI as decision-support systems in areas as sensitive as the justice system [DF18; Wan+19], interpretability can be beneficial to auditing decisions and uncovering biases stemming from data, in compliance with policies introduced by governance bodies such as the European Union.

[DF18] Dressel and Farid, "The accuracy, fairness, and limits of predicting recidivism"

[Wan+19] Wang et al., "An Empirical Study on Learning Fairness Metrics for COMPAS Data with Human Supervision"

Another unrelated advantage of interpretability is the ability to observe how a representation is composed, which dimensions of the space encode a feature that is relevant to a group of items. What the conjunction of these dimensions simultaneously tells about the data they contribute to represent.

Yet, some interpretable methods add a non-negligible overhead by disentangling dimensions from dense models. Thus, while they provide interpretability, they do so without improving efficiency. We will see how interpretability and frugality can be combined into a method.

For all these reasons, we introduce an interpretable embedding method able to provide vectors whose dimensions can be explored and from which we can gain insight into the latent space's structure with frugality as a primary objective. To get across the philosophy behind the model and the broad objective, we will start with a visual introduction leveraging visualizations on an airport network of domestic flights within the U.S.A.

2.2.1 *Use case of an airport network*

To present the philosophy behind the framework called *Lower Dimension Bipartite Graph Framework* (LDBGF) that we will more formally introduce in Section 2.2.2, we first consider a use case based on an airport network of domestic flights in the United States of America. From a graph standpoint, we can represent the network of airports in multiple ways. The simplest is to

connect any two airports between which a direct route exists. The graph can be weighted in numerous manners according to the number of daily flights, the number of passengers or the distance between airports. For the sake of simplicity and because most plane routes are bidirectional, let us consider an undirected and unweighted graph $G = (V, E)$ of U.S. airports with V the set of nodes representing the airports, E the set of edges representing the existence of a flight between two airports. Graph G is drawn over a map of the U.S.A. in Figure 2.2 and shows the sheer number of domestic connections between airports. Our goal is to derive a representation that is able to embed how an airport is connected.

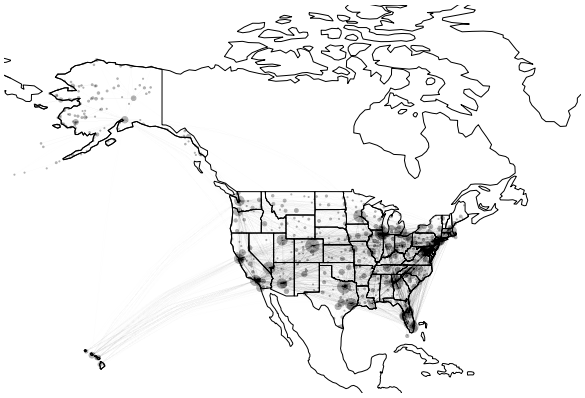
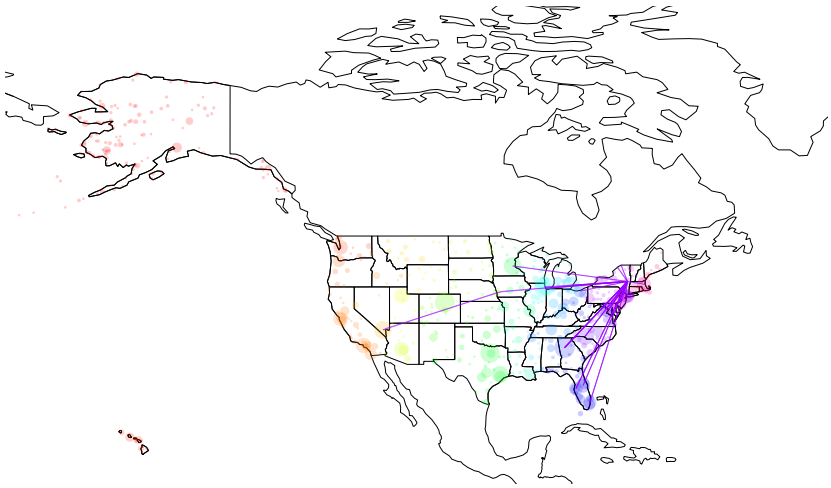


FIGURE 2.2: An airport network of the United States of America (size of nodes proportional to their degrees, number of routes).

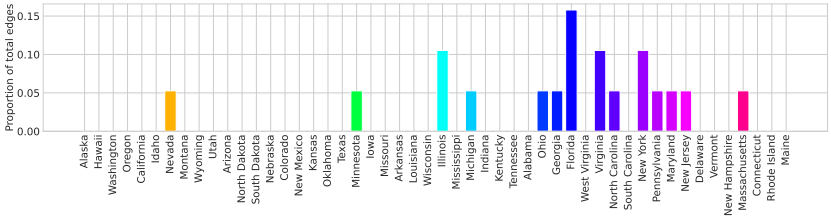
We assume that there is a hierarchy among airports: for instance, international airports act as hubs to smaller, mostly domestic airports. For example, if you wish to fly out of *Santa Fe Regional Airport* (SAF), New Mexico to *Harry Reid International Airport* (LAS), Las Vegas, Nevada, chances are you are going to transit through *Phoenix Sky Harbor International Airport* (PHX), Arizona. Relying solely on a visual representation is intricate, as the underlying hierarchy is challenging to highlight. It is thus laborious to distinguish the busiest airports from those having fewer inbound and outbound flights. However, we wish to encapsulate more than just connectivity between airports, namely the spatial structure of the network, and how flights between airports connect states. To that end, let us cluster together the airports of the network based on the state they are located in. By doing so, we obtain fifty groups of airports, each of those corresponding to a U.S. state. The question is now: how can one derive a **visual representation** of each airport in the network that encompasses its medium haul (domestic) connectivity as well as its local (state-level) neighborhood?

The solution lies with the state partition we produced: by considering the connectivity of each airport to each state instead of one-to-one connection, we can encapsulate local and broader patterns of connectivity. More precisely, we quantify the strength of connectivity of an airport to a state by considering the proportion of airports reached in that state over the total number of airports that are served. We thus obtain a visual representation that displays the pattern of connectivity of each airport. The higher the value for a state, the stronger the connection of the airport.

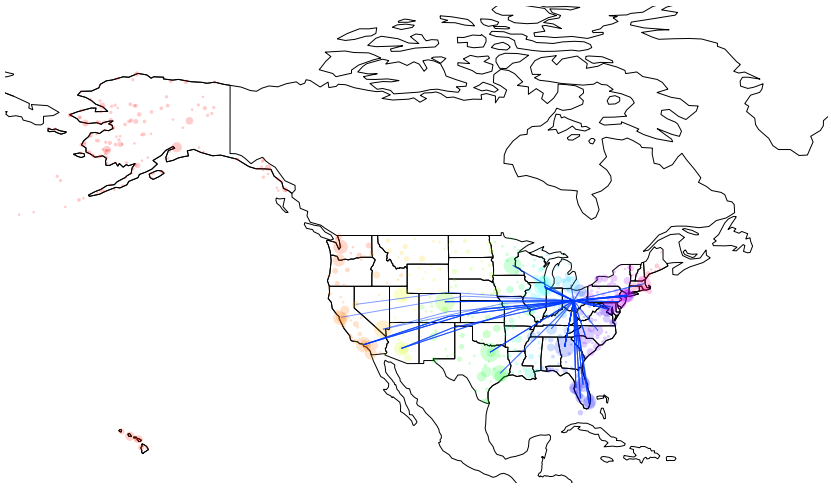
Let us demonstrate that through two examples. We can first consider



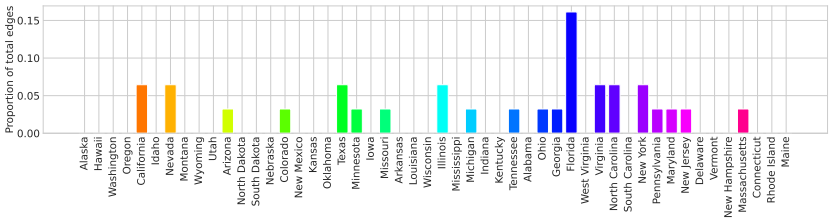
(a) Flights connecting Albany (ALB) to other US airports.



(b) Distribution of the connectivity of Albany International Airport (ALB) to US states.



(c) Flights connecting Columbus (CMH) to other US airports.



(d) Distribution of the connectivity of Columbus Airport (CMH) to US states.

FIGURE 2.3: Flights connected to Albany and Columbus and distribution of connections towards states.

two airports on the east coast of the U.S.A.: *Albany International Airport* (ALB), NY and *John Glenn International Airport* (CMH) in Columbus, IL. Connections between ALB and CMH are presented Figures 2.3(a) and 2.3(c). These two airports do not play a major role in their respective states of New York and Illinois but a role with neighboring states. The distribution of these airports over the states Figures 2.3(b) and 2.3(d) seems similar, mostly to airports in the northeast, midwest, and south of the country. On top of the connectivity to each individual state, the color gradient reveals another level of hierarchy related to the different regions of the west, southwest, midwest, southeast, and northeast. From the distribution of connectivity Figures 2.3(a) and 2.3(c), ALB and CMH are primarily connected to airports in the east to midwest region.

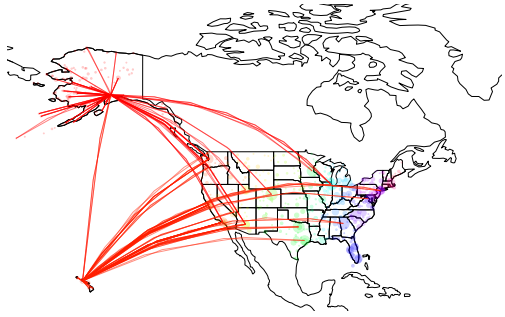
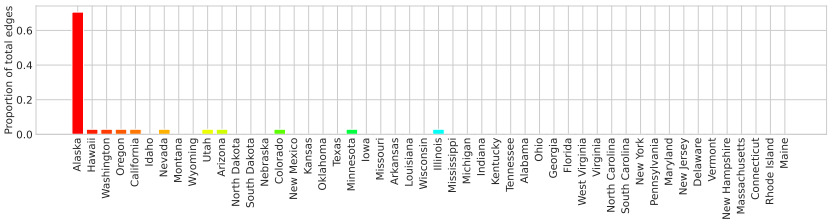
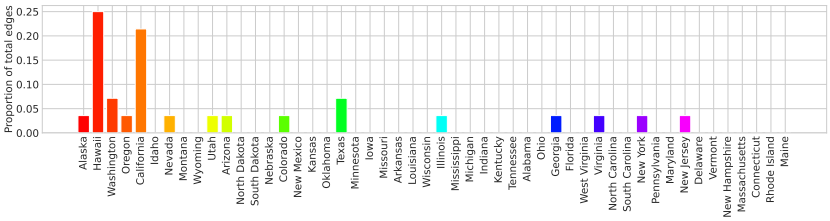


FIGURE 2.4: Flights connecting Anchorage and Honolulu to regional and mainland airports.

We also consider the representation of two peripheral airports, *Ted Stevens Anchorage International Airport* (ANC) in Alaska and *Daniel K Inouye International Airport* (HNL), in Hawaii. These airports should, according to our representation Figure 2.4, exhibit a connection to similar states, mostly in the west of the country, that act as gateways to these territories. Plotting the distribution over the states Figure 2.5 confirms this hypothesis, ANC has its highest value related to the state of Alaska, as does HNL for the state of



(a) Distribution of the connectivity of Anchorage Airport (ANC) to US states.



(b) Distribution of the connectivity of Honolulu Airport (HNL) to US states.

FIGURE 2.5: Distribution of connections for Anchorage Airport (ANC) and Honolulu Airport (HNL).

Hawaii. Since we are measuring the distribution of connections over the partition of states, and because these two airports play a major role in connecting local airports to the mainland, the bar related to their respective states has the highest value. Nevertheless, our representation also includes information about connections to the West and Midwest of the country and, in the case of HNL, connections to airports in the east.

EXTRACTING VECTORS. These visual representations are in fact embedding vectors and this pipeline can derive node embeddings that are interpretable by design. Each component of the vector space constructed is related to a state where airports are located. Indeed, the representation is a measure of the strength of connection between an airport and each state, thus providing a representation in lower dimension than that of the full network, over a tangible structure, the partition of airports grouped by state. Moreover, vectors are sparse as not every airport is connected to every state. These embeddings are inexpensive to compute and produce visually interpretable vectors. The question we will attempt to answer is the following: how can such a pipeline be applied to various kinds of networks to produce sparse, interpretable embeddings adapted to many types of data?

The representation of the nodes from the airport network is achieved through the use of an intermediary grouping of the nodes. By grouping nodes into clusters, the graph could be represented in a bipartite manner and the connection of bottom nodes to their top nodes used as a representation. Based on this idea, we detail our node embedding framework Section 2.2.2. Grouping similar nodes in clusters is at the heart of the `LDBGF` philosophy we introduce and also of its two implementations: `SINr-NR` and `SINr-MF`, that produce sparse and interpretable node representations.

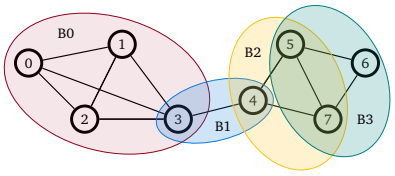
2.2.2 *LDBGF: vectors from bipartite projection*

The Lower Dimension Bipartite Graph Framework (`LDBGF`) aims at producing an accurate yet compressed representation of a network. The problem we formulate here is: how can we project a graph $G = (V, E)$ with an adjacency matrix $\mathcal{A}$ in a bipartite form to a obtain graph $G' = (\top, \bot, E')$ and its adjacency matrix $\mathcal{A}'$ which can be projected back to one mode G? Ideally, $\mathcal{A}'$ would be in lower dimension than $\mathcal{A}$ and each node $\bot$ could be represented using its connection to $\top$ nodes.

`LDBGF` benefits from the observation made by Guillaume and Latapy [GL06] that all networks can be represented in a bipartite mode. This projection can be performed using cliques and refers to an EDGE CLIQUE COVER problem. Namely, finding a minimal set of cliques to cover all graph edges. Figure 2.6 demonstrates the process over a toy graph. From a unipartite graph $G = (V, E)$ with an adjacency matrix $\mathcal{A}$, `LDBGF` finds the minimum number of cliques to cover all the graph edges. Based on the cliques uncovered, the graph can be projected in a bipartite form $G' = (\top, \bot, E')$ by linking nodes to the cliques they are part of. Since the minimum number of cliques to cover all edges is lower than the number of nodes, the bipartite representation G' Figure 2.6(c) admits an adjacency matrix $\mathcal{A}'$ in lower dimension than $\mathcal{A}$. Projecting back G' to a unipartite graph yields G. The

[GL06] Guillaume and Latapy, "Bipartite Graphs as Models of Complex Networks"

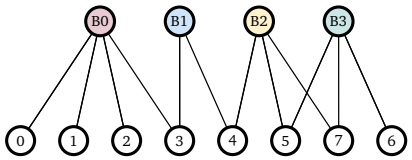
dimension reduction operated by finding a set of cliques to cover G is thus without loss of information. Furthermore, it provides embeddings with a direct and tangible link to the graph structure: cliques represented by the $\top$-nodes.



(a) A graph $G = (V, E)$ with 8 vertices and 12 edges. Edge clique covering of G with 4 cliques.

$\mathcal{A}$	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	0	1	1	1	0	0	0	0
1	1	0	1	1	0	0	0	0
2	1	1	0	1	0	0	0	0
3	1	1	1	0	1	0	0	0
4	0	0	0	1	0	1	0	1
5	0	0	0	0	1	0	1	1
6	0	0	0	0	0	1	0	1
7	0	0	0	0	1	1	1	0

(b) The adjacency matrix $\mathcal{A}$ of G. The dimension of the vector for each vertex is $|V|$ (8).



(c) Projection of G into bipartite graph $G' = (\top, \bot, E')$ with the minimum number of cliques.

$\mathcal{A}'$	*B0*	*B1*	*B2*	*B3*
0	1	0	0	0
1	1	0	0	0
2	1	0	0	0
3	1	1	0	0
4	0	1	1	0
5	0	0	1	1
6	0	0	0	1
7	0	0	1	1

(d) Adjacency matrix of bipartite graph G'. The dimension of the vector for each vertex is the number of cliques (4).

FIGURE 2.6: Illustration of the LDBGF, vertices are linked to the cliques they belong to.

However, finding and approximating the set of cliques to cover the edges (EDGE CLIQUE COVER) is NP-Hard [Kar72]. In the example presented, the number of cliques to cover all edges is lower than the number of nodes, allowing to compress the representation of nodes via the bipartite projection. Yet, it is not always the case. The upper bound for the number of cliques in a graph was estimated by Erdös et al. [EGP66] to $\min(m, \frac{n^2}{4})$ which sounds ridiculously high. Based on this estimation, finding the minimum set of cliques to cover all edges is no mean feat.

[Kar72] Karp, "Reducibility among Combinatorial Problems"

[EGP66] Erdös et al. "The Representation of a Graph by Set Intersections"

Because of the NP-Hard aspect of EDGE CLIQUE COVER, learning embedding models that need less compute with LDBGF is unlikely. Yet, we wish to provide a representation learning algorithm, analogous to LDBGF, able to compress information via a bipartite projection but with a lower computational complexity. Furthermore, the $\top$-nodes will help gain interpretability of the representation, thus satisfying the two objectives of low compute and interpretability.

Communities seem to be the ideal structure to replace cliques in the $\top$-part of the bipartite graph. Many community detection algorithms have been developed to uncover dense groups of nodes that are more connected together than with the rest of the network. Furthermore, community detection can be performed very efficiently using heuristics. We detail the process of detecting communities in Section 2.3. We will first question why commu-

nity detection is useful in Section 2.3.1 before detailing how some community detection algorithms can partition nodes and the challenges they sometimes face. Finally, we introduce in Sections 2.3.3 and 2.3.4 our two implementations of `LDBGF` based on communities.

2.3 COMMUNITY DETECTION, A SOLUTION TO LDBGF?

2.3.1 *Community detection: principles and usage*

In the real world, communities are usually groups of people connected by social structures such as families, schools, athletic clubs, friendship circles, towns, nations, *etc.* Communities also exist on the internet in the form of forums on Reddit, for example, where users are gathered around common interests. These groups where items or individuals interact with each other shape the graph in such a way that groups of nodes are more densely connected to one another than with the rest of the network. The presence of these groups within the network is what we call a *community structure* which is composed of multiple *communities*. Early on, Girvan and Newman [GN02] define communities as: "*subsets of vertices within which vertex-vertex connections are dense, but between which connections are less dense*". Detecting such communities in networks is analogous to a clustering task, except the topology of the graph and the connections between nodes can be exploited. Community detection is unsupervised and has been applied to many types of networks across varied domains in complex systems. We now introduce the notion of partition and how to measure its quality, before reviewing community detection methods and detailing the main algorithms employed in `SINr`.

[GN02] Girvan and Newman "Community structure in social and biological networks"

▸ METHODS TO DETECT COMMUNITIES

The idea of uncovering groups in complex systems appears as early as 1927 in Rice [Ric27] who attempted to group people based on their voting patterns. Later on, [Hom50] came up with the intuition of uncovering social groups by rearranging a matrix of their social ties into a diagonal block matrix. In 1955, Weiss and Jacobson [WJ55] uncovered communities in the form of work groups in a government agency network. In 2002, Girvan and Newman [GN02] introduce a graph partition method based on edge betweenness.

[Ric27] Rice "The Identification of Blocs in Small Political Bodies"

[Hom50] Homans, *The human group*

[WJ55] Weiss and Jacobson "A Method for the Analysis of the Structure of Complex Organizations"

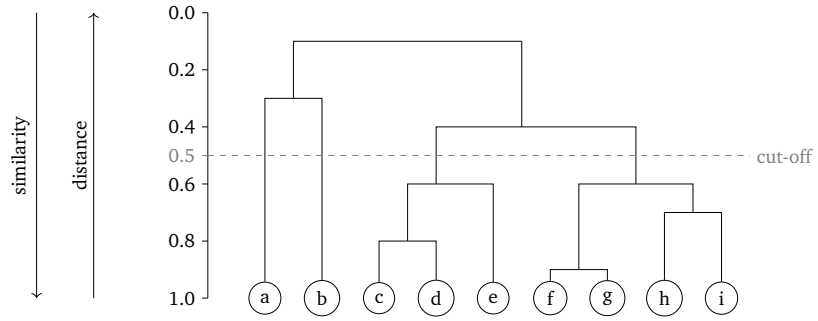


FIGURE 2.7: Example of a small dendrogram. Nodes are represented at the bottom, the hierarchical tree shows at which similarity S_{ij} the nodes join together.

Definition 16 (Edge Betweenness Centrality)**.** Betweenness centrality measures the fraction of shortest paths for all pairs of nodes that pass through an edge $e \in E$, $\sigma(u, v)$ is the number of (u, v)-paths, and $\sigma(u, v \mid e)$ is the number of those paths passing through edge e.

$$c_B(e) = \sum_{u,v \in V} \frac{\sigma(u, v \mid e)}{\sigma(u, v)} \tag{2.11}$$

[GN02] Girvan and Newman "Community structure in social and biological networks"

Girvan and Newman [GN02]'s algorithm iteratively removes edges which have the lowest *Edge Betweenness Centrality* until none remains. This allows to iteratively construct a dendrogram like in Figure 2.7 and cut-off at a certain level of the tree to obtain partitions.

We identify several families of approaches able to detect communities. We describe the broad principles of each approach in the next few paragraphs, starting with spectral methods that also employ clustering to some extent. Most of these algorithms are designed to scale up on large networks and have a feasible time complexity that makes them good contenders to supplant the EDGE CLIQUE COVER that would ideally describe the graph in `LDBGF`.

SPECTRAL METHODS. Most commonly relying on eigenvectors of matrices, spectral methods have been employed in graph partitioning [DH73; Fie73]. Spectral methods usually make use of the graph Laplacian (Equation (2.9), or a normalized variant) in combination with a clustering algorithm to partition nodes based on their eigenvalues. Other matrices have been considered to apply spectral clustering, such as the modularity matrix or the adjacency matrix [NG04; New13]. However, computing the exact eigenvectors of a Laplacian matrix comes with an $\mathcal{O}(n^3)$ complexity rendering it unfeasible for large networks. Fortunately, approximate methods can alleviate this problem [GV13].

[NG04] Newman and Girvan, "Finding and evaluating community structure in networks"

[New13] Newman, "Spectral methods for network community detection and graph partitioning"

STATISTICAL INFERENCE. Approaches grounded in statistical inference typically try to fit a statistical model to the topology of a graph. One of the widely recognized generative models applied to networks involving communities is the Stochastic Block Model (SBM). SBM involves optimizing the log-likelihood of communities within the provided graph [CL14]. SBM requires prior knowledge of the number of communities, a parameter often unknown in real-world networks. Furthermore, fitting the distribution of large networks is a challenge that results in significant time complexity [Kum+20]. `OSLOM` (Order Statistics Local Optimization Method) [Lan+11] is a community detection method that incorporates statistical principles. `OSLOM` is not a generative model in the sense of SBM, but it employs statistical significance measures to assess the quality and significance of communities detected within a network. `OSLOM` evaluates the likelihood of obtaining the same or a better partition in a randomized or null model. This assessment allows determining whether the communities uncovered are significant or have arisen by chance. It is able to uncover hierarchical structure of communities even on large networks but with a longer runtime than `Louvain` [DP22].

[CL14] Côme and Latouche, *Model selection and clustering in stochastic block models with the exact integrated complete data likelihood*

[Lan+11] Lancichinetti et al., "Finding statistically significant communities in networks"

[DP22] Dugué and Perez, "Direction matters in complex networks: A theoretical and applied study for greedy modularity optimization"

NETWORK DYNAMICS. Network dynamics such as diffusion, random walks and spin dynamics can help detect communities in networks. The `Label Propagation` algorithm introduced in Raghavan et al. [RAK07] uses diffusion to partition nodes in communities. The process of `Label Propagation` is simple: each node is initialized with a label, and during iterations, labels are propagated throughout the network. Each node adopts the majority label among its neighbors. Nodes with the same label form a community.

[RAK07] Raghavan et al. "Near linear time algorithm to detect community structures in large-scale networks"

The `Label Propagation` algorithm can lead to numerous partitions from the same initial state, as the final partition is an aggregation of intermediate solutions and random iterations over the nodes. This algorithm requires no prior knowledge about the network. Most communities are formed by the second or third iteration [RAK07]. However, on weighted networks like co-occurrence networks, the `Label Propagation` algorithm is sensitive to the weights applied to the links. Consequently, the partition can become degenerate with either an excessive number of communities or a single community containing all nodes[3].

[3] Section 2.3.1 details how reweighting edges can help with partitioning nodes with `Label Propagation` when a single partition is obtained.

Random walks (Definition 15) are another type of dynamic process over a network that may be used to detect communities. The `Walktrap` [PL05] method relies on the idea that nodes within the same communities are likely connected by short random walks. On the contrary, nodes in different communities are less likely to be connected by such walks. Thus, random walks tend to be "trapped" into densely connected parts of a network which may be a community. The algorithm performs random walks on the graph by randomly following edges. These random walks allow computing a similarity matrix based on the number of common nodes visited by random walks. From this similarity matrix, `Walktrap` can then apply a hierarchical clustering approach and build a dendrogram as presented Figure 2.7. The worst-case time complexity of `Walktrap` is $\mathcal{O}(mn^2)$ but generally closer to $\mathcal{O}(n^2 \log n)$ when networks are sparse and the height of the dendrogram remains relatively small in most cases [PL05].

[PL05] Pons and Latapy, "Computing Communities in Large Networks Using Random Walks"

OPTIMIZATION METHODS. `Infomap` optimizes a quality metric called the *Map Equation*. The *Map Equation* measures how much information is needed to describe a random walk process over a network, given a partition of the nodes into communities. Random walks are encoded with the Huffman Code [Huf52], like in language with words, short codes are used for frequently visited nodes and long codes for less frequently nodes. After this initial encoding step, nodes are grouped together in communities that best compress information about network structure. Based on the description length required to describe a community, the algorithm assesses the quality of the partition. Shorter description lengths indicate more meaningful communities that capture information flow. Afterward, the iteration process continues and nodes are moved between communities in an attempt to reduce the description length of the communities. This allows to construct a hierarchical structure iteratively, which reveals different granularities of community partitions. Another criteria that can be used to estimate the quality of a partition is the modularity.

[Huf52] Huffman, "A Method for the Construction of Minimum-Redundancy Codes"

Estimating the quality of a partition with the modularity (Equation (1.3)) opened for new methods using them as objective functions. First, as a way

[NG04] Newman and Girvan, "Finding and evaluating community structure in networks"

[Blo+08] Blondel et al. "Fast unfolding of communities in large networks"

to decide where to cut the dendrogram [NG04] to extract the optimal partition in [NG04]. In 2008, Blondel et al. [Blo+08] introduce `Louvain`, a community detection algorithm aiming to optimize modularity introduced Equation (1.3). The `Louvain` method consists of the following steps for a network $G = (V, E, \Omega)$:

(i) `Louvain` starts with a partition where all nodes are singleton, meaning that they each belong to a community of which they are the sole member.

(ii) Iteratively, from a random node u, `Louvain` computes the gain in modularity ΔQ (Equation (2.12)) that results from removing node u from its community and placing it in each of the communities of its neighbors. The node u is placed in the community for which the modularity gain is maximal (if any, otherwise it remains in its current community). The gain in modularity obtained by moving a node u into a community C can be computed with the following formula:

$$\Delta Q = \left[\frac{\sum_{\text{in}} + 2d_{\text{in}}(u)}{2m} - \left(\frac{\sum_{\text{tot}} + d(u)}{2m}\right)^2\right] - \left[\frac{\sum_{\text{in}}}{2m} - \left(\frac{\sum_{\text{tot}}}{2m}\right)^2 - \left(\frac{d(u)}{2m}\right)^2\right] \tag{2.12}$$

where $\sum_{in} = \frac{1}{2}\sum_{u,v\in C_i}\Omega(u,v)$ is the sum of weight of edges in $\mathcal{C}$, $\sum_{tot} = \sum_{\substack{u\in C\\ v\in C^{\complement}}}\Omega(u,v) + \sum_{\text{in}}$ the total sum of edges incident to nodes in C_i, $d(u)$ is the weighted degree of u, and $d_{in}(u)$ the weighted degree of u restricted to edges in C and m is the sum of all edges weights in the network.

(iii) The next phase of `Louvain` is to build a meta-graph where communities formed in the previous step are represented by a single node. Weight of the links between newly formed nodes are given by summing the weight of edges between nodes of the two communities. Intra-community links are represented by a self-loop over the node representing the community.

Steps (ii) and (iii) are repeated until convergence, that is, until the maximum modularity is reached and no more node changes are possible. Each iteration results in a partition $\mathcal{C} = \{C_1, \cdots, C_i\}$ of the nodes into i communities. By repeated iterations, `Louvain` provides a hierarchical structure of partitions of different resolutions, from the singleton partition to a partition that optimizes modularity according to the initialization of the algorithm. With two simple steps, `Louvain` is able to uncover partitions of good quality with a shorter runtime than most other methods [Blo+08].

Although very efficient, `Louvain` suffers from the resolution limit related to the modularity optimization that is employed. Indeed, the resolution limit is such that communities smaller than a certain size might be hard to uncover by optimizing modularity and can lead to badly connected communities–communities might be internally disconnected [FB07].

[FB07] Fortunato and Barthélemy, "Resolution limit in community detection"

To counter the resolution limit, the modularity can be parameterized with multi-resolution parameter γ[4]. The γ parameter [RB06] acts as a bias to increase the weight associated to the probability of the two nodes being connected based solely on their degree. This allows to parameterize the `Louvain` algorithm and control the granularity of communities, beyond exploiting the hierarchical partitions provided by the algorithm.

For all these reasons, we have chosen to implement `Louvain` in our community-based implementation of *Lower Dimension Bipartite Graph Framework* (`LDBGF`) that we introduced in Section 2.2.2. We now introduce the two implementations within `LDBGF` and the incremental process that allowed us to provide interpretable node representations with a low compute.

[4] Equation (1.3):

$$Q_{\mathcal{C}} = \frac{1}{2m} \cdot \sum_{i,j} \left[A_{ij} - \gamma \frac{d(i) \cdot d(j)}{2m} \right] \cdot \delta(C_i, C_j)$$

[RB06] Reichardt and Bornholdt, "Statistical mechanics of community detection"

2.3.2 *The road to community detection in SINr*

The *Lower Dimension Bipartite Graph Framework* (`LDBGF`) we introduced Section 2.2.2 can theoretically compress graph information about nodes in a lower dimension. However, the NP-Hard nature of the EDGE CLIQUE COVER problem prevents its implementation within our constraint of efficiency. To circumvent this complexity issue, we propose implementations that fit within `LDBGF`, but replace cliques with a substructure of the network that can be uncovered rapidly: communities. We present in the following paragraphs the main milestones that shaped our node embedding algorithm, `SINr`, from an unstable proof of concept to a full-fledged node and word embedding framework.

▸ THE GENESIS OF `SINr`

Palla et al. [Pal+05] observed that community information in word co-occurrence graphs conveys semantic information. Indeed, words that appear together in communities are densely connected, which means that they appear frequently together in text. This observation echoes with Firth's remark: "You shall know a word by the company it keeps" [Fir57], communities and their organization should thus be useful to represent word sense.

[Pal+05] Palla et al. "Uncovering the overlapping community structure of complex networks in nature and society"

[Fir57] Firth, "A synopsis of linguistic theory 1930-55."

Chen et al. [CZG08] introduced a document clustering approach based on the detection of thematic communities in keyword co-occurrence network. These word sense communities, are then leveraged, associating web pages to the most relevant thematic cluster, subsequently allowing their classification.

[CZG08] Chen et al. "An Unsupervised Approach to Cluster Web Search Results Based on Word Sense Communities"

In direct filiation with the method introduced by Chen et al. [CZG08], Connes and Dugué [CD19] introduced a method, able from a word co-occurrence network, to provide word embeddings based on the community membership of each word (represented by a node in the network). The objectives of this first iteration were the same we still have today for `SINr`: efficiency in compute, good performances regarding classical evaluation tasks, and interpretability. The objective is to project the adjacency matrix $\mathcal{A}_{n \times n}$ into a lower-dimension space $U_{n \times |\mathcal{C}|}$, $\mathcal{C}$ being the partition of nodes in communities. Their proof of concept focused on word co-occurrence networks, and specifically the Google books n-gram corpus [GO13].

[CD19] Connes and Dugué, "Apprentissage de plongements lexicaux par une approche réseaux complexes"

[GO13] Goldberg and Orwant, "A Dataset of Syntactic-Ngrams over Time from a Very Large Corpus of English Books"

A co-occurrence network is constructed by connecting nodes representing words if they co-occur together in a sentence from the corpus, edges

are weighted by the number of times two words appear simultaneously. However, proceeding that way leads to many words being connected, even though they rarely appear together. Yet, a network with many irrelevant connections renders community detection more complex and potentially yields low-quality partitions. Preprocessing text to remove less relevant co-occurrences is recommended both in methods on word embeddings [LGD15], but also to better detect communities [Yan+18] and can be performed by thresholding co-occurrence with the *Positive Pointwise Mutual Information* (PPMI, Definition 17).

[Yan+18] Yan et al., "Weight thresholding on complex networks"

Definition 17 (Positive Pointwise Mutual Information)**.** In the context of co-occurrences between words, the Positive Pointwise Mutual Information (PPMI) measures the probability of two words co-occurring over the probability that they should appear by chance. The Positive PMI removes excludes all negative PMI values. Given two words $w_u, w_v \in L$, the lexicon of the corpus, with $cooc(w_u, w_v)$ the number of co-occurrences between w_u and w_v, $occ(w_u)$ the number of occurrences of w_u:

$$p(w_u, w_v) = \frac{cooc(w_u, w_v)}{\sum_{w_i \in L} \sum_{w_j \in L} cooc(w_i, w_j)} \tag{2.13}$$

$$p(w_u) = \frac{occ(w_u)}{\sum_{w_i \in L} occ(w_i)} \tag{2.14}$$

$$\mathcal{A}_{(u,v)} = \begin{cases} 0, \text{ if } \mathrm{PMI}(w_u, w_v) = \log\left(\frac{p(w_u, w_v)}{p(w_u) \times p(w_v)}\right) < 0 \\ cooc(w_u, w_v), \text{ otherwise} \end{cases} \tag{2.15}$$

Additionally, they filter low and high degree nodes (low and high-frequency words) and keep only the k-core of the network[5]. All these steps were empirically determined to better detect communities with the `Label Propagation` algorithm before applying an *ad hoc* measure (Equation (2.16)) to weigh each component for each vector. Let e_u be the embedding vector representing node u, $e_u \in \mathbb{R}^{|C|}$ and e^{C_i} the value of the component of e_u corresponding to community C_i, $N(u)$ the set of neighbors of u (Definition 2). $\mu\left(\hat{e}_*^{C_i}\right)$ and $\sigma\left(\hat{e}_*^{C_i}\right)$ are the mean and standard deviation of values $\hat{e}_u^{C_i} \forall u \in V$. Using *Shifted*-PPMI[6] (SPPMI) alleviates the influence of node degree and the *z-score* the influence of community sizes, put together, the weight of each vector's component is:

$$\begin{aligned} e_u^{C_i} &= \frac{\hat{e}_u^{C_i} - \mu\left(\hat{e}_*^{C_i}\right)}{\sigma\left(\hat{e}_*^{C_i}\right)} \\ \hat{e}_u^{C_i} &= \frac{1}{|N(u) \cap C_i|} \sum_{v \in N(u) \cap C_i} \mathrm{SPPMI}(u, v) \end{aligned} \tag{2.16}$$

The first results with this model were promising, but the method was very *ad hoc*, both in terms of preprocessing and weighting scheme. This resulted in mixed results that were overall unstable (and hard to reproduce) on other datasets. However, this proof of concept motivated what

[5] recursively removing nodes with less than k neighbors in the network until all nodes have at least k neighbors

[6] $\mathrm{SPPMI}(u, v) = \max(\mathrm{PMI}(u, v) - \log(k), 0)$, $\mathrm{SPPMI}(u, v) \in [0, 1]$ with k usually set to 5 [LG14].

followed, namely rethinking the components of the pipeline and theorizing it into the `LDBGF` framework. In the following iterations of the method, we progressively reduced *ad hoc* components and gained stability, not only on co-occurrence networks, but also on other types of networks.

THE BIRTH OF `SINr`

`SINr` was born at the beginning of my thesis. Following the early development and advances made in Connes and Dugué [CD19], we wanted to theorize the idea of using an intermediary grouping of nodes as a way to derive a node representation. To that end, we introduced `LDBGF` (Section 2.2.2). In `LDBGF`, we rely on a bipartite representation where nodes partitioned into groups and nodes are connected to this structure. In this framework, we go back to square one regarding the *ad hoc* methods employed in the proof of concept by introducing a new implementation: *Sparse Interpretable Node representations* (`SINr`). In this first iteration of `SINr`, we extend the field of possibilities by not only considering word co-occurrence networks but also more traditional networks.

[CD19] Connes and Dugué "Apprentissage de plongements lexicaux par une approche réseaux complexes"

WORD CO-OCCURRENCE GRAPH PREPROCESSING. As text is not naturally represented as a graph, some specific preprocessing step needed to be applied to proceed with the `SINr` pipeline. We kept the PMI filtering of edges and introduced an edge reweighting scheme to dampen the influence of high-degree nodes, as they can be detrimental in community detection [Yan+18]. What does high-degree mean in the context of word co-occurrence networks? Since nodes represent words, a high degree node is a word that appears in a diverse variety of contexts and/or is among the most frequent terms in the lexicon. They may thus act as strong hubs between communities. Without reducing their influence, the application of `Label Propagation` would potentially lead to a degenerate partition with a single community. To prevent this outcome, the following reweighting scheme is applied:

Definition 18 (Iterative Pointwise Mutual Information (IPMI))**.** Let E_{ord} be the edges (u, v) of E ordered from the highest to the lowest sum of its weighted degrees $d_w(u) + d_w(v)$. For each edge (u, v) ordered as in E_{ord}, we iteratively update the weight of edge (u, v), $W_{u,v}$ with the `IPMI` value:

$$\text{IPMI(u, v)} = \frac{W_{u,v}}{d_w(u)d_w(v)} \tag{2.17}$$

We see Figure 2.8 the weighted degree of nodes before and after `IPMI`, the tendency towards an inverse function seems to show that it is effectively reducing the overall weight of high degree nodes. As stated previously, frequent words appear in a diverse set of contexts and can be polysemous. This polysemy and their weight in the graph topology may favor the formation of a community with semantically unrelated words based on their common neighbor. Subsequently, with the process described, we base the community detection on words that are potentially more specific and whose co-occurrence might be more relevant for community detection.

SINr PIPELINE. Following LDBGF principles, communities on graph and co-occurrence networks are respectively extracted with `Louvain` and `Label Propagation` since they provided the best performances with either data in comparison with other community detection algorithms [Pro+21]. An advantage of both algorithms against their contenders is their time complexity: Raghavan et al. [RAK07]'s `Label Propagation` algorithm has a near linear $\mathcal{O}(m)$ complexity with regard to $m = |E|$, the number of edges; Blondel et al. [Blo+08]'s `Louvain` has a complexity in the order of $\mathcal{O}(n \log(n))$.

[Pro+21] Prouteau et al., "SINr: Fast Computing of Sparse Interpretable Node Representations is not a Sin!"

[RAK07] Raghavan et al. "Near linear time algorithm to detect community structures in large-scale networks"

[Blo+08] Blondel et al. "Fast unfolding of communities in large networks"

Embedding vectors are weighted using Dugué et al. [DLP19]'s Node F-measure Framework. This framework, inspired by work on estimating the quality of clustering, can be used to quantify the interactions of a node within its community as well as with other communities. Most notably two measures are of particular interest to our framework: Node Predominance (NP, Definition 19) and Node Recall (NR, Definition 20). These two measures correlate with other centrality measures. NP is related to Lancichinetti et al. [Lan+10] embeddedness and correlates with PageRank betweenness. NR, on the other hand, correlates with the participation coefficient for well-defined community structures and provides a stable alternative in the case of less defined community structures [DLP19]. Furthermore, NP and NR rely directly on community structure and the weights of edges linking nodes inside and outside of communities. Thus, these measures are efficient to compute in comparison with measures involving z-scores like the intra-module degree of Guimera and Amaral [GA05] or shortest paths as required in Betweenness Centrality (Definition 35) [Ant71; Fre77] which are common in feature-selection approaches to node or edge embeddings[7] in machine learning.

[DLP19] Dugué et al. "Bringing a Feature Selection Metric from Machine Learning to Complex Networks"

[Lan+10] Lancichinetti et al. "Characterizing the Community Structure of Complex Networks"

[GA05] Guimera and Amaral "Cartography of complex networks"

[7] As for example the `Heuristics` [SCV19] method presented later Section 3.2.1.

Given a weighted graph $G = (V, E, \Omega)$, a node $u \in V$, a partition of the nodes $\mathcal{C} = \{C_0, \ldots, C_j\}$, and C_i the i_{th} community so that $1 \leq i \leq j, d_{C_i} = \sum_{u,v \in C_i} \Omega_{uv}$ the total weighted degree of C_i and, $d_{C_i}(u) = \sum_{v \in C_i} \Omega_{uv}$ is the weighted degree of u in community C_i, the node predominance and node recall of u considering the i^{th} community are :

Definition 19 (Node Predominance (NP))**.** Node Predominance measures the contribution of a node to the degree of a community. The higher the NP value, the more a node is connected to its com-

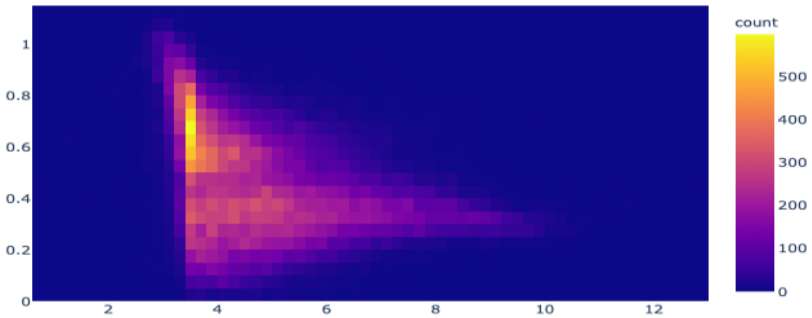


FIGURE 2.8: Heatmap of the weighted degrees of the graph extracted on `OANC` after IPMI according to the weighted degrees before applying IPMI, abscissa in logarithmic scale.

munity.

$$\mathrm{NP}_i(u) = \frac{d_{C_i}(u)}{d_{C_i}} \tag{2.18}$$

Definition 20 (Node Recall (NR))**.** Node Recall measures the contribution of each community to the degree of a node.

$$\mathrm{NR}_i(u) = \frac{d_{C_i}(u)}{d(u)} \tag{2.19}$$

If we recall `LDBGF`, the objective is to measure the relative connection of each node to each clique and provide a representation in lower dimension encompassing the clique membership. In the case of `SINr`, it is the same process that is performed with NP and NR, but with regard to communities. These two measures are computed for each node with regard to each community. The NP and NR values are concatenated in a single vector of dimension $2p$, $p = |\mathcal{C}|$. Vectors have many dimensions, but the model is sparse. Since not every node is connected to every community, only a subset of dimensions is activated for each node, thus putting the dimension of representations into perspective. We will study Chapter 5 the benefits of sparsity in embedding models.

Our *Sparse Interpretable Node representation* method is composed of two simple steps, detecting communities and weighing each node's vector with regard to the community structure. Following this new pipeline, we evaluated `SINr`'s first iteration on classical text and graph evaluation tasks such as *link prediction*, *word similarity*, *concept categorization*. `SINr` obtained encouraging results and an enthusiastic welcome from the community in Prouteau et al. [Pro+21] which motivated additional refinements of the method as well as a deeper dive into interpretability. We now present the refinements made to the model after the initial implementation and our first exploitation of the interpretability of models.

SINr for interpretability: refining the model

With our first interpretability study, we changed community detection algorithm as `Louvain`'s hierarchy allowed us to have a hierarchy of partitions from which we could choose, therefore having more control over the number of dimensions than with `Label Propagation`. This provided the added benefit of not requiring a reweighting of the edges, which removed an *ad hoc* component in the pipeline.

Our first dive into interpretability was in the form of a human evaluation. The human evaluation of interpretability that we call *word intrusion evaluation* is detailed more in depth Section 5.3.1. To put it simply, this task inspired by Chang et al. [Cha+09] aims at evaluating the coherence of vector spaces' components. That is, in our model, the coherence of the words with the strongest values for a component related to a community. When applied to word embeddings derived from a co-occurrence network, a well-formed model should exhibit related words when we select a component at random. We present Table 2.2 the words for which the value is the

[Cha+09] Chang et al. "Reading Tea Leaves"

highest among the strongest dimensions in the vectors of words "*insulin*" and "*mint*".

	SINr
insulin	hypertriglyceridaemia, mellitus, porcine aldosterone, aminotransferase, creatinine ulcerative, sulphasalazine, colitis
mint	tbsp, oregano, diced Gibson, gigged, charvel minted, minting, hoards

TABLE 2.2: Three (one per line) most activated dimensions for words "*insulin*" and "*mint*" and for each dimension, three words with the strongest values on each dimension.

We can see semantically related terms appearing on each component for the two words. More medicine-related terms for "*insulin*" and three dimensions that seem unrelated to one another for "*mint*" but in which words seem related. We have only sampled here the three strongest dimensions among the thousands available from the model. Because `SINr` considers the connection of a word to a community and not all words are connected to all communities, the embeddings are sparse. Therefore, each word is represented by a subset of dimensions that are hopefully interpretable like those Table 2.2. However, even a couple hundred dimensions activated per word is prohibitive in evaluating the interpretability of a word vector. That is why we later focus on sparsifying the vectors to a few activated dimensions per word to provide a subset of interpretable dimensions manageable for humans. The results of this experiment will be detailed Chapter 5.

In parallel with this interpretability challenge, we continue on improving upon the `SINr` pipeline, soon introducing an alternative matrix factorization-based model that completes our existing implementation. We also keep refining the original pipeline to gain efficiency and control over the representation.

2.3.3 *SINr-NR: community-distributed embeddings with heuristic*

All these iterations in the development of our community-based graph embedding helped us shape the method we have today. The first of our two implementations of `LDBGF` is `SINr-NR`: *Sparse Interpretable Node representations with Node Recall* presented in [PDG24].

[PDG24] Prouteau et al., "From Communities to Interpretable Network and Word Embedding: a Unified Approach"

With `Louvain`, we have found a way to control embedding dimension. However, relying on the `Louvain` hierarchy means that we are only able to control the size of embeddings by choosing a higher or lower-resolution partition in the dendrogram. Yet, the γ-resolution parameter is an alternative to using hierarchical levels of `Louvain`. The γ parameter acts directly on the modularity value of a partition (Equation (1.3)), subsequently, a high γ will prevent `Louvain` from gathering nodes in communities. Therefore, it is possible to influence the dimension of the output space by tweaking the γ value.

The method formerly known as `SINr` will from now on be known as `SINr-NR`, for *Node Recall* as we do not use *Node Predominance* anymore to

weigh vectors. Based on the partition in communities, `SINr-NR` weights the vector using the node recall (NR; Definition 20), the *Node Predominance* (NP; Definition 19) that was previously computed is not used in the representation as it increased vector dimensionality by a factor 2 while its contribution to the quality of representations was marginal and in some cases detrimental. For example, we proceed Section 3.3.2 to evaluate node embeddings via a *link prediction* task. The goal is to predict in a supervised setting whether an edge links two nodes. In Table 2.3, we can see that dropping NP has not only allowed to reduce embedding dimension but to also maintain *link prediction* results. In parallel to our work on `SINr`, Lutz [Lut22] developed a similar pipeline using the membership matrix obtained with `Louvain` and weighing each vector with NR, reaffirming our choice to only keep NR. In topic modeling, a task closely related to word embedding, Austin et al. [AZL22]'s `Community Topic` leverages community detection to provide a hierarchy of topics that is interpretable as it relies on communities and is close in philosophy to `SINr-NR`.

[Lut22] Lutz "Graph-based contributions to machine-learning"

[AZL22] Austin et al. "Community Topic: Topic Model Inference by Consecutive Word Community Discovery"

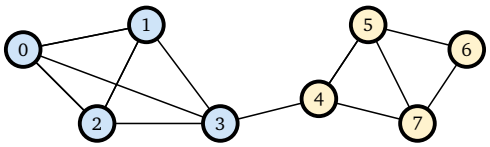
	`SINr` [Pro+21]	`SINr-NR` [PDG24]
`Cora`	0.83	0.85
`Eu`	0.88	0.86
`Cts`	0.88	0.88
`arXiv`	0.92	0.93
`Fb`	0.91	0.92

TABLE 2.3: Comparison of *link prediction* results (accuracy) using `Label Propagation` and NP+NR (`SINr`) using `Louvain` with NR (`SINr-NR`).

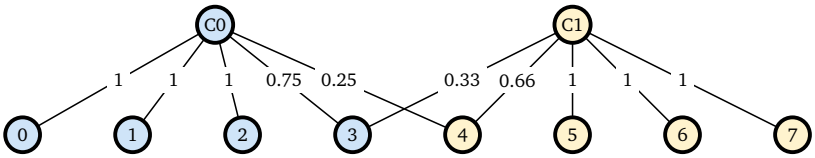
We now only use NR to weight the vectors. However, NP can still be used to quantify the importance of a node within the community. With NR, we are essentially quantifying the distribution of weighted degree of a node u towards each community C_i (2.9(b)). Upon computing *NR* for one node over all the communities, one obtains a vector representing the node considered. When computed for all vertices in the graph, we obtain the embedding model. The embedding vectors are thus still sparse (2.9(c)), but their dimension is linear with the number of communities. The `SINr-NR` pipeline is summarized Figure 2.9: from a weighted graph (Figure 2.9(a)), detect the communities, weight edges in a bipartite network where edges connect nodes to the communities they have neighbors in with NR (Figure 2.9(b)), and finally obtain a vector representing each node (Figure 2.9(c)).

We also introduce an alternative approach to extract community-based node and word embeddings: `SINr-MF`, for *Matrix Factorization* (Section 2.3.4). Subsequently, `SINr` is a family of methods basing node representation on communities detected in networks.

`SINr-MF` avoids *ad hoc* weighing schemes like NR. Instead, `SINr-MF` has the objective of factorizing the adjacency matrix of the graph into a product of the community membership matrix and a latent space represented by a matrix U that is uncovered by gradient descent.



(a) A graph $G = (V, E)$ partitioned in two communities.



(b) Bipartite projection of G into graph $G' = (\top, \bot, E', \Omega')$ along the communities. Weight on the edges is the *NR* value regarding the community (the proportion of neighbors in that community).

	C0	*C1*
0	1	0
1	1	0
2	1	0
3	0.66	0.33
4	0.33	0.66
5	0	1
6	0	1
7	0	1

(c) Adjacency matrix of G', each row is a `SINr-NR` embedding.

FIGURE 2.9: Illustration of `SINr-NR`, nodes are represented based on the communities they are linked to.

2.3.4 *SINrMF: finding the transition from network to communities*

Many embedding methods, both for word and graph embeddings, rely on some sort of matrix factorization. More specifically, as we have seen in Section 2.1.3, factorizing the adjacency matrix, the graph Laplacian or a matrix based on a similarity measure are popular methods.

Following this idea, the second implementation of `SINr`, does not rely on an *ad hoc* measure of strength of connection such as *NR*. Instead, it is inspired by embedding methods based on `SVD` and `NMF`. `SINr-MF` has a factorization objective, in line with `LDBGF` and the idea that community structure can be an informative substructure to represent nodes. We named it `SINr-MF` for Matrix Factorization. It attempts to factorize the adjacency matrix $\mathcal{A}$ of a graph G into the product of two matrices, U which is initialized at 1 and C–the community membership matrix extracted using `Louvain`. We believe that using C in the optimization of U is useful as the community membership matrix provides strong topological information with regard to nodes, and it allows interpretability. In our case, matrices $\mathcal{A}$ and C are known. By letting gradient descent handle the optimization of U we hope to avoid the *ad hoc NR* measure of `SINr-NR` and uncover the latent space between $\mathcal{A}$ and C. The goal of `SINr-MF` is thus to minimize the difference

between $\mathcal{A}$ and the product of U and C in the following model:

$$\text{SINr-MF}(G) = \underset{U}{\text{argmin}}(\text{MSE}(\mathcal{A}, UC^T)) \tag{2.20}$$

This optimization is performed by gradient descent using a *Mean Squared Error* (MSE) loss. Error is back propagated to U to iteratively adjust vectors' weights. `SINr-MF` comes with a higher time complexity than `SINr-NR` and also a large space complexity since $\mathcal{A}$, U and C^T need to be stored in memory at training time.

- 2018 – Genesis of `SINr` @ TALN [CD19]; Label Propagation & z-score embeddings; Proof of concept
- 10/2020 – Start of thesis
- 04/2021 – `SINr` v1 @ IDA [Pro+21]; Setting & Contributions: introduction of `LDBGF` & `SINr` w/ `Label Propagation`, reweighting of graph & NP + NR, quantitative evaluation of the model on graph and word embedding tasks
- 07/2022 – `SINr` v2 @ LREC [Pro+22]; Setting: `SINr` w/ Louvain Hierarchy & NP + NR; Contributions: human evaluation of interpretability
- 06/2023 – `SINr` v3 @ FRCCS [Pro+23]; Setting: `SINr` w/ Louvain + γ-resolution & NR; Contributions: release of `SINr` library on PyPi for the public (`pip install sinr`)
- 07/2023 – `SINr` v3 w/ Sparsification @ IWCS & TALN [GPD23b; GPD23a]; Contributions: extend interpretability to the vector-level (previously interpretability of dimensions), investigation of the contribution of sparsity to interpretability
- 04/2024 – `SINr` becomes a family of methods [PDG24] (Major revision); Setting: `SINr` becomes `SINr-NR` (*Node Recall*) and introduce `SINr-MF` (*Matrix Factorization*); Contributions: thorough evaluation on graph embedding and word embedding

FIGURE 2.10: Timeline of `SINr`'s evolution.

2.3.5 *SINr library*

Since the start of the development of `SINr`, we kept refining each component of the pipeline to meet our goals of frugality and interpretability. After three iterations of our model, the pipeline was stable enough for a library

to be released for use by the public. This stable pipeline coincided with the structuration of a project team around `SINr` thanks to the DIGING project funded by the ANR. We thus made `SINr` available on PyPI so that it can be installed with `pip`[8]. The source code for our implementation of `SINr` is available on GitHub, with documentation, and automated tests. These components are meant to structure and streamline the development process of `SINr` using a continuous integration, continuous deployment approach. This structuration goes hand in hand with the growing size of contributors to the project, from two at its beginnings, to now four regular contributors. `SINr` has also been integrated in the *KarateClub* library [RKS20]. This library provides numerous implementations of graph embedding techniques with a unique API.

[8] `pip install sinr`

[RKS20] Rozemberczki et al., "Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs"

Additionally to the implementation, we provide introductory material to learn and visualize `SINr` embeddings. This material has been used during a network theory class where third-year bachelor students implemented `SINr`. We provide Appendix B a demonstration of `SINr`'s python library and visualizations of word embeddings.

2.4 SUMMARY

This chapter was dedicated to representation learning and how vectors encompassing the topology of networks can be extracted. The key points we presented in this chapter are the following:

(i) Embeddings provide a vector representation of data in a latent space that is supposed to encapsulate the topology (word meaning, node position) in the system it models. In NLP, word embeddings are essential to most machine learning pipelines. The same goes for graph embedding, which has the objective of encapsulating the intricacies of network structure, for example, in node or edge embeddings.

(ii) Embedding methods share a kinship. They rely on the distributional hypothesis that states that items can be represented using surrounding elements (context in sentences, neighbors, and structure of the network). Methods in graph embedding and word embedding share similarities. We distinguished broad families of methods: eigendecomposition, factorization with similar objectives (`Word2vec` and `GloVe`), and deep neural approaches that make use of attention mechanisms.

(iii) Methods have limits. Long runtimes and huge amounts of data needed by some methods are concerning regarding sustainability objectives. The lack of interpretability and transparency of methods and model produced can be problematic in high-stake applications of representation learning (NLP for medicine, justice, *etc.*).

(iv) To alleviate the limitations of embedding models, we introduce *Lower Dimension Bipartite Graph Framework* (`LDBGF`) which compresses graph information using a bipartite projection according to cliques in the graph. Thus, representations are interpretable since they stem from the cliques. However, this framework remains theoretical, as the problem of finding cliques to cover all graph edges is NP-Hard.

(v) Communities are an alternative to cliques that can be detected efficiently. We introduce two implementations inspired from `LDBGF`, but with communities. The principle is the same, compressing graph information according to communities instead of cliques. Nodes are represented according to their connections to nodes in communities. `SINr-NR` is the first implementation and uses an *ad hoc* measure of connectivity between nodes and communities. `SINr-MF` is based on matrix factorization, trying to find the transition matrix from graph adjacency to community membership.

(vi) The implementation of our method is open source and available to use by the public on consumer hardware.

The next three chapters are dive into the specifics of graph and word embedding, demonstrating how `SINr-NR` and `SINr-MF`, at their level, contribute to breaking through the scientific bottlenecks that come with embedding methods that we have identified: computation cost and interpretability.

In Chapter 3, we focus on network embeddings, introducing additional methods to the ones presented Section 2.1.3. We then introduce evaluation methods to assess model performance on downstream tasks such as *link prediction* or *topological feature prediction*. Finally, we discuss `SINr-NR` and `SINr-MF`'s performances against popular graph embedding architectures on various network genres and sizes, showcasing the benefits of `SINr-NR` and `SINr-MF`.

Part II

BAPTISM OF FIRE FOR A UNIFIED MODEL

3

Modeling networks: network embedding

Chapter contents

Synopsis

Graph embeddings are supposed to capture graph topology and summarize it in low-dimension vector representations. Methods rely on different techniques to capture as much topological information as possible. However, they sometimes suffer from long runtimes that is detrimental to their use. Furthermore, not all methods embed topology equally and some models are more inclined to perform well on specific tasks. In this chapter, we investigate the abilities of `SINr-NR` and `SINr-MF` on a variety of tasks against competing graph embedding approaches. Our experiments will focus on several tasks assessing the information content of representations: *link prediction*, *graph-feature prediction*, and *community clustering* from their representation. We show that `SINr-NR` and `SINr-MF` are versatile models worth considering when learning graph representations.

Résumé

Les plongements de graphes ont pour objectif de capturer la topologie d'un réseau et de la résumer dans une représentation vectorielle compacte. Les méthodes permettant d'obtenir des plongements de graphes s'appuient sur différentes techniques, pour capturer, aussi précisément que possible cette topologie. Cependant, ces méthodes ont parfois pour inconvénient de nécessiter de longs temps d'apprentissage lorsqu'elles sont confrontées à de grands réseaux. De plus, toutes les méthodes ne capturent pas l'architecture des réseaux avec le même succès, certains ont de bonnes performances sur des tâches bien spécifiques. Ce chapitre à pour but d'évaluer les capacités de `SINr-NR` et `SINr-MF` à produire des représentations pertinentes. Leurs performances sont comparées à celles d'autres algorithmes permettant d'obtenir des plongements de graphes, et ce sur des tâches variées. Nos expériences attraient à évaluer la qualité des plongements de graphes produits au travers de: la *prédiction de liens*, la prédiction de caractéristiques des noeuds et du graphe (degré, coefficient de clustering, PageRank) et a former des groupes de noeuds à partir de leurs représentations. Nous montrons que `SINr-NR` et `SINr-MF` sont polyvalents et qu'ils peuvent fournir des plongements de graphes de qualité. `SINr-NR` a l'avantage de fournir ces représentations avec un temps de calcul réduit.

3.1 NETWORK EMBEDDING METHODS

3.1.1 *Methods*

We have observed in Section 2.1.3 that network embeddings and word embeddings share a kinship related to the distributional hypothesis (Section 2.1.1) and also in their implementation. We presented methods relying on eigendecomposition to factorize matrices, methods relying on optimization of objective function like `SGNS` or `GloVe` and neural methods that leverage attention mechanisms along with deep neural architectures. Node embedding methods are not restricted to those presented in Section 2.1.3, there exists other factorization methods such as `LINE` [Tan+15] which optimizes for first and second-order proximities between nodes. `GraRep` [CLX15] factorizes a k-step transition matrix between nodes. Among other popular neural node embedding methods, we can cite `SDNE` [WCZ16] and `DNGR` [CLX16] that make use of deep autoencoders to preserve graph proximities and model the PPMI (Definition 17). Some algorithms focus on preserving specific substructures such as communities (second or third-order): `GEMSEC` [Roz+19] combines the objective of `Skip-Gram` with vertex clustering to preserve community information, `M-NMF` [Wan+17] combines the modularity associated with community detection and `NMF`. Also using communities, `LouvainNE` [Bho+20] leverages the hierarchical structure provided by community detection to extract node representations. There are many more methods that aim at providing useful node representations. However, this chapter will not suffice to survey all of them.

[Bho+20] Bhowmick et al., "LouvainNE: Hierarchical Louvain Method for High Quality and Scalable Network Embedding"

The goal here is to present in detail the network embedding algorithms we have chosen to benchmark, along with `SINr-NR` and `SINr-MF`. The choice of network algorithms was made by considering multiple requirements. First, we focused on node embedding methods that provide a representation for nodes in a network. Edge embeddings exist [SCV19; Wan+20] but are largely underrepresented in comparison to node embedding methods, whose embeddings can be combined to form an edge embedding. Second, we discarded `GNN` methods as they do not comply with the goal of low-compute time and resources. Third, we selected methods for which implementations were provided by the authors or available in complex networks manipulation libraries such as *KarateClub*. Based on these requirements, we selected five methods that are evaluated alongside `SINr-NR` and `SINr-MF`: `Deepwalk`, `Walklets`, `HOPE`, `LouvainNE` and `VERSE`. We detail hereafter each method we have chosen to evaluate.

[SCV19] Sinha et al., "Systematic Biases in Link Prediction"

[Wan+20] Wang et al., "Edge2vec: Edge-based Social Network Embedding"

▸ DEEPWALK & WALKLETS

MODEL DESCRIPTION. We previously (Section 2.1.3) introduced `Deepwalk` [PAS14] and `Walklets` [Per+17] as methods relying on random walks (`RW`) (Definition 15) to compute node representations. They both rely on variation of `RW` to generate sequences of nodes that are in turn used in place of sentences in a `Word2vec` fashion with `Skip-Gram`. `Deepwalk` and `Walklets` handle `RW` differently. Let $G = (V, E)$ be an undirected graph. `Deepwalk` proceeds to sample γ RW of length l per node[1]. `Walklets` proceeds to sample

[1] Random walks can have restarts but it did not prove, at the time, to be beneficial.

γ RW per node too, but with the added objective of sampling RW to capture multiple scales of relationship. To that end, Walklets samples its sequences with a skipping mechanism (skip of s in the RW), such that some nodes are skipped in the RW and distant nodes that would not co-occur within the same co-occurrence window of the RW in Deepwalk co-occur in Walklets. By sampling RW with varying skip length, Walklets is supposed to capture more distant relations than RW rooted at a node u and of length t.

Based on these sequences, the Skip-Gram algorithm can be applied with the objective of maximizing the likelihood of predicting co-occurring nodes in the RW based on a target node. Mathematically, this results in the following objective, given $N = \gamma \times |V|$ the number of RW, l the length of each RW, ℓ the window size indicating the number of nodes to consider for each target node within a RW, $\mathcal{W}_i^t$ the t-th node in the i-th RW, $\mathcal{W}_i^{t+k}$ the context node k steps away from $\mathcal{W}_i^t$ within the same RW, and $e_{\mathcal{W}_i^j}$ the embedding of node i at step j the objective function is:

$$\mathcal{L} = \frac{-1}{N} \sum_{i=1}^{N} \sum_{t=1}^{l} \log\left(p(\mathcal{W}_i^{t+k} | \mathcal{W}_i^t)\right) \tag{3.1}$$

$$p(\mathcal{W}_i^{t+k} | \mathcal{W}_i^t) = \frac{\exp\left(e_{\mathcal{W}_i^t} \cdot e_{\mathcal{W}_i^{t+k}}\right)}{\sum_{j=1}^{|V|} \exp\left(e_{\mathcal{W}_i^t} \cdot e_{\mathcal{W}_i^j}\right)} \tag{3.2}$$

This is the Skip-Gram objective of Word2vec, but applied to RW sequences. In the case of Walklets, multiple models can be extracted by varying the skipping step s before concatenating them and reducing their dimension with a PCA. In the implementation used, γ RW are sampled for values of s varying between 1 and the maximum skip size set. After each sampling step, Skip-Gram is applied to the corpus extracted.

EXPERIMENTAL SETUP. In our experiments, we use the implementations of Deepwalk and Walklets provided in *KarateClub* [RKS20]. The parameters chosen are the following: 10 RW are sampled per node with a length l of 80. The window for co-occurrence ℓ is set to 5 for Deepwalk and the maximum skip step is set to 4 for Walklets. All models extracted from networks have 128 dimensions.

▸ HOPE

MODEL DESCRIPTION. *High Order Proximity Embedding* (HOPE) has the goal of factorizing a node similarity matrix into the product of two matrices U_s and U_t and preserve high-order proximities. To that end, multiple similarity metrics can be used: Katz Index Definition 26, Rooted PageRank (Definition 32), Common Neighbors (Definition 21), and Adamic Adar (Definition 22). Based on the pairwise similarity matrix S, the goal is to minimize the L2-norm between S and the product of U_s and U_t:

$$min||S - U_s \cdot U_t^\intercal||^2 \tag{3.3}$$

Matrices analogous to U_s and U_t can be obtained by applying SVD to decompose S into the product:

$$SVD(S) = X_s \Sigma X_t^\intercal \tag{3.4}$$

Matrices $U_s = X_s \cdot \Sigma$ and $U_t = X_t^\intercal \cdot \Sigma$ can then be used as representations by concatenating their vectors in turn resulting in an embedding model U.

EXPERIMENTAL SETUP. We use the implementation of `HOPE` provided in *Karate-Club*. The sparse `SVD` is applied to the neighborhood overlap matrix $(\mathcal{A} \cdot \mathcal{A})$ and the embedding model is the concatenation of vectors of $U_s = X_s \cdot \sqrt{\Sigma}$ and $U_t = X_t^\intercal \cdot \sqrt{\Sigma}$ in model U. The dimension of `HOPE` embeddings is set to 128 for all experiments related to node embeddings.

▸ LOUVAINNE

[Bho+20] Bhowmick et al. "LouvainNE: Hierarchical Louvain Method for High Quality and Scalable Network Embedding"

MODEL DESCRIPTION. Bhowmick et al. [Bho+20] introduced `LouvainNE` that is able to derive node embeddings from a hierarchy of communities. Behind `LouvainNE` is the same idea that brought `SINr` to life. The idea that communities can inform representation since they are groups of densely connected nodes. Thus, they likely share properties or play similar roles in the topology, and it seems logic to place them close in the embedding space. Furthermore, leveraging multiple granularities of communities as `LouvainNE` does may help preserve high-order as well as local proximities between nodes. To that end, `LouvainNE` has three steps:

(i) First, construct a hierarchical tree with a partitioning algorithm such as `Louvain` or `Label Propagation`. The root of the tree contains all graph nodes, each other level of the tree is a partition of the nodes of the previous level. This in turn returns a tree in which each child is either a community or a leaf, in the case where a community is a singleton. This hierarchy is extracted recursively.

(ii) Based on the hierarchy, two solutions are proposed to compute embeddings at each level h of the hierarchy. The first is to learn an embedding with a node embedding method (*e.g.*, `Deepwalk`, `node2vec`, `Walklets`) on a weighted meta-graph with children node from a same parent. The edge between two children is based on the number of shared edges between nodes in both communities in the original network[2]. The second solution proposed is to sample a random vector of d dimensions from the standard normal distribution $\mathcal{N}(0,1)$. The random nature of the sampled vectors will maintain a good separation between nodes that do not share the same communities after step (iii).

[2] Let S_1 and S_2 be two children such that given $G = (V, E, \Omega), S_1, S_2 \in E$, the weight of the edge $\Omega_{S_1,S_2} = \frac{|\{(u,v) \in E | u \in S_1, v \in S_2\}|}{|S_1| \cdot |S_2|}$, normalized to account for large communities.

(iii) Represent each node by the sum of the vectors from the top level $t = 1$ to the bottom level $t = h$. At the same time, it dampens the contribution of each representation with a parameter $\alpha \in [0,1]$ the further down the hierarchy it travels. Thus, the output embedding e_u for a node u is :

$$e_u = \sum_{t=1}^{h} \alpha^{t-1} e_u^t \tag{3.5}$$

This allows organizing the embedding space so that the further down nodes are next to each other in the hierarchy, the closer their representation.

EXPERIMENTAL SETUP. We use the C implementation of `LouvainNE` provided by Bhowmick et al. [Bho+20]. We use the default parameters in our experiments on graphs with $\alpha = 0.01$, the number of dimensions is set to 128 and the partitioning algorithm is `Louvain`.

VERSE

MODEL DESCRIPTION. *Versatile Graph Embeddings from Similarity Measures* (`VERSE`) [Tsi+18] leverages node similarity to extract word embeddings. `VERSE` starts by measuring pairwise similarities between nodes. Similarity metrics used are supposed to approximate high-order proximity (more than just local neighborhood) and include: *Personalized* PageRank (variation of Definition 32), *Adjacency similarity*, and *SimRank* (Definition 27) [JW02]. It optimizes the embedding space by trying to minimize the difference between the similarity S_E of the embedding of two nodes in the embedding space and their similarity S_G in the graph. Schematically, `VERSE` has the following objective:

[Tsi+18] Tsitsulin et al., "VERSE: Versatile Graph Embeddings from Similarity Measures"

$$\underset{X}{\operatorname{argmin}} \left(\sum_{u,v \in E} (S_G(u,v) - S_E(X_u, X_v)) \right) \tag{3.6}$$

Detailing the approach, we can see that the loss $\mathcal{L}$ is a cross-entropy between the similarity measured on the graph and the representation:

$$\mathcal{L} = -\sum_{u \in V} S_G(u,\cdot) \log(S_E(u,\cdot)) \tag{3.7}$$

$$S_E(u,v) = \frac{\exp(X_u \cdot X_v^\intercal)}{\sum_{w \in E} \exp(X_u \cdot X_w)} \tag{3.8}$$

given a graph $G = (V, E)$, $u, v \in V$ two nodes and X_u, X_v their respective embeddings in X. `VERSE` minimizes the cross-entropy loss Equation (3.7). However, training also includes negative examples in a `Word2vec`'s `SGNS` fashion. *Noise Contrastive Estimation* (NCE) is used to make the model converge as in `SGNS`, the goal is to train a binary classifier to distinguish between samples corresponding to S_G ($D = 1$) and samples from a noise distribution $\mathcal{Q}$ ($D = 0$).

$$\begin{aligned} \mathcal{L}_{NCE} = \sum_{\substack{u \sim \mathcal{P} \\ v \sim \text{sim}_G(u,\cdot)}} & \left[\log\left(p\left(D = 1 \mid S_E(u,v)\right)\right) + \right. \\ & \left. s\mathbb{E}_{\widetilde{v} \sim Q(u)} \log\left(p\left(D = 0 \mid S_E(u,\widetilde{v})\right)\right) \right] \end{aligned} \tag{3.9}$$

where p is computed from the X as the sigmoid σ of $X_u \cdot X_v^\intercal$. In this case, S_E is not normalized as Equation (3.8) but just the dot product of the embeddings. s samples are drawn from $\mathcal{Q}$ for each node u, thus, s times more negative samples are used in training.

`VERSE` objective and especially NCE's use of negative samples presents clear similarities with `Word2vec` that is also employed in `Deepwalk` and `Walklets`, although a similarity measure is used in place of random walks.

EXPERIMENTAL SETUP. We employ VERSE's implementation provided by the authors and that is implemented in C++. We keep the default parameters, that is personalized PageRank for S_G (with 0.85 alpha), 128 dimensions, and $s = 3$ negative samples.

Method	Language / Origin	Optimization	Dimension	Misc. Parameters
Deepwalk	Python/ KarateClub[3]	Word2vec (Skip-Gram) w/ RW	128	RW/node: 10 RW length: 80 window size (context): 5
Walklets				RW/node: 10 RW length: 80 maximum skip-step: 4
HOPE		SVD w/ Pairwise similarity		Similarity: neighborhood overlap matrix
VERSE	C++ / Authors[4]	SGNS w/ Pairwise similarity		Similarity: Personalized PageRank Negative samples/node: 3
LouvainNE	C / Authors[5]	Community Detection w/ hierarchy & aggregation		Community detection: Louvain Dampening coeff. α: 0.01
SINrNR	Python / Authors[6]	Community detection + *ad hoc* ponderation	Number of Communities	Community detection: Louvain
SINrMF		Community detection + Matrix Factorisation w/ SGD		

TABLE 3.1: Summary of graph embedding methods with their hyperparameters.

[3] https://github.com/benedekrozemberczki/karateclub/tree/master

[4] https://github.com/xgfs/verse

[5] https://github.com/maxdan94/LouvainNE

[6] https://github.com/SINr-Embeddings/sinr

3.1.2 *Limits of methods*

As stated Section 2.1.4, the drawbacks we have identified with embedding methods are related to their time complexity that we aim to minimize to scale even to large graphs. Indeed, training times and resources are by no means comparable with what large neural architectures require. Yet, scalability to large networks is an important feature. Hyperparameters are also worth considering in the choice of a representation method. Their multiplicity might mean training many models before obtaining satisfactory results. Furthermore, interpretability can also be a desirable feature that is not available with many node embedding methods. On attributed networks, it can help draw interpretations about the information modeled, like in our airport network example (Section 2.2.1).

3.2 EVALUATING GRAPH EMBEDDING QUANTITATIVELY

Graph embedding models have a real use in practical applications, they can help predict friendships on social media, identify potential points of failure in physical infrastructures or even uncover interactions in living organisms. These applications of graph embeddings are commonly called downstream tasks, as they intervene once the model is trained. Downstream evaluations are useful to assess the quality of a representation. Not all models will perform equally on all tasks, and they can help weigh the pros and cons of each method depending on the application.

The graph embedding literature provides evaluation settings to assess the performance of an embedding model. But what is a good graph embedding model? Our objective with SINr is to provide a hybrid model for word and graph embeddings with three objectives in mind:

(i) Efficient computation of the representation

(ii) Interpretability of the representation

(iii) Does not sacrifice quality for the first two objectives

The first objective is rather easy to evaluate, we can measure compute time against state-of-the-art methods and compare their respective runtime. We dedicate Chapter 5 to interpretability and demonstrate how `SINr-NR` embeddings can be used to probe word sense in large text corpora. Subsequently, we first focus on evaluating embedding quality. Namely, we want to investigate what topological information and to which extent it is captured by embedding methods. As embeddings are a representation of the graph at hand, ideally, they should capture similarities and dissimilarities between nodes. For example, if two nodes are connected in the network, this connection should be encoded in the representation, so is the fact that they may belong to the same community.

We scoured the literature, and the first downstream task that appeared as a *de facto* evaluation of embedding quality is *link prediction*. *Link prediction* as its name hints, aims at predicting missing links in a network. It can be performed in multiple ways, on dynamic social networks to predict future friendships or collaborations [LK07], on static networks to predict missing data [BG07]. When performed with embeddings, *link prediction* can be a proxy to quantify whether similarities between nodes are captured and can help predict an edge from the representation of the nodes at its extremities.

[LK07] Liben-Nowell and Kleinberg, "The link-prediction problem for social networks"

[BG07] Bhattacharya and Getoor, "Collective Entity Resolution in Relational Data"

When *link prediction* can be used to evaluate node-to-node connectivity, we might wish to look further than just a very local similarity. Nodes are part of a broader structure, the network. Among nodes characteristics, many topological features convey information about the node and its position in the graph. The combination of these topological features may act as an *ID-card* specific to the node considered. Bonner et al. [Bon+17] investigate the extent to which topological features such as *Degree* or *Clustering Coefficient* are present in representations. Salehi Rizi and Granitzer [SG17] also probe the presence of signal regarding topological features by predicting their value. If a model provides enough information to extrapolate the degree of a node or its clustering coefficient, then, we can be confident that the node embedding method captures this information. Furthermore, not all topological features lie at the same level of organization in the network. We can thus quantitatively evaluate whether a model is more inclined to embed local, or microscopic-level information–local interactions. Another aspect of network organization that we could wish for graph embedding methods to capture is the organization of nodes into communities. Communities are dense groups of nodes that are more connected to each other than with the rest of the network. Thus, nodes within a community should be more similar than with those lying outside their community. Some networks have known community structures that can be used to probe their encapsulation in representations. Tandon et al. [Tan+21], Perozzi et al. [PAS14], Bonner et al. [Bon+17], and Bhowmick et al. [Bho+20] try to predict the membership of nodes to communities and evaluate whether information

[Bon+17] Bonner et al. "Evaluating the quality of graph embeddings via topological feature reconstruction"

[SG17] Salehi Rizi and Granitzer "Properties of Vector Embeddings in Social Networks"

[Tan+21] Tandon et al. "Community detection in networks using graph embeddings"

[PAS14] Perozzi et al. "DeepWalk: Online Learning of Social Representations"

[Bho+20] Bhowmick et al. "LouvainNE: Hierarchical Louvain Method for High Quality and Scalable Network Embedding"

about communities is present in representations. Communities belong to the mesoscopic-level of information organization. The highest level of information organization we describe is macroscopic. It relates to phenomenons that happen at the scale of the whole graph.

We extensively introduce each evaluation task and the networks used for evaluation Section 3.2.1. These downstream tasks allow us to have an overview of `SINr`'s capacities in comparison with state-of-the-art methods. We then thoroughly and quantitatively evaluate these aspects of node representations Section 3.3.

3.2.1 *Tasks*

Link Prediction

[LK03] Liben-Nowell and Kleinberg "The Link Prediction Problem for Social Networks"

Our effort to accurately model complex systems represented by networks led us to the *link prediction* task. Liben-Nowell and Kleinberg [LK03] describe the problem of *link prediction* in social networks. Given an evolving social network at a time step t, is it possible to predict the connection of two vertices at a subsequent step t'? The subsequent question to such a problem on all sorts of networks is: how can information intrinsic to the network allow predicting a future interaction between vertices? Figure 3.1 is a schematic presentation of *link prediction* to predict missing links.

[Bil92] Bilal *Froid équateur*

Exogenous factors to the network may also factor in future interactions between two vertices [LK03]. For example, the invention by French comic book writer Enki Bilal [Bil92] of chess boxing, the combination of blitz chess and boxing rounds is an exogenous factor to a boxer and a chess payer being connected in a graph modeling athletes and their opponent. As a consequence, *link prediction* is complex as it relies solely on what is present within the network and endogenous to the system modeled. A subsidiary question about the relevance of *link prediction* ensues from the complexity and predictive aspect of the task: why would one want to predict future and potentially spurious missing links in networks? The *link prediction* task can be seen as a *de facto* evaluation to assess the capacity of a model to convey similarity between vertices in a network. Even though *link prediction* has become a benchmark for performance, its applications are nonetheless concrete.

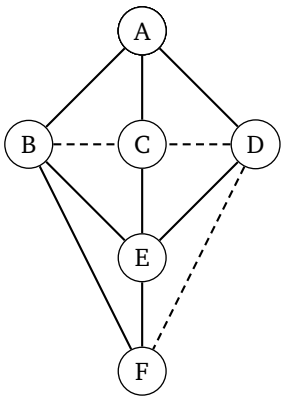


FIGURE 3.1: The *link prediction* predicts missing links (*i.e.*, (B, C), (C, D) and (D, F)).

One of the applications of *link prediction* is regarding protein-protein interaction networks (*PPI*). In a PPI network, proteins are represented as nodes, and the interactions between them are represented as edges connecting nodes. Understanding such interactions may help to understand cellular processes, disease mechanisms and help target medicine to certain interactions. *Link prediction* might help to focus attention on more probable interactions. Indeed, *in vitro* experiments to characterize which proteins interact are costly [SUF00; GR03; QBK06]. Recommender systems in e-commerce may use *link prediction* to recommend items of interest [HLC05; LZ11]. In the context of entity resolution, finding duplicate references or records in a dataset, *link prediction* can help to leverage context information (position and references to a record in structured data) instead of solely attributes. By doing so, Bhattacharya and Getoor [BG07] improve entity resolution. *Link prediction* also finds applications on social networks, like sug-

[BG07] Bhattacharya and Getoor "Collective Entity Resolution in Relational Data"

gesting acquaintances to users on social networks [LK07]. When it comes to security, *link prediction* may be used to uncover to some extent relations between individuals in criminal and terrorist organizations [Kre02; XC08; Su+19; ABG21]. Analyzing scientific collaborations through the lens of *link prediction* can help to uncover potential co-authors or collaborators within a company. In a static setting, *link prediction* can be used to infer missing links from networks constructed from citations [Ale03] or web pages and social networks [Tas+03].

[LK07] Liben-Nowell and Kleinberg, "The link-prediction problem for social networks"

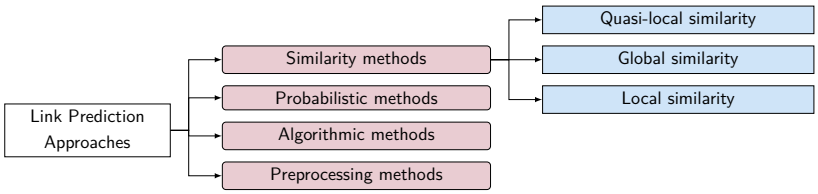


FIGURE 3.2: Taxonomy of link prediction methods.

Throughout the years, surveys on *link prediction* [HZ11; LZ11; MBC17; Kum+20] have shown the evolution of methods to better predict links. Figure 3.2 propose a taxonomy of *link prediction* methods based on Liben-Nowell and Kleinberg [LK03][7]. Four broad categories of methods are described depending on the underlying methods: similarity methods, probabilistic and maximum likelihood methods, dimensionality reduction methods and preprocessing methods. We detail the broad characteristics of each category and methods employed in our work in the following paragraphs.

[7] [LK03]'s taxonomy is further refined by [Kum+20]. Although their taxonomy is more detailed, some methods for *link prediction* fit under multiple categories. For example, our approach to *link prediction* fits both in the network embedding category and algorithmic methods with classification.

SIMILARITY-BASED APPROACHES

Similarity-based methods are the simplest among link prediction methods. They rely on a similarity metric s computed for a pair of vertices $u, v \in V$. We distinguish different types of metrics: local metrics are mostly related to a vertex and its neighbors, at the microscopic-level, global similarity that rely on the whole graph at the macroscale, and quasi-local similarity which is an intermediary between the two previous similarities. We detail hereafter some of the similarity measures that used in combination can help predict links with outstanding accuracy[8].

[8] see HTS results in Table 3.3.

LOCAL SIMILARITY HEURISTICS. Local similarity between nodes can help shape the decision whether two nodes are susceptible to be connected in the future. At this microscale, if two nodes are similar, chances are that they are connected. We detail the main similarity indices we will employ in our *link prediction* experiments with the following definitions.

Definition 21 (Common Neighbors). Let s_{uv}^{CN} be the overlap between neighborhoods $N(u)$ of u and $N(v)$ of v such that:

$$s_{uv}^{CN} = |N(u) \cap N(v)| \tag{3.10}$$

The more neighbors u and v share, the more likely they are to be connected [LK07]. For example, the more friends you have in common with someone, the more probable you are also friends with

that person, or at least you evolve in the same circles.

Definition 22 (Adamic Adar Index)**.** Let s_{uv}^{AA} be an adaptation of common neighbors that takes into account the degree of neighbors of vertices u and v and $d(x)$ be the degree of a vertex [AA03]. Adamic Adar Index gives more importance to lower degree, less connected vertices.

$$s_{uv}^{AA} = \sum_{w \in N(u) \cap N(v)} \frac{1}{\log d(w)} \tag{3.11}$$

Definition 23 (Resource Allocation Index)**.** Let s_{uv}^{RA} be the resource allocation index [ZLZ09] of vertices u and v. Let u and v not be directly connected, the resource allocation process considers how many units of resource u may be sent to v by considering the degree of their common neighbors and vice versa from v to u. Like Adamic Adar, Resource Allocation penalizes high degree vertices over low degree ones.

$$s_{uv}^{RA} = \sum_{w \in N(u) \cap N(v)} \frac{1}{d(w)} \tag{3.12}$$

Definition 24 (Jaccard Index)**.** Let $s_{uv}^{Jaccard}$ be the Jaccard index [Jac01] over nodes u and v. It is essentially a variation of the common neighbors index, which penalizes for each non-shared neighbor between u and v.

$$s_{uv}^{Jaccard} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{3.13}$$

Definition 25 (Preferential Attachment Index)**.** Let s_{uv}^{PA} be the preferential attachment index [Pri76]. Preferential attachment allows generating scale-free networks [BA99]. With preferential attachment, the probability of a new link being created towards a node u is proportional to its degree $d(u)$. Subsequently, the probability of a new link between nodes u and v is proportional to the product of their respective degrees $d(u) \times d(v)$.

$$s_{uv}^{PA} = d(u) \times d(v) \tag{3.14}$$

Many more indices can be used as heuristics to predict the appearance of an edge between two nodes. We presented this subset of local indices, as combined they allow predicting missing links with high accuracy [SCV19].

[SCV19] Sinha et al., "Systematic Biases in Link Prediction"

GLOBAL AND QUASI-LOCAL HEURISTICS. Global approaches, as their name hint, rely on measures over the global structure of the network to predict connections between nodes. Since they depend on the entire graph, their

algorithmic complexity can become exceedingly high, making them impractical for use on large networks [MBC17]. Global methods are varied in what they measure, for example shortest path between all pairs of nodes [Kat53; Lib05] with the *Katz Index*, probabilities of reaching a node with a random walk [JW02] with *SimRank*, *etc.*

[MBC17] Martínez et al., "A Survey of Link Prediction in Complex Networks"

Definition 26 (Katz Index)**.** Introduced by Katz [Kat53], this metric sums over all the paths that exist between pairs of nodes $u, v \in V$. *Katz* dampens the contribution of long links by a factor of β^l (≤ 1) depending on l, the set of all paths between u and v of length l is denoted $\text{paths}^l_{(u,v)}$.

$$s^{Katz}_{uv} = \sum_{l=1}^{\infty} \beta^l \cdot |\text{paths}^l_{(u,v)}| \tag{3.15}$$

A small value of β favors shorter paths. However, computing *Katz* easily becomes computationally expensive as one needs to consider a significant number of paths between u and v (which could be infeasible for large graphs). *Local Path Index* [LJZ09] is an approximation of *Katz* at path distance of 3 which is consequently more manageable for large networks.

[LJZ09] Lü et al., "Similarity index based on local paths for link prediction of complex networks"

Definition 27 (SimRank)**.** *SimRank* [JW02] relies on the idea that two nodes are similar if they are similar to similar nodes. *SimRank* captures this idea recursively by summing the *SimRank* of all neighbors N of nodes considered.

$$s^{SimRank}_{uv} = \begin{cases} 1, \text{if } u = v \\ \gamma \cdot \frac{\sum_{a \in N(u)} \sum_{b \in N(v)} s^{SimRank}_{ab}}{|N(u)| \cdot |N(v)|}, \text{otherwise} \end{cases} \tag{3.16}$$

$\gamma \in [0,1]$ is a parameter which can be considered as a confidence level or decay factor, it avoids scores such as $s^{SimRank}_{ab} = s^{SimRank}_{uu} = 1$ by dampening the similarity between a and b compared to the similarity of u with itself. As for *Katz*, *SimRank* is computed recursively and can be infeasible for large graphs. Pruning can be performed by only considering nodes up to a radius r, thus reducing time complexity to $O(n^2 d^{2r+2})$.

To alleviate the scalability concerns of global methods, quasi-local indices have been developed. They are presented as an intermediary solution between local indices that rely solely on a node and its neighbors, and indices capturing global topology. Some of these approaches have access to the whole graph but limit the distance at which to compute similarity to keep time complexity reasonable. Subsequently, for example, random walks from global approaches are rendered local with a limited number of hops [TFP06; LL10; LLC10] such as *Local Random Walk*, and limiting the length of paths to be explored [LJZ09] with *Local Path Index* to have a lower complexity. Quasi-local similarities mobilize information from different levels of organization of the network, they rely both on local information and broader topology between two nodes.

Similarity measures provide a straightforward representation of nodes' similarity. Furthermore, similarity-based approaches are interpretable, we know what formula was used to rank the most probable pairs of nodes to be linked. Moving on from similarity approaches, probabilistic methods are also contenders for link prediction, subsequently, we introduce the broad family of probabilistic methods, their advantages, and limits.

▸ PROBABILISTIC APPROACHES

Classifier-based methods, probabilistic models?
Classifier-based methods may also be probabilistic in the sense that they learn to discriminate between existing and non-existing links by learning the distribution of a train set and trying to predict based on the parameter of the model whether a link should exist.

Based on the graph structure, they optimize an objective function and fit parameters to model the network. From these rules and parameters, models can then estimate a posterior probability $P(A_{uv} = 1|\Theta)$, Θ being the set of parameters of the model. Probabilistic models go further than those relying solely on pairwise similarities, since they grasp the distribution of links within the network. *Link prediction* with statistical and maximum likelihood models has been proposed at multiple instances [CMN08; WSP07; Kuo+13; Wan+15; Sta+18; Wil16]. However, most approaches are parameter-dependent and need complex hyperparameter tuning or more information in the form of node or link label [Kum+20]. Furthermore, fitting the distribution of a network can come with a significant time complexity on large networks, which renders their application complex when millions of nodes need to be processed [Kum+20].

[Kum+20] Kumar et al., "Link prediction techniques, applications, and performance"

Dimension reduction approaches are helpful with large networks as a way to summarize structure and latent information in networks and provide rich representations to predict links.

▸ DIMENSION REDUCTION APPROACHES

Dimension reduction or graph embedding models[9] aim to summarize the topology of a network in lower dimension latent space than the adjacency matrix. The objective is to project nodes in a vector space such that similar nodes a close together in the target space. The difference with similarity-based approaches is that the similarity between nodes is not explicitly described. We have thus far covered graph embedding methods in Section 2.1.3 and Section 3.1.1. As with similarity measures, graph embedding can be used with *link prediction* in two manners. Either use similarities between nodes to rank the most probable pairs and select them, or use the representation as input of machine learning algorithms to decide whether the link should exist [Kum+20; Mak+21]. The setting we present Section 3.3.2 relies on graph embeddings and performs *link prediction* as a problem of binary classification of edges based on the representation of the nodes at their extremities.

[9] refer back to Section 3.1.1 for a more in-depth presentation of the graph embedding methods employed in this work

Other approaches provide similar frameworks as dimension reduction, and process the network before using machine learning approaches to predict links.

▸ OTHER APPROACHES

Preprocessing methods can help *link prediction* by reducing noise in the network and only keeping relevant information before applying, for example, a binary classification approach. As such, simplifying the structure of the network can be a useful step before predicting links. Kunegis et al. [KLB09]

[KLB09] Kunegis et al. "The Slashdot Zoo"

try to solve the low-rank approximation, which aims at minimizing a cost function between the adjacency matrix $\mathcal{A}$ of the network and an approximation of lower rank of $\mathcal{A}$–usually obtained after applying *Singular Value Decomposition* (SVD). Preprocessing a graph for *link prediction* can also be as simple as removing the weakest links according to a similarity index up to a γ threshold [Lib05].

Adapting similarity indices with a meta approach can also be used to help *link prediction*. Liben-Nowell [Lib05] in *Unseen Bigrams* make use of the distributional hypothesis and the fact that similar items tend to appear in similar contexts (*i.e.*, from "*hairy cat*", "*furry cat*", "*hairy dog*" we can infer "*furry dog*" as an unseen bigram by substituting a node by its more similar nodes in similarity metrics). Thus, a similarity metric can be bootstrapped by another similarity index. As a consequence, common neighbors can be adapted by considering the most similar nodes to u according to a similarity metric s of choice:

[Lib05] Liben-Nowell "An Algorithmic Approach to Social Networks"

$$s_{uv}^{CN} = |s_u \cap N(v)| \tag{3.17}$$

Predicting links is a complex task which attracts a lot of attention, it helps demonstrate whether a model suitably captures similarities between nodes. However, more topological information can be extracted from a network. We present, in the next section, the extent of topological features that we may expect node embedding models to capture.

Topological features prediction

Link prediction can help assess whether similarities between node representations can help predict their connections. However, it is only one aspect of a network's topology, namely the node-node similarity. Naturally, we wonder about the extent of additional topological information embedded in node vectors. More precisely, are embedding methods able to grasp local organization as well as more global structural information influenced by patterns away from the node and its neighbors?

We study network topology at three levels: microscopic (microscale), mesocsopic (mesoscale) and macroscopic (macroscale).

Definition 28 (Microscopic-level)**.** Microscopic-level interactions are limited to local connections between a node and its neighbors. What microscale analysis informs about the network is local topology and node to node interaction.

Definition 29 (Mesoscopic-level)**.** Mesoscopic-level is the intermediary level between microscopic and macroscopic. Mesoscopic-level interactions happen within groups of connected nodes. These groups can, for example, be cliques, subgraphs, communities, or groups of nodes that exhibit similar patterns of connection and organization.

Definition 30 (Macroscopic-level)**.** Macroscopic-level organization involves studying global properties and overall characteristics of the network taken as a whole.

[Bon+17] Bonner et al. "Evaluating the quality of graph embeddings via topological feature reconstruction"

Bonner et al. [Bon+17] probe the capacity of graph embedding models to capture topological features. To this end, they employ a classification approach to predict four node features: *PageRank* score, *Degree Centrality*, *Clustering Coefficient* (Definition 10) of a node, and *Number of Triangles*. If these topological features are latent in the embedding space, then one expects the classification accuracy to be high.

Definition 31 (Degree Centrality)**.** Let $G = (V, E)$ be a graph with V the set of nodes and E the set of vertices. The degree of node $u \in V$ is such that for the set of neighbors of u, $N(u)$:

$$d(u) = |N(u)| \tag{3.18}$$

Simply put, the degree of a node is the count of its neighbors. Degree centrality stems from the idea that if a node is connected to many other nodes, it is probably central and important in its structure. For example, in society, people with many connections are sometimes considered important.

Definition 32 (PageRank Centrality)**.** Brin and Page's [BP98] *PageRank* allows assessing the local importance of a vertex within a graph. Let $d^+ = |N^+|$ be the out degree of a node obtained from the set of outgoing neighbors N^+, and N^- the set of incoming neighbors with $d^- = |N^-|$ the indegree. In an undirected graph, we consider $N^+(u) \equiv N^-(u) \equiv N(u)$ and $d^+(u) \equiv d^-(u) \equiv d(u)$. $n = |V|$ is the total number of nodes and d is a damping factor.

$$PR(u) = \frac{1-d}{n} + d \sum_{u \in N^-(u)} \frac{PR(u)}{d^+(u)} \tag{3.19}$$

PageRank alleviates the sensibility of *Katz* to high centrality nodes. In *Katz Centrality*, a very central node may greatly influence the centrality of a neighbor. *PageRank* dampens this phenomenon with the degree of the node. A simple example of *Pagerank*'s fix to *Katz* is that not every individual that knows a famous person is also well known. Furthermore, *PageRank* is more scalable than *Katz* that is frequently approximated as with the *Local Index* [LJZ09].

Definition 33 (Number of Triangles)**.** The number of triangles containing u.

$$TR(u) = \Phi \tag{3.20}$$

The number of triangles is closely related to the clustering coefficient that counts the number of connected neighbors over the maximum if all

were connected. Subsequently, we will prefer using the clustering coefficient.

As we stated before, the conjecture in [Bon+17] is that if a graph embedding model can grasp these features and convey them in its vectors, then it should be possible to predict back these topological features through the use of a classification algorithm. The first thing that comes to mind when considering predicting features is to have a regression approach. In regression, the goal is to relate one variable (the topological feature, *e.g.*, *degree*, n*ode clustering coefficient*) to a set of explanatory variables (the vector representing a node). However, [Bon+17] choose a classification setting, thus binning values for the topological features with a histogram. They employ *Stochastic Gradient Descent* (SGD) and *RMSProp* to classify from the embedding into the topological categories created. They show that `node2vec` is able to grasp information related to *Degree Centrality*, *Clustering Coefficient* and *Page Rank*.

Salehi Rizi and Granitzer [SG17] also study the capacity of embedding models to predict topological features. They focus on *Degree Centrality*, *Closeness Centrality*, *Betweenness Centrality* and *Eigenvector Centrality* that we describe shortly.

[SG17] Salehi Rizi and Granitzer "Properties of Vector Embeddings in Social Networks"

Definition 34 (Closeness Centrality)**.** Closeness Centrality measures how central or peripheral a node is with regard to the rest of the nodes in the graph. Let $CC(u)$ be the closeness centrality of node $u \in V$ with $dist(u,v)$ the length of the shortest path between u and v, and $n = |V|$ the total number of vertices in the graph:

$$C_C(u) = \frac{n-1}{\sum_{v \in V} dist(u,v)} \tag{3.21}$$

The closer to 0 the closeness centrality is, the more peripheral a node is, inversely, the higher the closeness centrality (bounded by 1), the more central a node is. In short, the intuition behind closeness centrality is that the more central a node is, the more quickly they can reach other areas of the network (or be used as a way to reach another area).

Definition 35 (Betweenness Centrality)**.** Betweenness Centrality [Fre78] measures how important a node is with regard to its connections and the number of paths between nodes it enables. To put it more simply, betweenness centrality is higher for "hub-like" nodes that act as shortcuts within the network. Let $\sigma(vw)$ be the number of shortest paths from v to w, and $\sigma(v,w|u)$ the number of paths that hop through u. The higher $\sigma(v,w|u)$ is, the more central the node.

$$C_B(u) = \sum_{v \in V, w \in V, w > v} \frac{\sigma(v,w|u)}{\sigma(v,w)} \tag{3.22}$$

Definition 36 (Eigenvector Centrality)**.** Eigenvector centrality incorporates the centrality of its neighbors, similarly to PageRank that takes into account the PageRank score of its neighbors. The eigenvector centrality of a node u is a sum of the eigenvector centralities of its neighbors. λ is a fixed constant.

$$EC(u_i) = \frac{1}{\lambda}\sum_{j=1}^{n} A_{ij} EC(v_j), \tag{3.23}$$

[Bon+17] Bonner et al. "Evaluating the quality of graph embeddings via topological feature reconstruction"

[SG17] Salehi Rizi and Granitzer "Properties of Vector Embeddings in Social Networks"

Contrary to Bonner et al. [Bon+17], Salehi Rizi and Granitzer [SG17] employ a regression approach to predicting the topological features previously described. They do so with a simple feed-forward neural network that takes graph embedding vectors as input, through a fully connected layer and out to a sigmoid activation function that outputs a value between 0 and 1. We use the same setting for our regression experiments Section 3.3.3 by using a linear regression which in essence is similar to the approach employed in Salehi Rizi and Granitzer [SG17] except for the sigmoid activation in output.

Based on the predictions produced with their regressor, they conclude that the graph embedding models they investigate among which `Deepwalk`, `node2vec` and `LINE` can help predict closeness centrality accurately. On the other hand, they fail to provide enough information to accurately predict degree, betweenness, and eigenvector centralities.

Probing community information

An interesting feature of most real-world networks is their structuration in communities. As we described before, communities are dense groups of nodes that are more connected to each other than with the rest of the network. Community detection algorithms allow extracting an approximate partition of nodes into communities. For some networks, ground-truth communities are known and labeled. `Cora` is labeled according to seven categories representing the machine learning method employed in each paper of the network, `Citeseer` classifies papers into six categories corresponding to the field of computer science each paper is part of. `Email-eu` categorizes email exchanges between individuals based on which department within the research institution the network models. We write $\mathcal{L}$ the set of classes.

[Tan+21] Tandon et al. "Community detection in networks using graph embeddings"

Tandon et al. [Tan+21] investigate the capacity of embedding models to cluster nodes into communities when used as input to a *k-means* clustering algorithm. Based on artificial graphs with a corresponding community structure [LFR08], they investigate the capacity of different graph embedding techniques to accurately cluster nodes into communities in an unsupervised manner. Among these methods, they benchmark matrix-based methods such as `Laplacian Eigenmap` and `HOPE`, random-walk-based methods such as `Deepwalk` and `node2vec`, or node similarities with `LINE`. They come to the conclusion that, while embedding models are able, with a clustering algorithm, to group together nodes into communities, they do not perform better than `Louvain` and `Infomap`. Furthermore, graph embedding models need to be extracted, unlike community detection algorithms that are ap-

plied directly to the graph and have thus a lower complexity. The overhead of learning a representation and then applying a clustering algorithm seems unnecessary when community detection algorithms perform well.

Perozzi et al. [PAS14] and Bonner et al. [Bon+17], and Bhowmick et al. [Bho+20] adopt a supervised learning approach to predicting community labels for nodes. Instead of relying on a clustering algorithm, they employ classifiers. Each classifier is trained from a subset of node vectors, and their target community label $C_i \in \mathcal{L}$. Then a disjoint test set is used to evaluate the capacity of the classifier trained from the embeddings to predict the correct community.

[PAS14] Perozzi et al. "DeepWalk: Online Learning of Social Representations"

[Bho+20] Bhowmick et al., "LouvainNE: Hierarchical Louvain Method for High Quality and Scalable Network Embedding"

3.2.2 *Networks*

Before we can proceed with our experiments, let us introduce the networks which will be used throughout the evaluation of graph embedding methods. We chose five real-world graphs composite in fields and sizes ($n = |V|$ and $m = |E|$). The variety in network size allows studying the scalability of our approach regarding the downstream evaluation tasks. For the sake of our experiments, we consider each graph as undirected and extract their largest connected component.

a. `Citeseer` (`Cts`; $n = 2,110$; $m = 3,720$) and `Cora` ($n = 2,485$; $m = 5,069$) are networks of scientific citations.

b. `Email-eu` (`Eu`; $n = 986$; $m = 16,687$) is a sender-recipient email network within a European research institution.

c. `arXiv` ($n = 17,903$; $m = 196,972$) is a co-authorship network of articles published on ArXiv in the Astrophysics category between January 1993 and April 2003.

d. `Facebook` (`Fb`; $n = 63,392$; $m = 816,831$) is a graph of user friendships from Facebook.

3.3 BENCHMARKING SINR

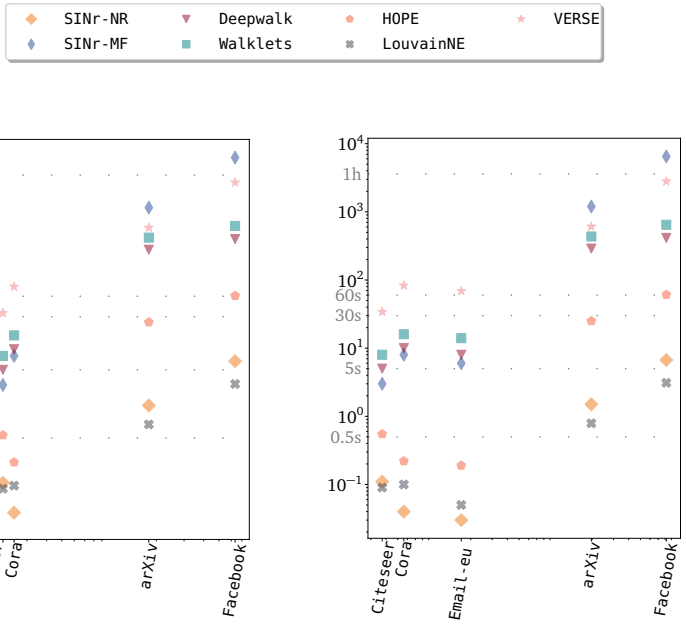
3.3.1 *Network embedding: from the compute standpoint*

	SINr-NR	SINr-MF	DW	WL	HOPE	LouvainNE	VERSE
Cora	**0.04/0.04**	8/161	10/36	16/31	0.22/9	0.10/0.09	83/331
Eu	0.11/0.11	3/50	5/13	8/14	0.55/20	**0.09/0.08**	34/138
Cts	**0.03/0.03**	6/123	8/22	14/25	0.19/7	0.05/0.04	69/274
arXiv	1.5/1.5	1.2K/20K	289/580	435/812	25/600	**0.79/0.72**	610/2.4K
Fb	6.7/6.7	6.5K/116K	414/822	646/1.2K	61/638	**3.1/2.8**	2.8K/11K

TABLE 3.2: Average runtime (left) and total CPU time (right, with parallelism) in seconds over 10 runs. Runtime is computed with two *Intel Xeon E5-2690 v2 3.00GHz CPU* : 20 cores, 250Go RAM.

The runtime of methods is measured over 10 runs Table 3.2. The first value for each method is the average runtime and the second value the total CPU time accounting for multiprocessing on multiple cores. From the compute time standpoint, we can identify two groups of models in Table 3.2. `SINr-NR`, `LouvainNE` and `HOPE` on the one hand, with the lowest compute

times, and SINr-MF, Deepwalk, Walklets and VERSE on the other hand with longer compute times. We have plotted runtime of each method according to number of nodes and number of edges in each network Figure 3.3.



(a) Function of n, number of nodes.

(b) Function of m, number of edges.

FIGURE 3.3: Runtimes (in seconds) of all methods over 10 runs, summary of Table 3.2.

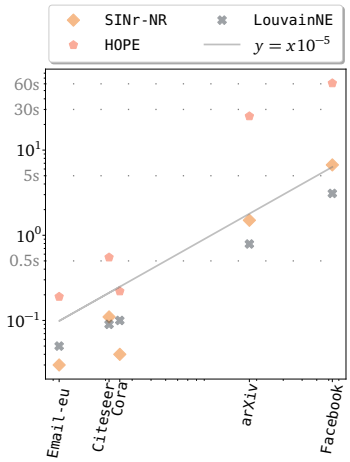


FIGURE 3.4: Fastest methods scale almost linearly with regard to the number of nodes.

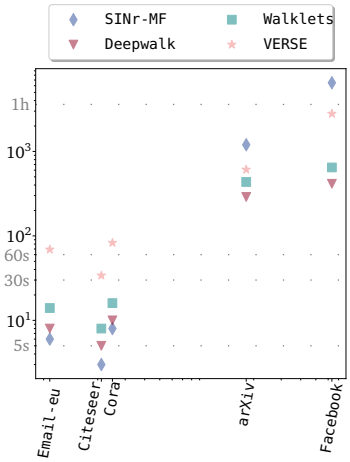


FIGURE 3.5: Slowest methods with regard to the number of nodes.

We can clearly see that SINr-NR, LouvainNE and VERSE have runtimes below 60 seconds, even on large networks such as arXiv ($n = 17.9k, m = 196.9k$) and Facebook ($n = 63.3k, m \approx 816.8k$). LouvainNE and SINr-NR are close to each other, although overall LouvainNE is more efficient. The highest runtime among the most efficient methods is HOPE that takes just over a minute on Facebook. However, when we take a look at the total CPU time, HOPE is the only one among the three most efficient methods to make use of parallelism with roughly 10 minutes on arXiv and Facebook. These low runtimes come down to the low complexity of the algorithms behind these methods. Indeed, the Louvain community detection algorithm has a near-linear time complexity and HOPE's SVD is heavily parallelized. We can see Figure 3.4 that HOPE, LouvainNE and SINr-NR, although with a bit more variation, seem to scale linearly. SINr-NR is close to $y = x10^{-5}$, at least for Cora, arXiv and Facebook.

On the other hand, methods that have a total runtime in the order of 5 to 60 seconds, that can be deemed acceptable on small networks (Eu, Cts and Cora) see their runtime increase to reach 7 minutes for Deepwalk, 10 minutes for Walklets, 46 minutes for VERSE and 1 hour and 48 minutes for SINr-MF. However, SINr-MF is not trained in optimal conditions, as, to compare its runtime on similar hardware, it is not trained on GPU that is the preferred hardware for neural networks. Naturally, the runtime of SINr-MF would be shorter on GPU. Subsequently, its multiprocessing time comes down to 116k seconds, which is equivalent to 32.2 hours. Figure 3.5

presents only the four slowest methods: `SINr-MF`, `Walklets`, `Deepwalk`, and `VERSE`, it is difficult to superimpose a linear function that would fit any of the runtimes. Unlike the fastest methods, they are not quasi-linear. This can yet again be explained by the underlying architectures of these methods that are based on neural networks. Thus, they come with a higher runtime on larger datasets.

Although we can distinguish fast from slower methods, a trade-off might come into play when we evaluate methods' models on downstream tasks. Since runtime remains, in most cases, under the 15-minute mark, we might be able to favor methods with a longer runtime if they have outstanding performances in comparison with faster methods.

3.3.2 *Link prediction with graph embeddings*

▸ LINK PREDICTION SETTING

Our approach to *link prediction* is classification-based using graph embeddings as input representation. This approach differs from *link prediction* on dynamic networks as instead of predicting whether a link is going to appear at a later date in the network, we remove edges and try to predict if they exist. We are thus in a binary classification setting [Ou+16; Kum+20; Mak+21]. Employing a classifier to predict links means that we need a train and test set. The large majority of networks are far from complete[10], as a consequence, if we only sample existing links, the classifier will not be able to predict missing links. Our train and test sets are sampled following a simple process. Let $G = (V, E)$ be an undirected graph, U the universal set containing $\frac{n(n-1)}{2}$ possible edges in V and $E^{\complement} = U \setminus E$ and $C_G = \{c_0, \ldots, c_j\}$ the set of connected components of G. We iteratively and randomly sample couples of nodes (edges) from the graph (20%), the constraint for sampling an edge $(u, v) \in E$ is that $|C_{G(V,E\setminus(u,v))}| = |C|$, *i.e.*, the number of connected components should not increase if we remove (u, v) from E. Sampling (u, v) into the test set S_{test_1} means removing it from graph G thus creating a graph $G\prime = (V\prime, E\prime)$, $E\prime = E \setminus S_{test_1}$. Our positive examples (existing edges) to train a classifier are sampled, the positive training set is $S_{train_1} = E'$ and our positive test set is S_{test_1}. We sample the same amount of negative (non-existing edges) training examples in $E^{\complement}$ thus obtaining a negative training set S_{train_0} and test set S_{test_0}. After concatenating and shuffling these disjoint sets, we obtain a training set S_{train} and a test set S_{test} that can be used with a classifier.

[10] *link prediction* on a complete graph would obviously not be relevant.

After testing multiple classification algorithms (logistic regression, random forest), we opted for `XGBoost`. Chen and Guestrin [CG16]'s `XGBoost` is an ensemble classification algorithm that stands for *Extreme Gradient Boosting*. An ensemble algorithm combines multiple *weak learners* to come up with a decision. In the case of `XGBoost`, the weak learners are an ensemble of decision trees. Each decision tree is trained on a subset of the training data, the predictions from these numerous decision trees are then combined to come up with a final decision. The strength of boosting is to iteratively adapt the parameters of a new tree based on the residuals (prediction error) of the previous tree. Gradient boosting adds a gradient descent process to

[CG16] Chen and Guestrin "XGBoost: A Scalable Tree Boosting System"

boosting to adapt the parameters of the next predictor. `XGBoost` is an implementation of gradient boosting with additional regularization and parallelization strategies that make it an efficient classifier with good accuracy.

Our *link prediction* approach aims at evaluating vector representations of networks, and thus, we selected six baseline algorithms to benchmark our `SINr-NR` and `SINr-MF` approaches along with. `HTS` is the combination of heuristics used in Sinha et al. [SCV19] we described it at the beginning of Section 3.2.1 and it is computed for each pair of nodes in S_{train} and S_{test}: *Common Neighbors*, *Adamic Adar Index*, *Resource Allocation Index*, *Jaccard Index* and *Preferential Attachment Index*. On the front of graph embedding algorithms, we use `Deepwalk` and `Walklets` which are based on random walks, in combination with `SGNS`, `HOPE` which factorizes a vertex similarity matrix, `LouvainNE` which relies on `Louvain` to derive a hierarchy of communities and a node representation based on this hierarchy. `VERSE` tries to reconstruct the distribution of a chosen similarity measure for each vertex using a single layer neural network. Our embedding models all provide one vector per node. Since we need an edge representation instead of a node representation in input of our classifier, we are employing the Hadamard (element-wise) product of the vectors representing the nodes at the extremities of the edge considered such that for an edge $(u, v) \in E$ and its embedding matrix U, the input representation $x_{uv} \in S$ of an example in train or test set S is $U_u \odot U_v$.

[SCV19] Sinha et al. "Systematic Biases in Link Prediction"

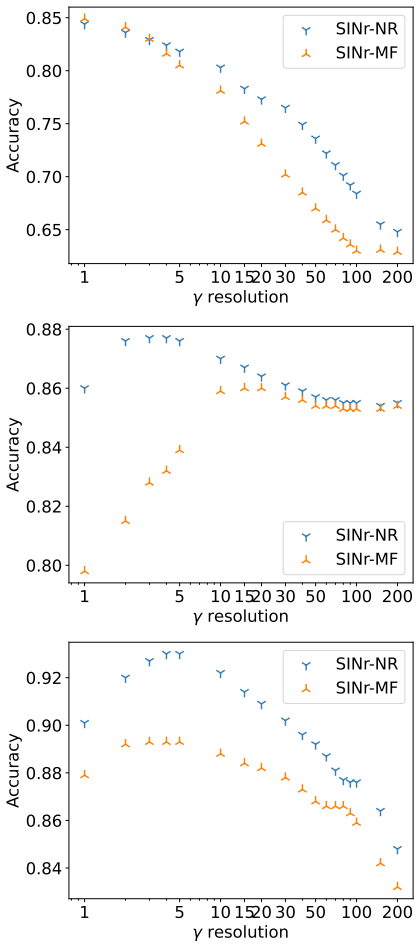


FIGURE 3.6: Average accuracy on Link Prediction for various γ resolution values in `SINr`. Top figure is `Cora`, middle is `Email-eu` and bottom is `arXiv`. Yellow series is `SINr-MF` and blue series is `SINr-NR`.

LINK PREDICTION RESULTS

Table 3.3 presents the average accuracy after training 50 instances of each model and applying our *link prediction* framework, the figures reported are significant to 10^{-3}.

	SINr-NR	SINr-MF	HTS	DW	WL	HOPE	LouvainNE	VERSE
Cora	0.845	**0.850**	0.728	0.708	0.773	0.760	0.827	0.809
Eu	0.860	0.798	**0.885**	0.790	0.876	0.847	0.752	0.852
Cts	0.877	**0.879**	0.755	0.736	0.820	0.832	0.863	0.859
arXiv	0.930	0.893	**0.980**	0.912	0.954	0.914	0.847	0.957
Fb	0.915	0.892	**0.930**	0.859	0.920	0.900	0.847	0.917

TABLE 3.3: Average accuracy for the link prediction task over 50 runs.

Overall, the two implementations of `SINr` (`SINr-NR` & `SINr-MF`) are on par with competing methods, especially on small networks.

`SINr-NR` is consistently better than `HOPE` and `Deepwalk` (`DW`) across all networks and close to `Walklets` (`WL`). The matrix factorization approach of `SINr` , `SINr-MF` is leading on the two smallest networks (`Cora` & `Cts`) closely followed by `SINr-NR`. It seems that `SINr-MF` is less efficient when networks become larger, the results would benefit from more training epochs on `arXiv` and `Facebook`. However, `SINr-MF` optimization is slow, and it appears that despite its apparent performance on smaller scale graphs, it is not adapted for large networks. `SINr-NR` remains competitive with the leading method, `Heuristics` (`HTS`), even on networks of higher magnitude. `SINr-NR` is third behind `HTS` and `Walklets` but with a significantly lower compute time of 6 seconds against the 646 seconds needed for `Walklets` for a 0.005

point increase. If we look further than raw performance, we may consider trading some performance for a lower runtime.

`SINr`'s γ multi-resolution parameter, inherited from `Louvain`, allows controlling the dimension of vertex embedding vectors. Higher values of γ results in vectors of higher dimension. Depending on the size of the network at hand, it might be beneficial to have more dimensions in the embedding space to better represent the complexity of the network structure. To that end, we plot Figure 3.6 the accuracy on the link prediction task according to the γ value for `Louvain`. For the smallest network `Cora`, the maximum accuracy is reached for a gamma value of 1. Our intermediary `Email-eu` network admits a maximum accuracy when γ is 3 for `SINr-NR` when `SINr-MF` needs more dimensions to reach its highest accuracy on link prediction, setting the γ between 10 and 20. On `arXiv`, the best accuracy is reached with a γ set to values around 5. A good rule of thumb is that the larger the graph is, the more dimensions are needed to represent vertices' interactions, and thus a higher γ value needs to be chosen. Based on these results, the γ value chosen for `Cora`, `Citeseer` and `Email-eu` is 1, when γ is set to 5 for `arXiv` and `Facebook` for all the experiments on these networks including results presented Table 3.3.

3.3.3 *Predicting graph features*

▸ FEATURE PREDICTION SETTING

Based on the topological features prediction of Salehi Rizi and Granitzer [SG17], we also investigated the topological content of embedding models. Especially in the case of `SINr`, we wanted to assess whether relying on communities, which encompass local organization within each community as well as more global organization when considering the interrelations between nodes of differrent communities, is beneficial to topological feature prediction. To that end, we evaluated `SINr` along with renowned baseline graph embedding models following Salehi Rizi and Granitzer [SG17]'s approach. Instead of relying on a neural network architecture, we chose to employ a more simplistic linear regression. In our context, the linear regression has the goal of finding the best fitting β coefficients to minimize the difference between the graph feature y and the predicted value by the linear equation. In this setting, each β_i coefficient is acting on a value e_i^n of a component of node embedding vector e to predict $\hat{y}_i$ so that it is as close as possible to y_i in Equation (3.25):

[SG17] Salehi Rizi and Granitzer "Properties of Vector Embeddings in Social Networks"

$$\hat{y}_i = \beta_0 + \beta_1 e_i^1 + \beta_2 e_i^2 + \ldots + \beta_n e_i^n \tag{3.24}$$

$$\beta_0, \beta_1, \beta_2 \cdots, \beta_n = \underset{\beta_0, \beta_1, \beta_2 \cdots, \beta_n}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \hat{y})^2 \tag{3.25}$$

As with the similarities described Section 3.2.1, we can classify topological features in the microscopic, mesoscopic and macroscopic levels. We thus focus our evaluation on three topological features that we consider respectively micro, meso and macroscale: Degree, Clustering Coefficient, and

PageRank. This allows us to gain insight into the quantity of topological information conveyed by the embedding vectors, as well as the distance from the node at which graph embedding models can grasp them. Namely, do models embed local information closer to the node better, are communities in `SINr` helping with prediction on mesoscopic features since they lie at the same level of organization, or is global structure also available in vectors.

The evaluation procedure is the same with all topological features studied. We start by extracting ten embedding models for each of our methods (five baseline methods and our `SINr` approaches). We then randomly split the data in a training set consisting of 80% of the nodes in the graph and their respective target feature value (the degree of each node, its clustering coefficient, or PageRank index). The remaining 20% constitutes the test set used to assess whether the embeddings can provide a correct representation to predict these topological features on unseen data.

We evaluate performance of prediction using the R-squared (R^2) coefficient of determination (Definition 37).

Definition 37 (R-squared (R^2))**.** The R-squared determination coefficient measures how good a regression model fits the observed data. It normalizes the *Mean Squared Error* (MSE) by the variance of data. In R^2, $SS_$residual (Sum of Squared Residuals) is a measure of the difference between the actual target values y and the predicted values $\widehat{y}$ of the regression model, such that $SS\_\text{residual} = \sum(y - \widehat{y})^2$ which is also the MSE. $SS_$total (Total Sum of Squares) is the sum of differences between the actual value of y, and the mean $\overline{y}$ such that $SS\_\text{total} = \sum(y - \overline{y})^2$ is the variance of data. The higher the R^2 (bounded by 1) value, the better the regression model fits the variability in the sample. Since we compute R^2 on data that wasn't previously seen, it is a good indicator to assess whether a vector space can provide enough information to learn a good prediction model.

$$R^2 = 1 - \frac{SS\_\text{residual}}{SS\_\text{total}} \tag{3.26}$$

Since we have described our evaluation procedure and our metric, we can now predict Degree Centrality, Clustering Coefficient and PageRank Centrality for graph embedding method.

▸ Feature prediction results

Degree Centrality

We predict the *Degree Centrality* of nodes in the graph. We fit the linear regression from the embedding of nodes, the target value for each node is extracted from the degree sequence of G. The set of vertices is split randomly, with 80% of the vertices in the training set and the remaining 20% in the test set. The regression is performed 50 times with different splits for each model , and the average R-squared (R^2) coefficient of determination is presented in Table 3.4.

	SINr-NR	SINr-MF	HOPE	Deepwalk	Walklets	LouvainNE	VERSE
Cora	**1.000**	-0.586	0.773	-0.122	-0.028	-0.067	0.227
Cts	**0.998**	-0.077	0.741	0.140	0.342	0.037	0.010
Eu	**1.000**	0.022	0.990	0.388	-0.783	-5.183	0.869
arXiv	**1.000**	0.543	0.937	0.419	0.725	0.144	0.585
Fb	**1.000**	0.647	0.935	0.358	0.766	0.091	0.706

TABLE 3.4: R^2 for vertex degree regression.

We can first notice that SINr-NR is suited to fit a model almost perfectly to predict the *degree centrality* of a node. Performances of SINr-NR on the degree regression is the direct consequence of the *ad hoc* embedding weighing method employed. Measuring the relative distribution of weighted degree of each vertex over the network community structure is an indirect encoding of a node's degree. The more diffuse the values in the embedding space, the higher the potential degree of the vertex, since it is connected to many vertices in varying communities. HOPE is second and VERSE third, with varying performances across networks. Although their results are, on average lower than those achieved by SINr-NR, *degree centrality* information is nevertheless present in the representations produced. Deepwalk and Walklets, which are both random-walk-based, do not manage to provide quality embedding to fit a model that accurately predicts *degree centrality*. This result further confirms what Salehi Rizi and Granitzer [SG17] concluded following their experiments, these models fail to provide enough information to predict node degree. SINr-MF achieves subpar performances on Cora, Citeseer and Email-eu but is in keeping with VERSE on arXiv and Facebook. It appears that SINr-MF matrix factorization's approach is not efficient at embedding node degrees. SINr-MF is less prone to embedding degree information, since it aims at finding the transition matrix between the network's adjacency matrix and the community membership matrix. Thus, although we are relying on the community membership of the nodes, unlike SINr-NR with the Node-Recall measure, we do not explicitly rely on the distribution of nodes' degrees to derive a representation. Yet, high degree nodes tend to have a high dispersion through communities, which may help prediction. Subsequently, these results are most probably dataset-dependent and a solely encouraging.

[SG17] Salehi Rizi and Granitzer "Properties of Vector Embeddings in Social Networks"

We have demonstrated that embedding models, to some extent, manage to encapsulate local interactions to predict a microscale feature: the *degree centrality*. We have observed that in most cases SINr-NR can provide classifiers with a representation able to help predict perfectly the *degree centrality* of a node. Now, if we take a step back from node-node interactions to the mesoscopic-level, we can study how the local structure around a node is embedded in its representation.

Clustering Coefficient

Clustering coefficient centrality is an indication of how clustered the neighborhood of a node is. The more intertwined the neighbors of a node are, the higher will its *clustering coefficient* be (see Definition 10). Regarding this measure of local connectivity, we wonder whether, like *degree central-*

ity, graph embedding models convey information to predict *clustering coefficient*. We adopt the same setting as with *degree centrality*, we extract the *clustering coefficient* of each node and fit a linear regression. We present Table 3.5 the averaged results for R^2 coefficient over 50 runs for each model.

	SINr-NR	SINr-MF	HOPE	Deepwalk	Walklets	LouvainNE	VERSE
Cora	0.007	0.027	-0.161	0.0001	-0.020	**0.052**	-0.001
Cts	**0.003**	-0.411	-0.493	-0.015	-0.006	-0.950	0.001
Eu	0.060	-0.011	-0.012	**0.065**	-1.756	-35.379	-0.078
arXiv	0.247	0.236	0.126	0.325	**0.344**	0.036	0.260
Fb	0.027	0.046	0.017	0.132	**0.167**	0.009	0.036

TABLE 3.5: R^2 for vertex clustering coefficient regression.

On average, we observe that the determination coefficient scores for the *clustering coefficient* prediction are low. The highest value of 0.344 for R^2 is achieved by `Walklets` on `arXiv`. Still, this highest value is far below what we observed for the Degree Centrality regression. `SINr-NR` is the sole model to always present a positive R^2 coefficient regardless of the network. It seems as though it is difficult to come with a hard conclusion on small networks (`Cora`, `Cts` and `Eu`) where R^2 values are really low and performance is variable from one network to the other. On larger networks, random-walk-based methods obtain more encouraging results followed by `VERSE` and the two implementations of `SINr`. None of the model manages to emerge as a clear contender for vertex *clustering coefficient* prediction.

From these observations, one can only wonder why the *clustering coefficient* cannot be easily predicted from the vertex embedding when degree is a property well embedded in their representation. *Clustering coefficient* is a measure of the connection among vertex neighbors and is thus rather local to a vertex. However, this interaction is happening at an edge away from the node considered and might be harder to model in the representation. Random walk based models might capture more of this structure on larger graphs with many edges. `SINr`'s usage of community structure might also explain part of its scores on coefficient regression. Having two nodes in the same community is a strong signal for them to be connected. Nevertheless, this structure does not seem to provide enough information to accurately fit a model that predicts *clustering coefficient*. Moving back from the mesoscopic-level to the macroscopic-level, at which the centrality is influenced by the topology of the whole graph, we measure the capacity of embedding models to provide information about *PageRank centrality*.

PageRank Centrality

PageRank is a local measure of the centrality of a node, influenced by the centralities of every node in the network (see Definition 32). As such, we classify it as a macroscale measure, even though it is measured for each individual node. Our target *PageRank* values are extracted from each node in each network. We keep the same regression setting as for *Degree* and *Clustering Coefficient* regression, averaging the R^2 values over 50 runs as presented Table 3.6.

	SINr-NR	SINr-MF	HOPE	Deepwalk	Walklets	LouvainNE	VERSE
Cora	**0.980**	-0.610	0.697	-0.191	-0.026	-0.092	0.185
Cts	**0.949**	-0.176	0.346	-0.050	0.182	-0.123	-0.125
Eu	**0.991**	0.013	0.959	0.347	-0.960	-0.271	0.856
arXiv	**0.955**	0.518	0.736	0.295	0.693	0.039	0.520
Fb	**0.961**	0.618	0.739	0.213	0.720	0.017	0.652

TABLE 3.6: R^2 for vertex PageRank regression.

Contrary to *Clustering Coefficient* centrality prediction, *PageRank centrality* prediction is unequivocal regarding the capacity of graph embedding models to provide a base for *PageRank* prediction. *PageRank*'s regression results give a clear advantage to SINr-NR that outperforms all other methods. HOPE is the second-best method for predicting *PageRank*, followed by VERSE. Except for the Facebook (Fb) network, Deepwalk, Walklets, LouvainNE and SINr-MF perform poorly on *PageRank* prediction. SINr-NR's domination on *PageRank* prediction may be correlated to its ability to predict degree. As a matter of fact, *PageRank* score is influenced both by the degree of a vertex at the microscopic level but also by the importance of its neighbors in the network. Both of these properties are at the heart of SINr-NR where we distribute the degree over the community structure, using the neighbors of the node in different communities. Thus, the stronger a value for a community may be, the more it may contribute to the *PageRank* score of a node. Subsequently, SINr-NR and HOPE, to a lesser extent, can embed a macroscopic property of a network, the *PageRank*.

3.3.4 *Community-centered evaluation*

Similarly to what was done with topological feature regression, the goal of these two tasks of *node community clustering* and *node multilabel classification* is to study whether topological information related to communities is present in the vector representations. We thus benchmarked state-of-the-art graph embedding methods along with our SINr algorithms to quantitatively assess this property. There is scrutiny around the evaluation of community detection and trying to predict so called *"ground-truth"* communities. The partitions of nodes we use are based on metadata associated with each node. We previously mentioned Chapter 1 that metadata-based communities may not match the communities detected by a community detection algorithm. This may be the consequence of diverse factors: metadata is not relevant to the network structure, metadata and uncovered partition may capture different aspects of topology, or the network may not exhibit a community structure [LC14; PLC17]. However, in our case, we do not evaluate community detection but rather the ability to find a clustering of nodes in groups. This task is performed with a clustering algorithm applied to the vectors and a classifier. Thus, we can perform our experiment, with the caveat that partitions based on metadata may not correspond to partitions that would be detected by community detection algorithms. We start with community clustering and then move on to multilabel classification.

[LC14] Lee and Cunningham, "Community detection: effective evaluation on large social networks"

[PLC17] Peel et al., "The ground truth about metadata and community detection in networks"

Community clustering

To cluster nodes into their respective communities, we need three things: a representation of our nodes, a clustering method and an evaluation metric. Naturally, our node representation will be obtained after applying each graph embedding method we previously investigated, namely, `SINr-NR` and `SINr-MF`, `HOPE`, `Deepwalk` `Walklets`, `LouvainNE` and `VERSE`. We chose spectral clustering as a clustering algorithm, as it allows us to alleviate dimension discrepancies between models. Indeed, `SINr` models have a varying number of dimensions compared to their node embedding counterparts, since the dimension of vectors depends on the number of communities detected by `Louvain`. As such, spectral clustering allows us to cluster nodes based on a node similarity matrix that is computed from the embeddings. This similarity matrix is based on the cosine similarity between the embedding vectors of each node:

Definition 38 (Embedding pairwise Cosine Similarity)**.** Let S_{cosine} be the cosine similarity such that for two nodes $u, v \in V$ and their respective embedding vectors X_u and X_v:

$$S_{cosine}(X_u, X_v) = \frac{X_u \cdot X_v}{\|X_u\| \, \|X_v\|} \tag{3.27}$$

Applying the cosine similarity for all pairs of vectors representing each node, we obtain an affinity matrix $\mathcal{A}$ such that for a pair of nodes (u, v):

$$\mathcal{A}_{uv} = S_{cosine}(X_u, X_v) \tag{3.28}$$

Using the affinity matrix based on cosine similarity, we can apply the spectral clustering algorithm. Spectral clustering relies on the eigenvectors of the graph Laplacian of $\mathcal{A}$ [11] to provide an alternate vector space in lower dimension that can then be clustered using a partition algorithm. In our case, we use the *k-means* algorithm, and thus our clustering method is *spectral k-means*. We evaluate the performances of clustering nodes from their embeddings with the *Normalized Mutual Information* (NMI) that is defined as follows:

[11] see Equation (2.9)

Definition 39 (Normalized Mutual Information)**.** Normalized Mutual information or NMI is a variant of the Mutual information (MI). NMI aims at measuring the similarity or dissimilarity between two partitions. Comparing the ground-truth labels associated to data with the partition obtained following a clustering task, NMI can assess the quality of the partition at hand. NMI combines key elements of information theory:

- **Entropy**: Entropy measures how uncertain or random a random variable is. In our context, the entropy of a partition indicates how mixed or diverse the cluster assignments are. Given the probability $p(x)$ of random variable X taking value

x, the entropy is defined as follows:

$$H(X) = -\sum_{x \in X} p(x) \log(p(x)) \tag{3.29}$$

- **Mutual Information (MI)**: Mutual Information, measures the amount of information shared by two random variables. Regarding our clustering task, the two variables are the ground-truth labels associated with the data, and the clustering labels. Given the joint probability $p(x, y)$ of a variable X taking value x and Y values y

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log\left(\frac{p(x, y)}{p(x) \cdot p(y)}\right) \tag{3.30}$$

  Mutual Information can also be expressed as the combination of the entropies of two variables X and Y:

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \tag{3.31}$$

Normalized Mutual Information combines Entropy and Mutual Information into a normalized measure with $I(X, Y)$ MI between variables X and Y, in our case two different partitions, $H(X)$ is the entropy of variable X.

$$\text{NMI}(X, Y) = \frac{2 \times I(X, Y)}{H(X) + H(Y)} \tag{3.32}$$

NMI values are between 0 and 1. When two partitions X and Y are identical then $\text{NMI}(X, Y) = 1$, if X and Y are completely dissimilar, then $\text{NMI}(X, Y) = 0$

Following this procedure, we partition the nodes of our three graphs that have a ground-truth. If the NMI is high, then the embeddings must convey strong community information. The task is performed 50 times and the average NMI values are presented Table 3.7.

	SINr-NR	SINr-MF	HOPE	Deepwalk	Walklets	LouvainNE	VERSE
Cora	0.364	0.047	0.368	0.020	0.296	0.311	**0.435**
Cts	0.331	0.112	0.322	0.009	0.177	0.147	**0.406**
Eu	0.567	0.266	0.682	**0.703**	0.670	0.644	0.698

TABLE 3.7: Spectral clustering results based on embedding similarity measured in *NMI* between ground-truth labels and predicted clusters.

Regarding unsupervised detection of communities, scores for SINr-NR are close to that of HOPE. SINr-MF does not allow to cluster the embeddings efficiently into communities from the similarity between embeddings. Contrary to Deepwalk, all models seem not to show great discrepancies across networks even though it is the best performing model on Email-eu. VERSE shows abilities to learn useful vectors for community clustering. Although SINr-NR has low results on Email-eu, it manages to be respectively third and second-best model on Cora and Citeseer. The major lesson drawn from these scores is that detecting communities from the similarity between

embedding is a complex task. Let us now consider a more comfortable setting that is also more artificial, since part of the community labels will be known by the classifier. In this setting, the label prediction achieves better results.

Community classification

Following our unsupervised setting to characterize the amount of community information present in graph embedding models, we now adopt a supervised approach. The setting is simple, we split the set of nodes into a train (80%) and test set (20%). The train set is used to train a `XGBoost` classifier to discriminate between communities. The higher the accuracy in predicting the right community for each node in the test set, the more community information can be extracted from graph embeddings. Similarly to previous quantitative evaluations, we perform 50 train and test samplings and train 50 classifiers. The average accuracy for multilabel classification is presented Table 3.8.

	SINr-NR	SINr-MF	HOPE	Deepwalk	Walklets	LouvainNE	VERSE
Cora	0.752	0.716	0.799	0.719	**0.852**	0.831	0.850
Cts	0.690	0.685	0.718	0.630	**0.758**	0.722	**0.758**
Eu	0.432	0.222	**0.721**	0.686	0.700	0.672	0.656

TABLE 3.8: Vertex classification results, average accuracy over 50 runs.

Regarding supervised node community labeling, `Walklets` is the best performing model. On smaller networks (`Cora` and `Citeseer`), `SINr-NR` is ahead of `Deepwalk` and `SINr-MF` has similar performances. However, a more significant gap between the baseline methods and `SINr`'s two approaches appears on `Email-eu` (`Eu`). It appears that `SINr-NR` and `SINr-MF` do not manage to grasp community membership as well as other methods on a network with far more edges and fewer nodes than `Cora` and `Citeseer`. When we slightly increase the γ value for community detection in `SINr` on this task, we obtain more dimensions in the representation space and results improve. When we set $\gamma = 5$, `SINr-NR` and `SINr-MF`, results on `Cora` jump to 0.786 when they increase to 0.718 and 0.694 on `Citeseer`. The strongest increase is observed for `Eu` where changing to a γ value of 5 leads to respective accuracy results of 0.719 and 0.604 for `SINr-NR` and `SINr-MF` thus making `SINr-NR` the second-best model. The γ parameter has an undeniable impact on performance for *vertex community classification*, as for *link prediction* and illustrated Figure 3.6.

3.4 SUMMARY

In this chapter, we have explored graph embeddings and their evaluation. The key information and findings of our work on graph embeddings are the following:

(i) Graph embedding methods adopt diverse approaches to derive node representations: random walks, factorization of adjacency or similarity matrices, methods based on communities, and graph neural networks (`GNN`).

(ii) Although SINr-MF does not quite satisfy our low runtime objective, SINr-NR is faster than other methods (except LouvainNE) on all networks.

(iii) SINr-NR and SINr-MF encapsulate useful information to predict missing links in networks.

(iv) SINr-NR is efficient at capturing information to predict network characteristics (degree, PageRank) at different levels of organization in networks (microscopic, mesoscopic and macroscopic).

(v) SINr-NR and SINr-MF capture useful information to reconstruct communities from node embeddings.

These positive evaluations demonstrate the versatility of SINr-NR to embed useful information for a variety of tasks. Based on these results, SINr-NR seem to be a well-rounded approach for node embedding. Yet, the method is adapted to any data that can be represented as a network and with a detectable community structure. This includes word embedding extracted from word co-occurrence networks. Subsequently, in Chapter 4, we describe word embedding approaches and compare the performances of SINr-NR against competing approaches on evaluations specific to word embeddings.

4

Modeling text: word embedding

Chapter contents

Synopsis

Word embeddings have the objective of capturing the complexity of language and provide accurate representations of words' meaning in vector form. Throughout the last decade, many methods have been developed to tackle this task. They focused mainly on obtaining the best performing models without necessarily considering runtime. Yet, with global warming, taking into account efficiency of computation is in line with sustainability objectives in machine learning. Furthermore, the emergence of new architectures, needing large quantities of data (*Large Language Models*) for training, and thus long training times also motivates the search for more reasoned approaches. Another aspect to take into consideration is that most word embedding models provide opaque representations that cannot be easily audited. Their lack of transparency is problematic in high-stake applications of machine learning where text data is involved. Indeed, word embeddings are often the first brick of these systems. This chapter studies the abilities of `SINr-NR` to overcome these limits, and particularly regarding runtime and performance. To that end, we benchmark `SINr-NR` against competing approaches, measuring runtime and its ability to convey word similarity in representations. We show that, on top of being able to provide graph embeddings, `SINr-NR` is suited to learn word embeddings.

Résumé

Les plongements de mots ont pour objectif de capturer la complexité du langage et fournir des représentations pertinentes du sens des mots sous forme de vecteurs. Dans la décennie passée, de nombreuses nouvelles méthodes ont été développées pour apprendre des plongements de mots. Cependant, certaines limites liées à ces méthodes subsistent. Leur développement s'est concentré sur l'obtention de modèles performants. Cependant, dans le contexte de sobriété énergétique, imposé par le réchauffement climatique, le temps d'exécution est un aspect des méthodes d'apprentissage automatique qu'il est nécessaire de prendre en compte. Or, les grands modèles de langage (LLM) développés ces dernières années mobilisent de grandes masses de données qui impliquent de longs temps d'apprentissage. Par ailleurs,la plupart des méthodes d'apprentissage de plongements de mots fournissent des représentations opaques peu auditables par les humains. Or, ce manque de transparence des modèles est un problème lorsque ceux-ci sont utilisés pour des applications sensibles. En effet, les plongements de mots sont souvent la première brique des chaînes de traitement en NLP. Ce chapitre évalue SINr-NR au regard des limites des modèles précédents et notamment en termes de temps d'exécution et de performance. Pour ce faire, SINr-NR est évalué aux côtés d'approches concurrentes. Nous mesurons notamment le temps d'exécution des méthodes et leurs capacités à fournir une représentation fidèle des mots en préservant leur similarité dans l'espace de plongements. Nous montrons qu'en plus d'être un modèle performant pour produire des plongements de graphes, SINr-NR est adapté pour l'apprentissage de plongements de mots.

In Chapter 3 we have demonstrated the abilities of `SINr-NR` and `SINr-MF` on graph data. This chapter is dedicated to word embedding. Word embeddings aim to uncover a latent space in which similar words are put close to one another and dissimilar words are far from each other. We aim to show in this chapter the versatility of `SINr-NR` and demonstrate that it is well suited to represent text via word-co-occurrence networks. Word embedding quality can be assessed by intrinsic evaluations that rely solely on the internal structure of the model. In the next section, we introduce our evaluation protocol before presenting the baseline models we will use to compare the results obtained with `SINr-NR` and the results of these evaluations.

4.1 TASKS

Evaluating the semantic content of representations and whether they properly model sense as humans mentally represent it is complex. If a model properly conveys the meaning of words, we should be able to design human evaluations to assess the relevance of a particular model. We will see in Chapter 5 that this can be done for interpretability. However, doing so for a large quantity of words, and reliably for several models, requires ground truth datasets that can act as benchmarks. These evaluation tasks are reproducible, and can be applied to all word embedding models. They allow not resorting to human evaluations for every new model, and reduce the cost associated with evaluation.

We introduce hereafter two tasks based on datasets constructed with the help of human annotations that can be performed automatically. They are called intrinsic evaluations because they are aimed at assessing the quality of word embeddings based on internal properties of the latent space uncovered. Furthermore, extrinsic evaluations were already performed on graphs in Chapter 3. Intrinsic tasks do not involve downstream applications but focus on analyzing the embeddings themselves. They can, for example, measure the ability of a model to capture syntactic or semantic features for a given corpus.

▸ WORD SIMILARITY

TASK DESCRIPTION. The *pairwise word similarity* evaluation stems from the early work of Osgood et al. [OST57] and later Rubenstein and Goodenough [RG65] in psycholinguistics to test the distributional hypothesis. These experiments were later reintroduced by Baroni et al. [BDK14] as a way to evaluate the quality of distributional representations. The task relies on human constructed and annotated datasets containing pairs of words. Each pair of words is given a score by annotators: the higher the score, the more similar the two words presented are. For example, we could expect animals such as *"elephant"* and *"cat"* to be closer to each other than to the tool *"hammer"*. Figure 4.1 illustrates the process of the word similarity evaluation, where the score of pairs in the evaluation dataset is compared to the value of *cosine similarity* for the same pair in the word embedding model. The *Spearman correlation* is used as a quality metric. It is computed be-

[OST57] Osgood et al. *The measurement of meaning*.

[RG65] Rubenstein and Goodenough, "Contextual correlates of synonymy"

[BDK14] Baroni et al. "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors"

tween the ratings given by humans in the dataset and the cosine similarity between word pairs in the model. The higher the correlation, the better a model is supposed to convey meaning as mentally represented for humans.

Definition 40 (Spearman's correlation)**.** *Spearman*'s correlation measures the strength and association between two ranked variables (in our case ranked by human rating). This correlation assesses the monotonic relationship between these two variables, whether one variable tends to increase or decrease as the other variable increases (if there is systematic association between these two variables). *Spearman*'s correlation ranges from -1 to 1, where -1 (respectively 1) indicate a perfect negative (respectively positive) monotonic relationship and 0 indicates not monotonic relationship. Spearman's correlation coefficient (ρ), is calculated using the formula:

$$\rho = 1 - \frac{6 \sum_{i \in R} d_i^2}{n(n^2 - 1)} \tag{4.1}$$

where d is the difference between the ranks of corresponding observations in the two variables and i the rank of the first variable, and n is the number of paired observations.

Thus, the *word similarity* assesses the quality of the neighborhood of the vectors: "*tiger*" and "*lion*", or other words with a high similarity score, should indeed be close in the representation space. But it also evaluates the existence of a larger structure in the vector space. Since datasets are comprised of word pairs within a spectrum of scores, they allow observing some kind of distance between immediate neighborhoods : "*lion*" and "*tiger*" may be very close, but "*cat*" should not be very far away, and "*dog*" may follow, or at least be much closer to "*lion*" than to "*hammer*" which all things considered should be far.

w_1	w_2	human rating
tiger	cat	7.35
plane	car	6.31
drink	mother	2.85
forest	graveyard	1.85

Spearman Correlation ×

cosine_sim(w_1, w_2)
0.73
0.65
0.20
0.12

FIGURE 4.1: Example of word similarity rating from the `MEN` dataset and cosine similarity between vectors.

EVALUATION DATASETS. Datasets used to evaluate word similarity have been constructed to evaluate a diversity of words. They exist mainly in English. Many datasets are described in Bakarov [Bak18]. We have selected three datasets that each had features of interest either in terms of word frequency, type of words, part-of-speech and statistical significance.

[Bak18] Bakarov "A Survey of Word Embeddings Evaluation Methods"

a. `MEN` [BTB14] is composed of 3,000 pairs selected among the 700 most frequent words in `UkWac` and `WaCkypedia` corpora combined. Ratings were collected on Amazon Mechanical Turk.

[BTB14] Bruni et al., "Multimodal Distributional Semantics"

[Fin+01] Finkelstein et al., "Placing search in context: The concept revisited"

[Hua+12] Huang et al., "Improving word representations via global context and multipleword prototypes"

b. `WS353` [Fin+01] *WordSim-353* contains 353 noun pairs.

c. `SCWS` [Hua+12] *Stanford Contextual Word Similarity* dataset comprises 2003 pairs of mixed part-of-speech with senses sampled from WordNet [Mil95].

All three datasets comprise pairs of words both representing word *similarity* (approximately synonymy, or at least substitutability like "*cat*" and "*feline*") and word *relatedness* (much broader, encompasses pairs like "*cup*" and "*coffee*"). However, the datasets differ regarding the parts of speech included: `WS353` only includes nouns, while `MEN` and `SCWS` include nouns, verbs, and adjectives.

The *pairwise word similarity* evaluation is our main means of evaluating the intrinsic quality of our representations. Yet, we complete our word similarity evaluation with a *concept categorization* task that is also intrinsic. The *concept categorization* evaluation is complementary to the *pairwise word similarity* evaluation, as it evaluates the capacity of gathering thematically related words from their embeddings. With this evaluation, we go beyond evaluating only pairs of words, but rather the capacity of discriminating groups of related words from their embeddings.

▸ CONCEPT CATEGORIZATION

TASK DESCRIPTION. Concept categorization or word clustering is the text pendant of our community clustering approach previously presented Section 3.2.1. Instead of communities, the goal of *concept categorization* is to properly cluster a subset of selected words from their embedding into preset categories. Since the categories encompass basic-level concepts ("*cat*" and "*dog*" over "*golden retriever*" and "*german shepherd*"), their assessment implies the existence of a larger structure than immediate proximity for substitutable words: topical consistency in regions of the representation space. Categories in datasets range from animals to feelings or legal documents to cite only a few (for example, the following words: "*banana*", "*berry*", "*cruiser*", "*helicopter*" should be put in two categories, edible fruits and vehicles). Clustering is operated on the embedding vectors of the words in each dataset provided by each method. Words are thus clustered into categories from their vectors using the *K-means* and *Hierarchical Agglomerative Clustering* (HAC) algorithms, only the best purity is retained for each run and purities from 10 consecutive runs are averaged.

EVALUATION DATASETS. There are fewer concept categorization datasets used to benchmark word embedding models than for the word similarity evaluation. We chose three datasets that cover different categories of words, either semantic, relative to a specific word category (verbs) or more abstract (concreteness level). The chosen datasets are the following:

[AP05] Almuhareb and Poesio, "Concept learning and categorization from the web"

a. `AP` [AP05] is a categorization dataset constructed with the goal of being balanced in class type, term frequency and ambiguity. The dataset contains 21 different categories.

[BL11] Baroni and Lenci, "How we BLESSed distributional semantic evaluation"

b. `BLESS` [BL11] contains 200 nouns (100 animate, 100 inanimate) from

17 categories (*e.g.*, appliance, bird, vehicle, vegetable).

c. `ESSLI-2008` [ESS08] datasets have been created for the shared tasks of *Workshop on Distributional Lexical Semantics* that took place during the 2008 *European Summer School in Logic, Language and Information*. Three datasets for concepts categorization were constructed regarding three tasks. `ESSLLI-2a` aims at grouping 44 nouns into semantic categories (4 animate, 2 inanimate). `ESSLLI-2b` focuses on categorizing 40 nouns in three concreteness levels: low, moderate, high. `ESSLLI-2c` evaluates the clustering of 45 verbs into 9 categories.

[ESS08] ESSLLI, *Shared Tasks from the ESSLLI 2008 Workshop*

4.2 WORD EMBEDDING METHODS

4.2.1 *Methods*

Several word embedding methods have been developed throughout the years to capture word meaning into vectors. From eigendecomposition and other factorization methods to the famous `Word2vec`, early methods were static. They provided a single representation for all occurrences of a word. This presented problems with polysemic words that may change meaning depending on context (e.g *"mint"* is a herb but also an adjective meaning good condition). Later, *Pretrained Language Models* (`PLMS`) emerged, and provide contextual representations with one vector per token (the use of a type in context). A frenzy later started around *Pre-Trained Language Models* (`PLMS`) which were then supplanted by today's *Large Language Models* (`LLM`) that have become ubiquitous. `PLMS` and `LLM` allowed tremendous improvements in many areas of NLP on tasks such as: *Named Entity Recognition*, *translation*, *text summarization* etc.

Despite this leap forward with the rise of deep neural architectures, concerns regarding their interpretability and their use of computational resources remain. In this work, we will not consider large neural architectures, as they do not fit within the scope of frugal and interpretable representation learning. However, we consider previous discrete word embedding methods that despite not being interpretable are more frugal in comparison with `PLMS` and `LLM`. This includes `Word2vec` that was introduced in Section 2.1.2 that is a key method in the history of word embeddings.

To complete our pool of embedding methods, we sought methods that could fit within the scope of interpretability. A body of work exists about interpretable word embeddings and their extraction from large corpora. To compare `SINr-NR` on the grounds of interpretability as well as performance on intrinsic evaluation, we chose `SPINE`. It is a neural embedding model that presents state-of-the-art performances on interpretability evaluations (introduced Chapter 5) and good performances on word similarity benchmarks.

- SPINE

  `SPINE` is a neural word embedding model introduced by Subramanian et al. [Sub+18]. Its objective is to project dense discrete vectors extracted with a method such as `Word2vec` or `GloVe` into a sparse and higher dimension representation space. More precisely, `SPINE`'s goal is to increase the

[Sub+18] Subramanian et al. "SPINE: SParse Interpretable Neural Embeddings"

dimension of word embedding vectors while enforcing sparsity. We have previously mentioned, while describing SINr in Section 2.3.2 that sparsity could play a central role in a context of interpretability, by providing dimensions that are activated by a few related terms. The combination of these dimensions that we hope to be topical in turns shapes the sense of words represented. We will investigate the benefits of sparsity for interpretability in Chapter 5.

In practice, SPINE has an autoencoder architecture composed of two modules, an encoder, that projects the data in a higher-dimensional representation than the one provided by the input dense model. And the decoder, that tries to reconstruct the input representation from the higher-dimensional one. The autoencoder is trained to minimize the reconstruction error in the output regarding the input representation. SPINE's loss is based on three cost functions:

(i) *Reconstruction Loss* (RL): optimizes for the good reconstruction of the input representation in output of the decoder.

(ii) *Average Sparsity Loss* (ASL): promotes the sparsity of the representations. It encourages the majority of the neurons or dimensions in the latent space to remain inactive for a given input.

(iii) *Partial Sparsity Loss* (PSL): promotes the sparsity of each representation by skewing values towards 0 and 1.

These three losses are summed at training time, $\mathcal{L}_{\text{SPINE}} = \mathcal{L}_{\text{RL}} + \mathcal{L}_{\text{ASL}} + \mathcal{L}_{\text{PSL}}$ fostering sparsity in the intermediary representation that is used as an embedding vector. We will see Section 4.4.1 that training SPINE presents some challenges.

4.2.2 *Experimental setup: models*

The baseline models we selected aim at representing a diversity of approaches to embedding words. Classical word embedding methods include Word2vec [Mik+13], SPINE [Sub+18] that will be based on embeddings extracted with Word2vec, and finally SINr-NR. As we have seen Chapter 3, SINr-NR can handle graph embedding on networks of emails or co-authorship. Our objective is to demonstrate that it can also be used with word co-occurrence networks and be a unified approach for both fields. Subsequently, we wondered whether other *a priori* graph embedding methods were suited for word embedding. To that end, we chose to experiment with HOPE that factorizes a node similarity matrix and is similar in philosophy to GloVe. Our second graph embedding model applied to text data is LouvainNE. LouvainNE shares its community detection algorithm with SINr-NR: Louvain. It is very efficient for graph embedding and relies on the hierarchy of communities. We thus wanted to investigate whether it could perform for text applications. Approaches based on a combination of random walks and Word2vec (Walklets and Deepwalk) are not relevant to word embedding, since they employ Word2vec that already works with text data. Word2vec, HOPE and LouvainNE are of dimension 300 for all corpora described in Section 4.3.2. SPINE has 1000 dimensions, but we will see that its training presents some

[Mik+13] Mikolov et al., "Efficient estimation of word representations in vector space"

challenges in Section 4.4.2. SINr-NR's model size depends on the number of communities detected, therefore it varies based on corpus, and has 4,460 on OANC, 8,454 on BNC and 5,700 communities on UkWac [Bér+24]. Our three chosen corpora are described in the next section.

[Bér+24] Béranger et al., "Filtering Communities in Word Co-Occurrence Networks to Foster the Emergence of Meaning"

4.3 EVALUATING WORD EMBEDDING QUANTITATIVELY

To build co-occurrence networks, we need text data that can ideally convey word sense for a large lexicon. Our evaluations in Section 3.2.1 are dataset-based, subsequently, we wish to preserve as much vocabulary from these datasets as possible while providing representations of high quality.

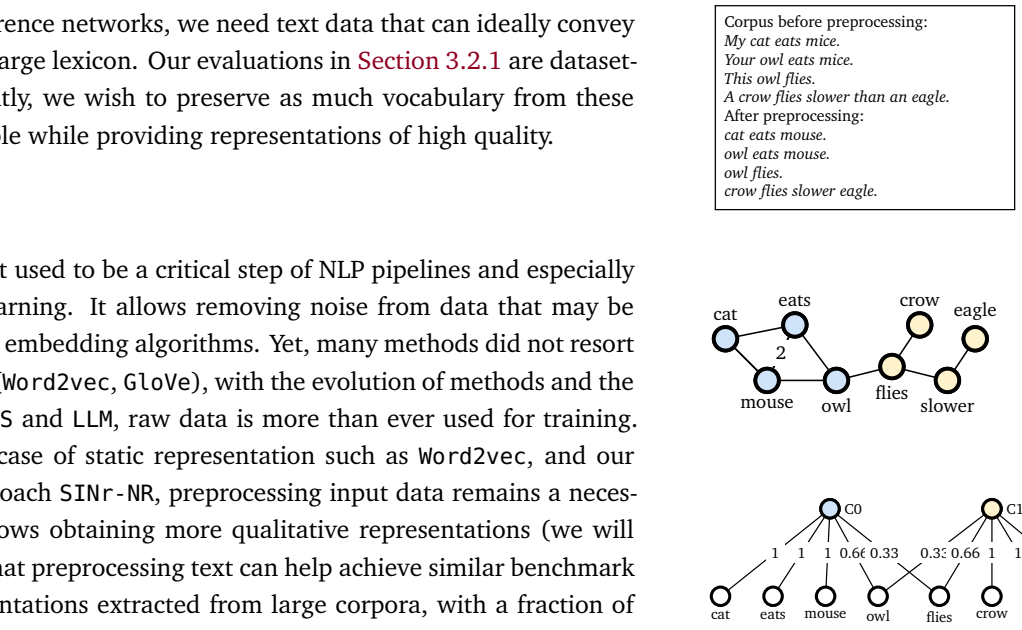
4.3.1 *Preprocessing*

Preprocessing text used to be a critical step of NLP pipelines and especially representation learning. It allows removing noise from data that may be captured by word embedding algorithms. Yet, many methods did not resort to preprocessing (Word2vec, GloVe), with the evolution of methods and the shift towards PLMS and LLM, raw data is more than ever used for training. However, in the case of static representation such as Word2vec, and our graph-based approach SINr-NR, preprocessing input data remains a necessary step that allows obtaining more qualitative representations (we will see in Table 4.4 that preprocessing text can help achieve similar benchmark results to representations extracted from large corpora, with a fraction of the data).

To construct the adjacency matrix of a weighted co-occurrence graph, in reality a word a co-occurrence matrix, we apply a sliding context window over the corpus. This sliding window retrieves co-occurring words within a distance ℓ of each other so that, given a word w, context is sampled on the left and the right of w with a maximum of 2ℓ co-occurring words. Co-occurrences are only considered within sentence boundaries. This co-occurrence matrix is not extracted on the raw text content, it is preprocessed to ease the representation learning process. Our preprocessing pipeline is common to all word embedding approaches employed in this work. The goal is to keep only the relevant co-occurrences and standardize as much as possible the lexicon of the corpora. With preprocessing, we can limit the size of the lexicon and increase the number of occurrences per type. Let us detail the various steps of the pipeline.

To limit the number of vertices in the co-occurrence graph, we apply several preprocessing steps:

- **Stop words and minimum length**: To avoid processing some words, we keep a list of exceptions, but we filter out words with fewer than three characters as a proxy for stop words (*e.g.*, "*a*", "*or*"). Stop words are words that do not typically carry sense and are often filtered out.
- **Minimum frequency**: It is customary to set a lower bound for the minimum number of occurrences of a word, in our case to 20. Indeed, rare words increase the vocabulary size without providing enough occurrences in context to build a relevant representation.



	C0	*C1*
cat	1	0
eats	1	0
mouse	1	0
owl	0.66	0.33
flies	0.33	0.66
crow	0	1
eagle	0	1
slower	0	1

FIGURE 4.2: Processing steps to obtain SINr-NR word embeddings, from top to bottom:

- Preprocessing of text to remove unwanted words.
- Construction of network $G = (V, E, \Omega)$ from co-occurrences and community detection (2 communities).
- Bipartite projection of G into graph $G' = (\top, \bot, E', \Omega')$ along the communities. Weight on the edges is based on NR, the proportion of the weighted degree of each node related to the community.
- Adjacency matrix of G', each row is a SINr-NR embedding.

- **Named entity chunking**: We chunk named entities with `SpaCy` under a single type and lowercase the vocabulary (*e.g.*, "*Emperor Julius Caesar*" becomes "*emperor_julius_caesar*"). This allows standardizing named entities and capturing more occurrences under the same type before learning a representation.

- **Lemmatization**: In order to standardize further the vocabulary, we lemmatize the words with `SpaCy`'s large english model, that is, representing inflected forms of a word under a single lemma.

- **PPMI co-occurrence filtering**: Prior to constructing a co-occurrence graph from the co-occurrence matrix $\mathcal{A}$, we apply an additional filtering step. Using the *Pointwise Mutual Information* measure (PMI), we remove co-occurrences which appear less than they would by chance as a way to limit as much as possible edges between nodes representing words whose co-occurrence is not relevant (refer back to Section 2.3.2 for more information).

These preprocessing steps allow transitioning from sentences of words to sequences of lemmas, as presented in the first subfigure of Figure 4.2. `Word2vec` will make use of this preprocessed text to learn its embeddings. For graph-specific methods, we first need to construct the co-occurrence network.

After applying these preprocessing steps and retrieving co-occurrences with the sliding context window we described previously, we obtain a word co-occurrence matrix $\mathcal{A}$ that will be used as the adjacency matrix of our weighted co-occurrence network. Let a weighted network $G = (V, E, \Omega)$. In our context, V is the set of vertices representing words in set L, the lexicon extracted from the corpora. E is the set of edges such that two vertices u and v representing two words w_u, $w_v \in L$ are connected when they appear together within the context window of size ℓ. The edge weight $\omega_{u,v} \in \Omega$ is the count of how many times w_u and w_v have been observed together. A toy co-occurrence network from the previously preprocessed text is presented in Figure 4.2.

This network can be used by graph embedding approaches to extract node (word) representations. The last two schemas in Figure 4.2 are `SINr-NR` steps. We first detect communities with `Louvain` and afterward we measure the connection of each node (word) to each community with NR to build a vector for each word. After all these steps, we obtain community-based word embeddings.

To learn relevant representations of words, we need large quantities of text data. Ideally, words in these corpora need to appear in various contexts so that their embeddings are as accurate as possible.

4.3.2 *Corpora*

Co-occurrence networks can be constructed from any collection of texts. We employed three corpora that are composite in genres, open and readily available to the public.

[Nan11] Nancy Ide, *The Open ANC (OANC)*

a. *Open American National Corpus* (`OANC`) [Nan11] is the text part of

the collection in contemporary American English, with texts from the 1990s onward. The corpus contains 11 million tokens prior to preprocessing and 20,814 types (vocabulary) for 4 million tokens after preprocessing.

b. *British National Corpus* (BNC) [Con07] is the written part of the corpora from a wide variety of sources to represent a wide cross-section of British English from the late 20th century. The raw corpus contains 100 million words. After preprocessing, the corpus is reduced to 40 million occurrences for a vocabulary of 58,687 types.

[Con07] Consortium, *British National Corpus, XML edition*

c. *Uk Web As Corpus* (UkWac) [Bar+09] is a cleaned crawled corpus from the *.uk* internet domain with over 2 billion tokens. After preprocessing, the corpus is reduced to 882 million occurrences for a vocabulary of 269,493 types.

[Bar+09] Baroni et al., "The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora"

These corpora are increasingly large, allowing to show how performances vary based on the number of tokens in a corpus. Now that we have defined the tasks, methods and data, we can benchmark word embedding methods. Our evaluation aims at showing that SINr-NR applied to text can compete on the grounds of runtime and obtains competitive results on classical intrinsic evaluation tasks. We dedicate Chapter 5 to exploring the interpretability aspects of SINr-NR and SPINE.

4.4 BENCHMARKING SINR

4.4.1 *Word embedding: from the compute standpoint*

If we consider runtime as shown in Table 4.1, SINr-NR results are convincing. Even if the approach is slower than Word2vec on BNC, it actually requires less compute (CPU time). We shall recall that the total run-time of SINr-NR is the sum of times required to extract the co-occurrence network and to train the embeddings. It is interesting to note that for 10 runs of SINr-NR (for instance to tune γ), the co-occurrence extraction is only performed once, which makes it faster than Word2vec in that particular case (3,658s for SINr-NR and 4,890s for Word2vec). When compared to SPINE (4k epochs as recommended by authors), the competing interpretable approach, SINr-NR is far more time-efficient. First, SPINE is based on dense vectors such as Word2vec that are sparsified to make them interpretable, as a consequence, we have to account for the runtime of Word2vec in SPINE.

Furthermore, the k-sparse auto-encoding of the dense vectors to make them interpretable requires some time. For OANC, the total CPU time required to run the 4k epochs of SPINE is $1,000$ times the runtime required for SINr-NR. As shown Figure 4.3, the losses seem to indicate that $4,000$ epochs are needed, but similarity results seem to indicate that $1,000$ is enough. SPINE's tuning is complex due to the loss not decreasing at the same rate *pairwise word similarity* improves. If we stop training once optimal similarity is reached, SPINE's run-time is divided by four, which is still above the runtime of SINr-NR, our interpretable approach, by a large margin.

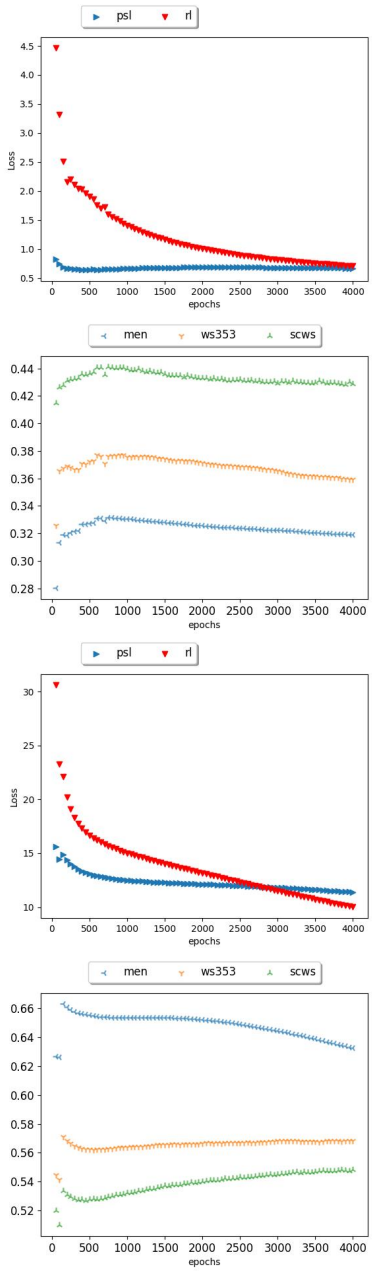


FIGURE 4.3: Training of SPINE. PSL and RL loss according to number of epochs. Spearman correlation for word similarity evaluation on MEN, WS353, and SCWS according to the number of epochs. OANC top two figures, BNC bottom two figures.

	OANC		BNC	
	CPU time	run-time	CPU time	run-time
Word2vec	175	49	1805	**489**
SPINE	50k	15k	130k	36k
Graph extraction	6	7	188	188
+ SINr-NR training	38	34	383	347
= SINr-NR total	**44**	**41**	**571**	535

TABLE 4.1: Average total CPU time and run-time in seconds over 10 runs. Run-time is computed with four cores on *Intel Xeon E5-2690 v2 3.00GHz CPU*. For SPINE, "k" indicates kilo-seconds. SINr-NR total is the sum of SINr-NR training (SINr-NR column) and of the co-occurrence matrix computing (co-occ column).

If we now consider the runtimes of competing graphs approaches on the textual data Table 4.2, results are also interesting. LouvainNE is the fastest approach, but it does not achieve good results for this similarity task in Table 4.3.

HOPE which is the best competing graph approach regarding similarity results runs ten times slower than SINr-NR. This demonstrates that SINr-NR is versatile and can work efficiently with graph and text data.

	OANC		BNC	
	CPU time	run-time	CPU time	run-time
SINr-NR	38	34	383	347
LouvainNE	**3**	**3**	**16**	**16**
HOPE	492	425	3799	3748

TABLE 4.2: Average total CPU time and run-time in seconds over 10 runs. Run-time is computed with four cores on *Intel Xeon E5-2690 v2 3.00GHz CPU*. The time required to compute the co-occurrence graph is not included, figures only report the training time.

2 *Word similarity*

We now evaluate our embedding models on the classic word similarity evaluation, 10 models are trained for each method and each corpus. *Pairwise word similarity* are averaged across these 10 models and presented Table 4.3. For SPINE, the best results regarding the $4k$ epochs of training are kept, as shown Figure 4.3. The average correlation values are rather close across models and datasets. Interpretable models (SPINE, SINr-NR) obtain results that are not far from Word2vec. The gap is even tighter on our larger corpora BNC. SINr-NR and SPINE remain very close to Word2vec. These results strongly suggest that one can build interpretable representations that still perform close to dense embedding models such as Word2vec. Regarding graph-based approaches, HOPE also performs very well on OANC with results similar to those of SPINE. However, it does not scale up with the higher number of co-occurrences of BNC, results are lower than those of SPINE and SINr-NR. LouvainNE which achieves good performances on other tasks does not seem to be fitted to deal with textual graph data.

When we scale up to our largest corpora, we can see that, although SINr-NR is behind SPINE, the difference is at most 0.08 points. What is even more interesting is comparing Word2vec's results on UkWac (850M tokens

OANC	MEN	WS353	SCWS
Word2vec	**0.43**	**0.50**	**0.46**
SPINE	0.33	0.38	0.44
SINr-NR	0.39	0.44	0.39
HOPE	0.33	0.43	0.39
LouvainNE	0.29	0.37	0.23
BNC	MEN	WS353	SCWS
Word2vec	**0.72**	**0.65**	**0.57**
SPINE	0.66	0.57	0.54
SINr-NR	0.66	0.62	0.54
HOPE	0.53	0.54	0.53
LouvainNE	0.28	0.36	0.25

TABLE 4.3: Average Spearman correlation over 10 runs for MEN, WS353 and SCWS word similarity datasets.

after preprocessing), with our processing pipeline, and Word2vec trained by Bommasani et al. [BDC20] on a corpus larger by multiple orders of magnitude (100B tokens, Table 4.4). On unprocessed corpora, the results of the similarity evaluation on WS353 seem to be positively impacted by preprocessing, since Word2vec achieves the same correlation as SINr-NR on a much smaller corpus with preprocessed text. We could thus conjecture that preprocessing is important beyond corpus size, and that with adequate preprocessing and larger corpora, Word2vec and SINr-NR could reach higher scores and come close to BERT's 0.73, with the added benefit that SINr-NR provides interpretability by design (Appendix A provides additional results that show that SINr-NR can catch up with state-of-the-art methods). In Chapter 5 we show that we can improve our performances and interpretability by increasing the sparsity of our representation and, in turn, filtering-out unwanted noise from vectors.

[BDC20] Bommasani et al. "Interpreting pretrained contextualized representations via reductions to static embeddings"

UkWac (2B)	MEN	WS353	SCWS
Word2vec	**0.75**	0.66	**0.64**
SINr-NR	0.70	**0.68**	0.56
GoogleBooks (100B)	MEN	WS353	SCWS
Word2vec [BDC20]	—	0.68	—
BERT [BDC20]	—	0.73	—

TABLE 4.4: Average Spearman correlation over 10 runs for MEN, WS353 and SCWS word similarity datasets.

4.4.3 *Concept categorization*

Categorization results on OANC, our smallest dataset show close purity results between all methods. Word2vec is leading on most datasets by a short margin. We can also see that SPINE and SINr-NR, that are interpretable, have quite similar results. Methods not necessarily dedicated to the task of word embedding perform lower. Overall, the purity scores on our smaller

OANC	SINr-NR	SPINE	Word2vec	HOPE	LouvainNE
AP	0.299	0.325	**0.353**	0.233	0.186
BLESS	0.402	0.376	**0.411**	0.276	0.225
ESSLLI-2c	**0.489**	0.444	0.469	0.431	0.378
ESSLLI-2b	0.688	0.680	**0.702**	0.615	0.635
ESSLLI-2a	0.516	0.573	**0.593**	0.457	0.475
BNC	**SINr-NR**	**SPINE**	**Word2vec**	**HOPE**	**LouvainNE**
AP	0.541	0.567	**0.589**	0.396	0.183
BLESS	0.755	0.774	**0.832**	0.455	0.357
ESSLLI-2c	0.580	0.538	**0.594**	0.489	0.329
ESSLLI-2b	0.708	0.703	0.700	**0.738**	0.675
ESSLLI-2a	0.786	0.732	**0.798**	0.630	0.543

TABLE 4.5: Concept categorization purity scores over 10 runs for BNC.

corpora are lower than on BNC which contains more occurrences. On BNC, the trend is the same as on OANC: SINr-NR, SPINE and Word2vec are very close to one another. HOPE and LouvainNE have subpar performances except for HOPE on ESSLLI-2b for word concreteness level categorization, similarly to what was already observed on the *pairwise word similarity* evaluation, they cannot compete with methods dedicated to word embedding.

In summary, the *concept categorization* task is complex but interpretable methods, although mostly behind Word2vec manage to remain close. As we stated before, they bring interpretability which is not a feature provided by most embedding models, subsequently there might be a small trade-off to find between performance on intrinsic evaluations and interpretability.

4.5 SUMMARY

The goal with word embeddings is to capture words' semantics from large corpora into a summarized representation. Much like graph embeddings capture the topology around a node, word embedding captures the context in which words appear. In this chapter, we focused on word embeddings and their evaluation. The key elements of this chapter are the following:

(i) Word embedding employs similar approaches to graph embedding: eigendecomposition, factorization with objective functions (Word2vec, GloVe), and deep neural architectures (PLMS, LLM).

(ii) Runtime and lack of interpretability are common limitations of models. Interpretability is especially critical in sensitive applications related to representation learning.

(iii) Models fostering interpretability have been introduced so that humans can audit their structure.

(iv) SINr-NR is the fastest word embedding method we benchmarked (actually, LouvainNE is faster but does not provide qualitative representations).

(v) `SINr-NR` achieves close results to `Word2vec` on two classical intrinsic evaluation tasks (*pairwise word similarity* and *word categorization*), while having a lower runtime and interpretability.

On top of low runtimes, the real added benefit of `SINr-NR` is interpretability, which is the focus of Chapter 5. We will see that `SPINE` and `SINr-NR` are interpretable, investigate methods to improve the quality and interpretability of representations, introduce a new kind of interpretability and visualize how it materializes in representations.

5

Interpreting representations

Chapter contents

Synopsis

Explainability and interpretability are growing concerns in machine learning. Bringing more explainability to decisions and interpretability to systems can help mitigate potential biases, shortcomings of models, and foster confidence in algorithmic pipelines. Specific word embedding models were developed to fulfill the needs for more interpretability in representations. Their main benefit is that they are self-sufficient for interpretability, no external information source is required to draw interpretations from a model. This chapter focuses on the interpretability of such models, and especially of `SINr-NR`. We show that `SINr-NR` competes with state-of-the-art interpretable word embedding methods. Furthermore, we extend the definition of interpretability for word embeddings that opens new opportunities to understand how word embedding models are structured. Finally, we visualize interpretable characteristics of our model, giving a glimpse at the internal structure of word embeddings.

Résumé

L'explicabilité et l'interprétabilité sont des sujets qui prennent de l'ampleur dans les domaines de l'apprentissage automatique. Rendre les décisions explicables et les systèmes interprétables peut être utile pour prévenir les biais potentiels présents dans les systèmes et augmenter la confiance dans ces derniers. Des modèles interprétables ont été développés pour l'apprentissage de plongements de mots. Leur avantage principal est de ne pas recourir à d'autres informations que celles présentes dans le modèle pour interpréter les représentations. Ce chapitre s'intéresse à l'interprétabilité de tels modèles et spécifiquement à l'interprétabilité de `SINr-NR`. Nous montrons que `SINr-NR` est au niveau des méthodes de plongements de mots interprétables à l'état de l'art. Dans un second temps, nous étendons la définition d'interprétabilité pour les plongements de mots. Cela ouvre de nouvelles perspectives pour les modèles interprétables et pour comprendre comment l'information s'y organise. Enfin, nous montrons, au travers de plusieurs visualisations, l'interprétabilité des modèles et donnons un aperçu de leur structure interne.

5.1 INTERPRETABILITY: DEFINING THE CONTOURS

5.1.1 *Interpretability vs. explainability: definitions*

Interpretability and explainability co-exist in machine learning. They allow grasping part of the complexity of models and may allow more trust in systems. With the emergence of machine learning in sensitive fields such as medicine or justice, there is a growing interest in explicable and interpretable models. Since explainability and interpretability do not give the same insight about a model, we start by defining explainability and interpretability in machine learning.

▸ EXPLAINABILITY

With the emergence of artificial intelligence (AI) and particularly neural networks, it has become increasingly hard to understand the inner mechanics of AI models. Neural networks are deemed *black box*, they have inputs and outputs for which we have no understanding of the internal workings. With explainable models, explainable artificial intelligence (XAI) has become a popular subfield of artificial intelligence. The goal of XAI is to provide processes and methods that allow humans to comprehend the results and outputs of black box machine learning algorithms. It is used to gain insight into the inner mechanics of a model and what led to a particular decision. Because AI models optimize mathematical functions, they may use unforeseen optimizations to come up with a decision. Gaining insight into what led to a particular decision is the goal of explainability in machine learning as defined by Broniatowski et al. [Bro+21]

[Bro+21] Broniatowski et al. "Psychological foundations of explainability and interpretability in artificial intelligence"

> "[...] an explanation of a model result in a description of how a model's outcome came to be"
>
> Broniatowski [Bro+21]

An ISO [ISO20] technical report defines explainability as:

[ISO20] ISO Central Secretary, *ISO/IEC TR 29119-11:2020*

> "level of understanding how the AI-based system [...] came up with a given result"
>
> ISO/IEC TR 29119-11:2 [ISO20]

Generally, a second model on top of the black box model is used to gain insight into the decision process. It is used to draw explanations from the model weights and bring insight into the decision process.

> "A second (post hoc) model is created to explain the black box model"
>
> Rudin [Rud19]

Naturally, having a second model on top of the black box model requires training and fine-tuning parameters of the second model, providing explanations. It is thus an overhead on top of training the base black box model.

Furthermore, explainability requires technical expertise to analyze the contribution of model parameters identified by the post-hoc model, and to understand the decision process.

▸ INTERPRETABILITY

Interpretability is another means of gaining insight into a model. An interpretable model should be shaped so that humans can try to understand how it is structured and relates with the representation a human has of what it models. Interpretability sometimes overlaps with explainability, as interpretations are drawn from post-hoc models. However, with interpretability, the sole requirement to make sense of the internal workings of a model should be world knowledge (in opposition with technical expertise to make sense of explanations in XAI). A broad definition of interpretability is given in the same ISO report about testing AI-based systems. The definition given is the following:

> "level of understanding how the underlying (AI) technology works"
>
> ISO/IEC TR 29119-11:2 [ISO20]

This definition emphasizes on the underlying system, when for explainability, the focus was on the process rather than the inner mechanics. That is the main difference between explainability and interpretability. Broniatowski et al. [Bro+21] provides a more precise definition of interpretability from the human standpoint.

> "Interpretation refers to a human's ability to make sense, or derive meaning, from a given stimulus so that the human can make a decision. Interpretations are simple [...] gist mental representations that contextualize a stimulus and leverage a human's background knowledge. [...] Thus an interpretable model should provide users with a description of what a stimulus, such as a datapoint or model output, means in context"
>
> Broniatowski [Bro+21]

With interpretability, the model should be structured so that using their knowledge, humans can understand how it is structured and be audited without requiring external information. The context or internal organization of the model should suffice to understand the structure or an output.

▸ ARGUMENTS FOR INTERPRETABILITY OVER EXPLAINABILITY

Explainability and interpretability have the same line of sight: gaining insight into machine learning models. However, they have different approaches that might favor one over the other. Rudin [Rud19] highlights the main hurdles pertraining to explainable and interpretable models.

The first critiques are targeted towards explainable approaches. They usually rely on a post-hoc model, that is supposed to give explanations from

[Rud19] Rudin "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead"

the output of a black box model. For example, in word representation learning, an explainable model might be used to interpret the contribution of activations in a neural network to a representation using probing [RKR20]. It is also possible to analyze an embedding matrix with eigenvector analysis [SMF18]. Regarding these explainable approaches, multiple questions arise. What happens if the explanations and interpretations made by the post-hoc model are not faithful to the underlying black box model? In such a case, an explainable model might, for part of the feature space, provide explanations that are not faithful to the original model. The second argument comes in the form of accuracy. Because explainable models are used on top of black box models, the explanations they derive are approximate and not the exact decision process performed by the model. Possibly, these explanations are wrong for parts of the feature space and lead to wrong explanations. This potential inaccuracy is detrimental to the confidence systems involved in high-stake decisions, such as explaining why an AI model detected a tumor on an X-ray scan or why a model predicted high-recidivism for an individual applying for parole. Another risk with explainable models is an overconfidence in the explanations given. Explanations might be incomplete or not detailed enough so that it is not possible to understand the decision process of the underlying model.

[RKR20] Rogers et al., "A Primer in BERTology: What We Know About How BERT Works"

[SMF18] Shin et al., "Interpreting word embeddings with eigenvector analysis"

Subsequently, interpretable approaches should be favored, where applicable, in place of explainable ones. Nevertheless, there are also obstacles to the development of interpretable models. Rudin [Rud19] identifies three. The first one is related to the commercial advantage of having a black box model. If interpretable AI models were to be popularized, companies that have capitalized on providing individual prediction could lose money. Furthermore, keeping models as black boxes would prevent them from being reverse-engineered and used for illicit applications, for example biasing the decision process of an algorithm. The second argument is not related to black box model, but rather to the complexity of constructing interpretable models. The constraint of interpretability complexifies training and optimization and impedes their democratization. Finally, black box models are supposed to uncover "hidden patterns" in data that might not be uncovered by an interpretable approach. Rudin [Rud19] counters this argument by noting that if a pattern is important enough that it is picked by a black box model, an accurate interpretable model should be able to identify it too.

[Rud19] Rudin "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead"

Whether they are explainable or interpretable, the objective of understanding model's mechanics and outcome has made it into the European Union *General Data Protection Regulation* (GDPR), safeguarding decision-making based solely on algorithms, and introducing a right to explanation.

> "The data subject should have the right not to be subject to a decision, which may include a measure, evaluating personal aspects relating to him or her which is based solely on automated processing and which produces legal effects concerning him or her or similarly significantly affects him or her, such as automatic refusal of an online credit application or e-recruiting prac-

> tices without any human intervention. [...] such processing should be subject to suitable safeguards, which should include specific information to the data subject and the right to obtain human intervention, to express his or her point of view, to *obtain an explanation of the decision reached* after such assessment and to challenge the decision."
>
> GDPR, Recital 71, Profiling [EC16]

The French *Loi pour une république numérique* goes beyond the GDPR by introducing a provision to explain decisions made by public bodies about individuals when they are taken because of an algorithmic treatment. The principal characteristics of that treatment should be communicated upon request: contribution of algorithmic processing to the decision, data processed and source, treatment parameters and their weights where applicable, and the operations carried out by the algorithm.

This work focuses on interpretability, as explainability with post-hoc models does not satisfy our objective of frugality. Post-hoc interpretability requires training another model to provide explanations, which involves training and parameter tuning. This presents a significant overhead on top of the base model. Therefore, "*by design*" interpretable models should be favored, they guarantee more accuracy [Rud19]. We now describe interpretability in the specific context of word embeddings. We will focus mainly on the criteria for interpretable word embeddings that are essential to our word embeddings obtained with `SINr-NR`.

5.1.2 *Interpretability of word embeddings and criteria*

If we recall our definition of interpretability Section 5.1.1 and transpose it to word embedding, interpretability is the possibility for humans to make sense of the structure of an embedding space. In general, interpretability materializes in the form of semantically coherent components. Let us start with an example of interpretable models in Table 5.1 that will help to support the criteria we present in the second part of this section. We present three words for which we have extracted from the embedding matrix the three dimensions for which the value in the vector was the highest. Afterward, we rank the words according to their values on these components and select the top four (all models have been trained on `BNC`). In the case of `Word2vec`, a dense model, we see that the strongest dimension (scalar, tablespoon, etc.) appears for each of the words selected, and words with the highest values on each dimension are not semantically consistent ("*scalar*" and "*tablespoon*", "*geranium*" and "*curiosities*", "*herbicides*" and "*menstrual*"). In the case of `SPINE` and `SINr-NR`, words on the top dimensions make sense with one another. Naturally, "*insulin*" and *"oxygen"* are largely represented by medical and physiological terms. The third-strongest component for *"oxygen"* on `SPINE` is related to forms of atoms. More interestingly, "*mint*" has more variety in terms of topics, especially for `SINr-NR`. Indeed, the first dimension is related to herbs and cooking. The second di-

	Word2vec	SPINE	SINr-NR
insulin	scalar, tablespoon, vesicular, dystrophy antiserum, falsifiable, experimenter, internat PBS, NC, arginine, IFN	glutathione, pancreas, gastroduodenal, vitamin immunologically, hyperplasia, transgene, nociceptive insulin, sulphasalazine, interferon, cholangitis	hypertriglyceridaemia, mellitus, porcine, insulin aldosterone, aminotransferase, creatinine, glycated ulcerative, sulphasalazine, colitis, sera
mint	scalar, tablespoon, vesicular, dystrophy cube, geranium, Berowne, curiosities polyunsaturated, misfire, margarine, methile	spoonfuls, parsnips, kebabs, preheat onion, basil, yogurt, coriander dial, screams, vibration, spadefoot	tbsp, oregano, diced, dijon Gibson, gigged, charvel, Ibanez minted, minting, hoards, coinages
oxygen	scalar, tablespoon, vesicular, dystrophy herbicides, menstrual, deprave, angiotensin pou, tenascin, cytoplasm, platelet	glutathione, pancreas, gastroduodenal, vitamin lipid, crypt, tris, calcium monoxide, oxides, sulphuric, nitrogen	monoxide, dioxide, nitrous, oxides supplemental, hypoxaemic, electrocardiographic, gastroscopy diastolic, systolic, transfusion, transfusions

TABLE 5.1: Words with the highest values on the top three dimensions of "*insulin*", "*mint*" and "*oxygen*" in Word2vec, SPINE and SINr-NR sparsified to 100 active dimensions per vector according to the protocol described Section 5.5.1.

mension is more surprising, since it seems to be about guitar brands. However, we know that BNC contains classified ads about the sale of guitars and "*mint*" is an adjective related to the condition of an object that may be used in classified ads. The third-strongest dimension is related to the monetary aspect of "*mint*" and terms related to minting money. This example shows the polysemy of the term that can be captured from the co-occurrences in the corpus.

As we see in Table 5.1, interpretable dimensions can be obtained with specialized word embedding algorithms such as SPINE and SINr-NR. Murphy et al. [MTM12] published a seminal paper about interpretable word embeddings. This article lays the first criteria to foster interpretability in word embeddings and obtain intepretable dimensions. According to Murphy et al. [MTM12], criteria necessary to have an interpretable word embedding model are: (i) sparsity, (ii) non-negativity, and (iii) performance. Let us now detail the contributions of these three criteria to interpretability.

[MTM12] Murphy et al. "Learning effective and interpretable semantic models using non-negative sparse embedding"

(i) The first criterion for an interpretable model is sparsity. If we recall dense models like Word2vec or SVD, their vectors have activations for all dimensions. Yet, if a dimension is activated (has a value different from 0) for all words, then it likely is not a thematic group as we have observed in Table 5.1. For dimensions to be thematic, and approximately correspond to a trait (medical term, herb), not all words should use this dimension. Thus, the embedding matrix should be sparse and only a restricted set of words should activate each dimension. And in return, only a subset of dimensions should be used to represent a word. Depending on whether a word is generic or specific, it may activate more or less dimensions. Furthermore, most dense models only have 300 to 500 dimensions. It is sufficient if all these dimensions are used for all words. However, the sheer variety of a language's lexicon requires far more dimensions to express the diversity of word senses and thematics. Subsequently, to foster interpretability, an embedding space needs to be sparse and have many dimensions (at least 1,000 for most models).

(ii) The second criteria to foster interpretability is non-negativity. Humans rarely represent items by what they are not. Indeed, an interpretable model should only store positive facts and not for example that a *"hammer"* is not a *"sock"*[1]. Subsequently, representations in an interpretable embedding space should all be positive.

[1]"It would also be uneconomical for people to store all negative properties of a concept, such as the fact that dogs do not have wheels, or that airplanes are not used for communication." Murphy [MTM12]

(iii) The last property that stems from Murphy et al. [MTM12] is quality of the representation. For a representation to be relevant, it should model accurately the representation of words for humans. To that end, the good performances of an interpretable embedding model on tasks such as *word similarity* and *concept categorization* are indicative of a model of good quality.

These criteria are the basis of most interpretable methods. We will later see that our work allowed to extend the list of criteria for interpretable word embedding models and go beyond the interpretability of dimensions. For now, we will focus on interpretable models and especially Murphy et al. [MTM12]'s `NNSE` approach.

5.2 INTERPRETABLE MODELS

There are a few methods aimed at providing sparse interpretable representations. They all share the characteristics described by Murphy et al. [MTM12]: sparsity, many dimensions and good performances on benchmarks. Some models like Subramanian et al. [Sub+18]'s `SPINE` and [Far+15]' `SPOWV` post-process dense models trained with `Word2vec` to project them in higher dimensions and in a sparse latent space. More recently, Piaggesi et al. [Pia+23] introduced `DINE` that builds interpretable node embeddings using an orthogonality loss that prevents dimensions from being redundant so that each dimension represents distinct concepts.

[Sub+18] Subramanian et al. "SPINE: SParse Interpretable Neural Embeddings"

[Far+15] Faruqui et al., "Sparse Overcomplete Word Vector Representations"

[Pia+23] Piaggesi et al. *DINE*

Other methods like `SINr-NR` and `NNSE` that we describe hereafter make use of the co-occurrence matrix without requiring a dense embedding space in input. In the case of `SINr-NR`, on top of being interpretable, the model is explainable as we have a direct link to a physical structure: communities. These communities allow tracing back the path from the clustering of words in communities to the vector representation.

The first notable interpretable word embedding model was introduced by Murphy et al. [MTM12] in his seminal paper about interpretable models. It introduces Non-Negative Sparse Embedding (`NNSE`) that caters to the interpretability criteria previously described (*i.e.*, sparsity, high-dimension, and positivity), we will present it in the next paragraph.

▸ NNSE

Murphy et al. [MTM12] introduced `NNSE` as a matrix factorization algorithm that had for main objective interpretability. According to the definition given in the same paper, interpretability of the dimensions could be achieved by enforcing positivity and sparsity. When it comes to positivity, `NNSE` enforces a regularization on U so that it only contains positive values. For sparsity, `NNSE` introduces a sparsity constraint that skews values toward 0 or 1. Given an input co-occurrence matrix X, `NNSE` tries to factorize them into the product of two matrices: U the sparse embeddings and V a new representation base. This is performed under the constraints that rows of V are upper-bounded by 1 and the output embedding matrix U is sparse. Our `SINr-MF` implementation (Section 2.3.4) is inspired by the `NNSE` model, except that we set the U matrix to be the community membership matrix. Put

together, the NNSE model has the following reconstruction objective under the previous constraints:

$$\underset{U,V}{\operatorname{argmin}}(||X - U^{\intercal} \cdot V|| + ||U||) \tag{5.1}$$

Applying NNSE to a corpus of 40,000, the authors were able to show the semantic coherence of some dimensions. They also performed a human evaluation of their model to assess the semantic coherence of dimensions in the model. Already used to evaluate the coherence of topics in topic models [Cha+09], the *word intrusion* evaluation was used to qualitatively quantify the extent of NNSE's intepretability.

[Cha+09] Chang et al., "Reading Tea Leaves"

5.3 INTERPRETING VECTOR-SPACE DIMENSIONS

5.3.1 *Word intrusion*

The question of interpretability is intertwined with human perception of dimensions' coherence. Subsequently, an evaluation task specifically designed to evaluate dimension interpretability first appeared in Chang et al. [Cha+09] to evaluate the coherence of words describing topics in *topic models*. The *word intrusion* evaluation task aims at assessing the extent of a models' dimension interpretability and has become the *de facto* evaluation of interpretability [MTM12; Far+15; Sub+18; Pro+22].

The task is based on a simple principle: if a vector space is well structured, words that are semantically close should lie close together. This is the gist of the distributional hypothesis. Now, for words to be close in a space, chances are that their representation relies on common dimensions. That is where the *word intrusion* becomes useful. If we select a dimension of the vector space and rank the words according to their value on this dimension, according to our precedent hypothesis, words with the strongest values should be semantically related. Now, how can we be assured that the words are related? We use an *"intruder"*, a word selected at random among those with the lowest values on our dimension of interest, but that is still strong on another dimension (to avoid selecting too specific or rare of a word). If a native speaker of a language can find the intruder among this set of words, then the top scoring word of the dimension must possess some semantic consistency. This semantic consistency for dimensions corresponds to interpretability for word embeddings. In our work, we use this task to compare the interpretability of SPINE and SINr-NR (NNSE had lower results than SPINE and W2V was proved to not be interpretable in [Sub+18]).

EXPERIMENTAL SETUP. We evaluated two models with such *word intrusion* protocol. The experiment is, as far as we know, the first of its kind on a large French corpus. Our models (SPINE, SINr-NR) were trained on a news corpus in French, it contains articles from the news outlet *Le Monde* (1987-2006), *AFP* (1994-2006) and news articles crawled on the web (2007). The text is purposely lemmatized, named entities are chunked under a single type and stop words are removed along with words occurring less than 10 times. Most named entities were removed to construct intrusion tasks. The

corpus spans multiple decades, and by removing named entities, we want to avoid relying too much on annotators' general knowledge of an era but instead on semantics. The preprocessed corpus contains 330M tokens and 323K words of vocabulary. We train a `SPINE` model which has 1,000 dimensions and `SINr-NR` with 4,708 dimensions.

In our *word intrusion* protocol, each task is extracted as follows: first, we sample a dimension. Then, we select the top 3 words having the highest values on the coordinate corresponding to this sampled dimension. We also sample an intruder which is part of the lower 30% of values in the coordinate and in the top 10% of another coordinate. In total, 200 dimensions were sampled for `SPINE` and `SINr-NR`. For each word in the intrusion test, three possibilities are presented: (−, ±, +). When annotators can easily detect the intruder, they should select +. When annotators hesitate between two words, they should select ±. If the annotators find all the words consistent (everything seems coherent) they should select −. This allows to finely analyze the interpretability of dimensions (hesitations, all coherent words or no coherence). The intrusion tasks were served through a web interface on a *Label-Studio* [Tka+20] server. Table 5.2 presents a selection of intrusion tasks, and Figure 5.1 presents an example of an intrusion task as presented in the interface.

[Tka+20] Tkachenko et al., *Label Studio: Data labeling software*

Model	Top Words			Intruder
`SPINE`	suffrage	urne *(ballot box)*	législative *(legislative)*	colmatage *(sealing)*
	tramway	ferroviaire *(rail)*	rail	orientation
`SINr-NR`	réseau *(network)*	chaîne *(channel)*	groupe *(group)*	déclencher *(trigger)*
	Intel	microprocesseur *(microprocessor)*	processeur *(processor)*	garder *(to keep)*

TABLE 5.2: Examples of tasks extracted for each model.

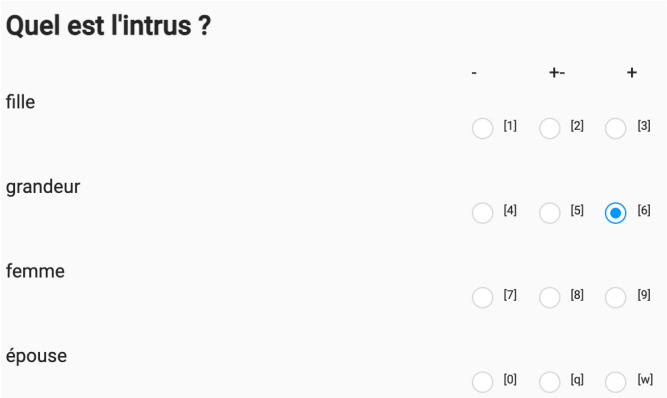


FIGURE 5.1: Example of a word intrusion task annotated. Among words: *"daughter"*, *"size"*, *"woman"*, *"wife"*. The intruder is *"height"*.

The pool of annotators was composed of 19 master students in NLP. The

participants had prior knowledge of distributional models and were literate in French, with at least 4 months in France. They evaluated 66 tasks in random order. Each task was solved by three or four participants.

	SPINE	SINr-NR
IntruderOK	**36%**	35%
+ HesitateOK	56%	**60%**
+ Consistent	57%	**62%**

TABLE 5.3: Positive and cumulative results of the intrusion detection task.

RESULTS. Table 5.3 presents the percentages of tasks for which the intruder was correctly detected (IntruderOK), the HesitateOK category corresponds to a hesitation between two words in which the intruder is one of them. The Consistent category corresponds to tasks for which annotators found consistency in terms of sense across all the words in the task. Percentages are cumulative. What we can see is that SPINE and SINr-NR are shoulder to shoulder. Percentages remain low, which indicates that the task is hard oe the embeddings are not interpretable.

[Sub+18] Subramanian et al., "SPINE: SParse Interpretable Neural Embeddings"

In their paper, *Subramanian et al.* [Sub+18] operated a similar experiment, but on a reduced vocabulary size of 15k words, which is much lower than our 323k words. In such a case, Word2vec obtains a score of 26% that should be compared to the 62% of SINr-NR. We can already see that by considering IntruderOK, SPINE and SINr-NR are ahead of Word2vec on a smaller English vocabulary. Furthermore, when we consider the IntruderOK + HesitateOK + Consistent, we can see that SINr-NR performs better and has more Consistent cases. It might be the consequence of the larger number of dimensions (4,708) that leads to redundancy in the dimensions.

Table 5.4 further analyzes the *word intrusion* evaluation results. We can see that SINr-NR has fewer instances where the annotators were quite sure about an intruder but failed to predict the right one (IntruderKO). On the other hand, SPINE manages to have fewer instances where subjects hesitated between two words and none of them was the intruder (HesitateKO). Lastly, there seems to be fewer instances for SINr-NR where subjects were unable to discern a semantic coherence among the words they were presented with (No consistency).

	SPINE	SINr-NR
IntruderKO	14%	**12%**
HesitateKO	**10%**	11%
No consistency	19%	**15%**

TABLE 5.4: Negative results of the intrusion detection task.

Regarding inter-annotator agreement, SPINE has the highest agreement, in 58% of cases, at least two annotators agree on their decision. SINr-NR is just behind with 55%. When we consider the three annotators, the percentages of agreement drop significantly, SPINE is leading with annotators

SPINE	SINr-NR
58%, 21%	55%, 13%

TABLE 5.5: Inter-annotator agreements across all models presented and overall for the *word intrusion* evaluation. For each model, the first value is the percentage of tasks where at least two evaluators annotated similarly. The second value is the percentage of tasks where the three evaluators annotated similarly.

agreeing 21% of the time. SINr-NR has a lower three annotators agreement with only 13%. We also computed *Fleiss' kappa*: SPINE has a 0.26 κ and SINr-NR a 0.21 κ. These agreements fall in the *fair agreement* category. The low κ scores further confirm the difficulty of the *word Intrusion* detection task. Furthermore, we voluntarily left more choices than the original evaluation task, thus impeding the potential for agreement.

To conclude on the *word intrusion* detection, we have shown that there is potential for interpretability of word embeddings in SPINE and SINr-NR even though it is a complex feature to evaluate. A more systematic evaluation of models' dimensions would be useful to have more quantitative and a more controlled benchmarking environment. Lau et al. [LNB14] and Sun et al. [Sun+16] have made propositions in this direction that now need to be systematized to benchmark interpretable models (see Chapter 6).

[LNB14] Lau et al. "Machine Reading Tea Leaves"

[Sun+16] Sun et al. "Sparse word embeddings using l1 regularized online learning"

5.4 INTERPRETABILITY BENEFITS FROM STABLE REPRESENTATIONS

Another aspect to consider when it comes to providing interpretable representations is whether representations returned are stable across runs. Instability impedes interpretability in the sense that we are not guaranteed to get the same model across runs. If the embedding space varies by a large margin across runs, interpretation can vary too. In such a case, it is difficult to trust interpretations drawn from an unstable model, as we are not sure that they hold true for all instances. They may be artifacts of training. Stable representations are preferable for applications requiring models to be audited, in the context of digital humanities [Gef+17], and when using diachronic alignments [HLJ16; Gar+18].

Pierrejean [Pie20] investigated the variance of neural methods to derive word embeddings. Word2vec is notoriously unstable across runs and word neighborhoods may be distorted with consequences on *pairwise word similarity* evaluations. To measure variability in models, we first evaluate the stability of SINr-NR's Louvain community detection, since it is the only random process in the algorithm. In a second evaluation, we take a look at the variation of neighbors between models. Finally, we study the stability of SINr-NR and SPINE on the *pairwise word similarity* evaluation.

[Gef+17] Gefen et al., "Vector based measure of semantic shifts across different cultural corpora as a proxy to comparative history of ideas"

[HLJ16] Hamilton et al., "Cultural shift or linguistic drift? comparing two computational measures of semantic change"

[Gar+18] Garg et al., "Word embeddings quantify 100 years of gender and ethnic stereotypes"

[Pie20] Pierrejean, "Qualitative Evaluation of Word Embeddings: Investigating the Instability in Neural-Based Models"

▸ COMMUNITY STRUCTURE STABILITY.

Inconsistency in SINr-NR vectors may stem from Louvain's random iteration on vertices. As a result, communities may change between instances of SINr-NR. With a change in community structure comes a change in the representation of items. To measure the extent of variation in community structure, we compare 10 community structures that allowed to extract SINr-NR vectors. We evaluate the pairwise *Normalized Mutual Information* (NMI)

and present the averaged NMI for all pairs of community structures in Table 5.6.

	OANC	BNC
NMI	0.967	0.959

TABLE 5.6: Average NMI comparing 10 community structures detected with `Louvain` on `OANC` and `BNC` co-occurrence networks introduced Section 4.3.2.

NMI values are high, meaning that despite randomness in `Louvain`'s order of iteration over the vertices, community detection leads to similar partitions of the vertices. Preprocessing described in Figure 4.2 using PMI filtering may explain this. Similar partitions should lead to little variation in embedding space geometry. More precisely, words neighborhoods in `SINr-NR` should not vary too much for two models extracted from the same network. This is the subject of our next experiment on word nearest neighbors variation.

▶ WORD NEIGHBORHOOD VARIATION.
Variation in neighbors between models can be detrimental to interpretability, as hypotheses drawn from one model do not necessarily hold true for another model trained with the same algorithm and same data. We know that `SINr-NR`'s communities may vary by a small extent between instances of two models. Variation in neighborhoods was demonstrated for other word embedding methods [Pie20]. To evaluate this variation, we do a pairwise comparison of word neighbors for 10 models. The *Nearest Neighbor Variation* [Pie20] described in Equation (5.2) measures the proportion of varying nearest neighbors between two models M_1 and M_2 for a number N of nearest neighbors nn according to the cosine distance. For a word w, the *Nearest Neighbor Variation* ($varnn$) score is:

[Pie20] Pierrejean, "Qualitative Evaluation of Word Embeddings: Investigating the Instability in Neural-Based Models"

$$varnn^N_{M_1,M_2}(w) = 1 - \frac{|nn^N_{M_1} \cap nn^N_{M_2}|}{N} \tag{5.2}$$

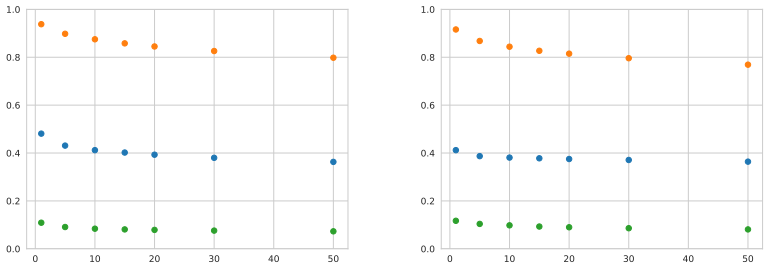


FIGURE 5.2: Neighborhood stability (average $varnn$) according to the number of nearest neighbors in cosine similarity for `OANC` (left) and `BNC` (right). Stability of: `SPINE` (orange) topmost values, `Word2vec` (blue) middle values and `SINr-NR` (green) bottom most values for a pairwise comparison of 10 models.

In Figure 5.2, we can see the variation in stability $varnn$ according to the distance at which we are retrieving nearest neighbors, *i.e.*, N in Equation (5.2). Foremost, `SINr-NR`'s neighbors variation between models is weak both on `OANC` and `BNC`. `Word2vec` is right in between `SINr-NR` and

SPINE in terms of variation with values between 0.38 and 0.50. SPINE's word embedding neighbors vary a lot more, with variation proportion being mostly over 0.8. The instability in SPINE could be expected, SPINE is derived from a dense Word2vec space, which varies. Nearest neighbors remain similar for SINr-NR even at a distance of 50. On the other hand, SPINE and Word2vec's nearest neighbor strongest variation seems to be located within the first 20 nearest neighbors.

▸ STABILITY ON WORD SIMILARITY EVALUATION

Following our investigations of the stability of communities and neighbors in interpretable models, we also perform an experiment where we measure the variation of the results on the *word similarity* evaluation. To that end, we train 10 models for each method and evaluate them on our three pairwise *word similarity* datasets: MEN, WS353, and SCWS.

As reported in Table 5.7, the three models achieve correlation scores in close ranges, with all models showing some degree of variability, their standard deviation being non-zero across ten runs. While Word2vec and SINr-NR seem more stable than SPINE, the overall observed variability on the small samples of the vocabulary present in the similarity datasets hinders reproducibility and is a flaw to the three model's interpretability.

BNC	MEN		WS353		SCWS	
	$\overline{Spearman}$	σ	$\overline{Spearman}$	σ	$\overline{Spearman}$	σ
Word2vec	0,72	0,002	0,65	0,005	0,57	0,002
SPINE	0,65	0,006	0,57	0,01	0,60	0,004
SINr-NR	0,66	0,0006	0,62	0,002	0,54	0,001
OANC	**MEN**		**WS353**		**SCWS**	
	$\overline{Spearman}$	σ	$\overline{Spearman}$	σ	$\overline{Spearman}$	σ
Word2vec	0,43	0,002	0,50	0,005	0,46	0,003
SPINE	0,36	0,009	0,43	0,01	0,39	0,01
SINr-NR	0,39	0,0008	0,44	0,002	0,39	0,002

TABLE 5.7: Stability results for the word similarity evaluation on BNC (top), and OANC (bottom). Average Pearson correlation coefficient and standard deviation σ over 10 runs.

5.5 TOWARDS VECTOR-LEVEL INTERPRETABILITY: THE CONTRIBUTION OF SPARSITY

We want to extend the set of criteria described by Murphy et al. [MTM12] to open new ways of interpreting word embedding content. In our review of the literature, we noticed that dimension-level interpretability has remained the main focus. However, the interpretability of word embeddings should not stop at the dimension level. If we recall Murphy's thoughts on describing words with features, we might wish to understand what features contribute to a representation. With a low enough number of active dimensions, humans could assess whether the combination of dimensions makes sense. When dimension-level interpretability has the objective of studying the semantic coherence of words in dimensions, vector-level interpretability aims at assessing the relevance of the dimensions representing a word.

[MTM12] Murphy et al. "Learning effective and interpretable semantic models using non-negative sparse embedding"

Vector-level interpretability is possible only if the set of active dimensions to represent a word is small enough. Murphy et al. [MTM12] had already mentioned the idea that in order for word embeddings to be *"cognitively plausible"*, or able to emulate the representation of words in the human mind, only a small set of features should be activated:

[MTM12] Murphy et al. "Learning effective and interpretable semantic models using non-negative sparse embedding"

> "So, for cognitive plausibility, we claim that a feature set should have three characteristics: it should only store positive facts; it should have a wide range of feature types, to cover all semantic domains in the typical mental lexicon; and **only a small number of these should be active to describe each word/concept**"
>
> Murphy [MTM12]

The number of dimensions a human will be able to work with in working memory is bounded by two different kinds of psycholinguistic experiments. *Semantic features production* [Gar+01; McR+05] and *semantic features retention* [Mil56; PP59] experiments come to the conclusion that the bound is at around ten. However, most models have far more than ten dimensions activated per word representation. We consider that reaching ten dimensions activated per vector is a desirable horizon for vector-level interpretability. Therefore, we extend the set of criteria for interpretability by adding increased sparsity of each vector to reduce as much as possible the number of activated dimensions for the representation of a word.

[Gar+01] Garrard et al., "Prototypicality, distinctiveness, and intercorrelation: Analyses of the semantic attributes of living and nonliving concepts"

[McR+05] McRae et al., "Semantic feature production norms for a large set of living and nonliving things"

[Mil56] Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information"

[PP59] Peterson and Peterson, "Short-Term Retention of Individual Verbal Items"

5.5.1 *Experimental framework.*

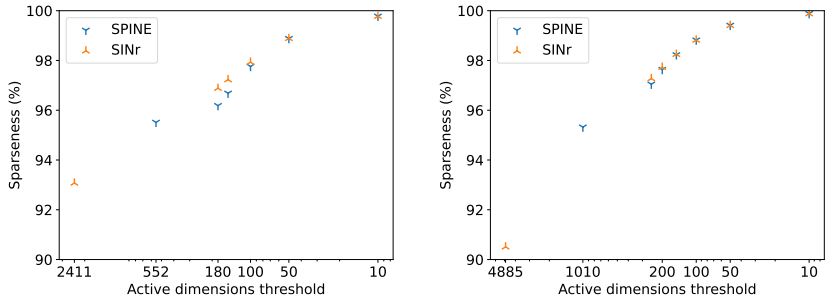
We choose an experimental framework allowing to evaluate the feasibility of vector-level interpretability. We first consider a performance-sparsity compromise. Our hypothesis is that sparse vectors are both more interpretable and psycholinguistically plausible. To control sparseness, we introduce our sparsification method: from each embedding model, we keep only the k top strongest dimensions by value in each vector, k being in range $250 - 10$. Components not in the top k for the vector are set to zero. Figure 5.3 presents the sparseness of `SPINE` and `SINr-NR` regarding the active dimension threshold k. In the case of `W2V`, we keep the top k dimensions out of the absolute values from the vectors.

In our second setup, we study the impact of switching to binary vectors and thus the contribution of real values to representation. The binarization step is straightforward, we simply replace all non-zero values in each sparsified and unsparsified model by 1 as in [Far+15].

[Far+15] Faruqui et al., "Sparse Overcomplete Word Vector Representations"

To evaluate the quality of the representations after sparsification and binarization, we use the word similarity evaluation with the three datasets of Section 4.1: `MEN`, `WS353`, and `SCWS`.

5.5.2 *Results*



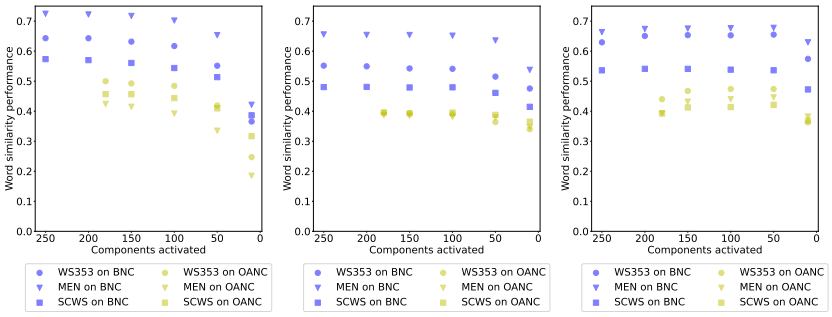
(a) SPINE (b) SINr-NR

FIGURE 5.3: Sparseness of SPINE and SINr-NR according to the maximum number of activated dimensions per vector on OANC (top) and BNC (bottom). First data point of each model is sparseness before sparsification.

SPARSIFICATION

Results presented Figure 5.4 show the Spearman correlation scores on the similarity evaluation according to the number of components activated. The similarity scores are given according to the maximum number of top values kept in each vector, in line with our sparsification procedure. We can see that the three models achieve comparable results to those reported in Table 5.7 up until 50 dimensions. More surprisingly, sparsifying SINr-NR embeddings seem to improve performances. Sparsification may filter out noise from the base SINr-NR model. Subsequently, there is not necessarily a trade-off between sparseness and efficiency. Furthermore, the fact that results remain satisfactory on our Word2vec control model despite the sparsification is a behavior we were not expecting. It seems that not all dimensions are essential in Word2vec to maintain good results on the word similarity evaluation.

At 10 dimensions, our cognitive plausibility objective, we observe an overall drop in performances and especially for Word2vec. Yet, it appears that a lot of semantic information is contained in these 10 dimensions allowing to solve, at least partially, the *pairwise word similarity* task. The low number of active dimensions renders these models compatible with theoretical models leveraging semantic features, thus paving the way for new empirical opportunities to interpret word embeddings.



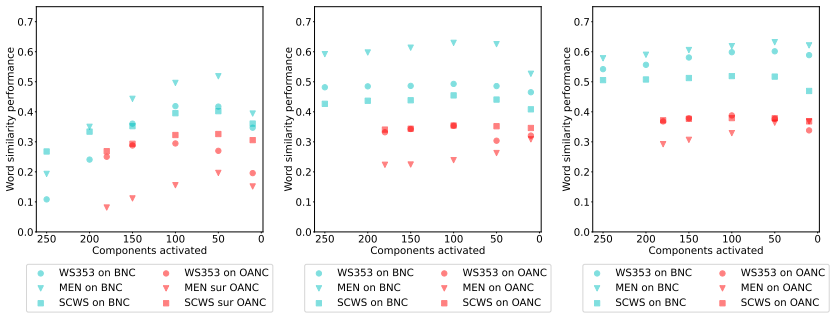
(a) W2V (b) SPINE (c) SINr-NR

FIGURE 5.4: Word similarity performance (Spearman correlation) against maximum number of activated dimensions per vector for Word2vec (left), SPINE (middle) and SINr-NR (right). Performances on OANC are reported in yellow, and performances on BNC in blue.

BINARIZATION

Results in Figure 5.4 are presented in the same way sparsity results were, except that all models are binarized. Overall, we observe drops in performance across all models, but to varying extents. While SPINE and SINr-NR lose some semantic information compared to the sparsified weighted models, they tend to retain performances of the same magnitude. This is especially true for models trained on BNC, considering that the models trained on OANC shows bigger drops in word similarity performance. On the other hand, overall, Word2vec performances crumble with binarized vectors. This result is to be expected since Word2vec is a dense model and most vectors will be filled with ones.

We can observe a common pattern across all models, where performance of binarized embeddings increases with sparsification until 100 or 50 activated dimensions. Binarizing while maintaining many active dimensions flattens the hierarchy between components with strong values and others with low activations, and thus otherwise weak activations may gain weight in the vector as a result of binarization. In this case, the sparsification may remove noise from representations, by restoring a hierarchy between the few strong dimensions, activated with a 1 value, and the others set to 0. This denoising behavior resulting from sparsification seems common to binarized models, and weighted SINr-NR.



(a) W2V (b) SPINE (c) SINr-NR

FIGURE 5.5: Word similarity performance (Pearson correlation) on binary models against maximum number of activated dimensions per vector for Word2vec (left), SPINE (middle) and SINr-NR (right). Performances on OANC are reported in magenta, and performances on BNC in cyan.

These results on the word similarity seem to indicate that there is not necessarily a trade-off between interpretability and performance. Increasing sparseness and stability over multiple training instances can improve results. Performances might be deemed acceptable at sparseness thresholds close to our objective of 10 features. Binarization can be performed to reduce model size, but real-valued vectors should be preferred as they allow achieving more accurate representations (according to *pairwise word similarity)*.

We will now focus on examples of interpretations and visualizations of dimensions in our interpretable models. Visualizing interpretable vectors is hard without a set goal in mind, as we have seen, models usually have hundreds to thousands of components. We will show some interesting observations of the organization of dimensions.

5.6 VISUALIZING INTERPRETABLE VECTORS

We showed with our *word intrusion* evaluation that dimensions of SINr-NR and SPINE are interpretable. We recall that Table 5.1 presents an assortment of dimensions from our models that show semantic relatedness, like medical terms for "*insulin*" and "*oxygen*", cooking terms for "mint". Visualizing dimension content, we can grasp some of the complexity that comes with combining dimensions about diverse topics to build sense. This corresponds to the interpretability of dimensions and was investigated in numerous articles [MTM12; Far+15; Sub+18; Pro+22].

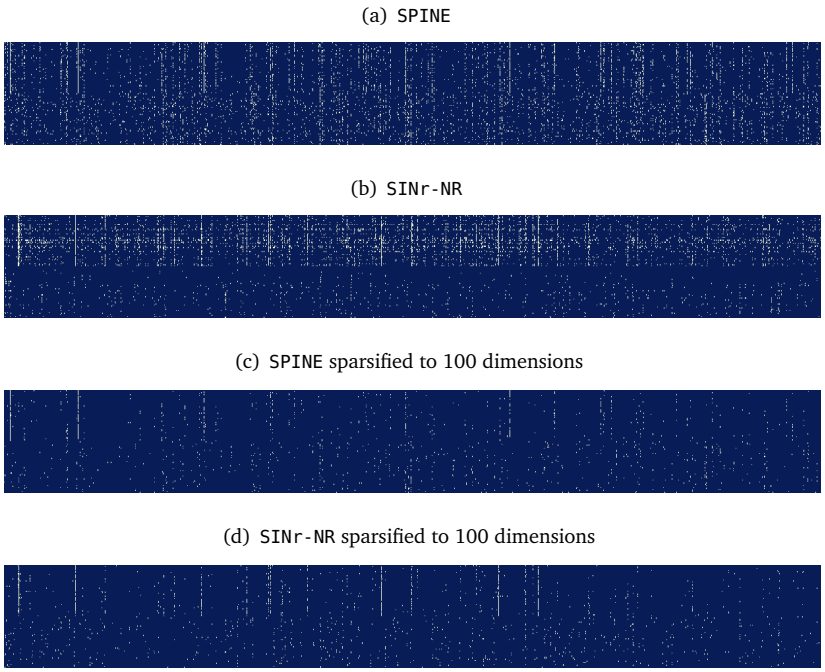


FIGURE 5.6: Shared dimensions across 50 most and least similar words to "*mint*" in SPINE and SINr-NR. The models are trained on BNC both without sparsification, and with a threshold set to 100 dimensions. The top half of each figure represents the most similar words and the bottom half the least similar words.

Vector-level interpretability, allowed by increased sparseness without sacrificing performance, is a new feature. Before visualizing labeled dimensions for words, let us first take a step back and consider the entirety of vectors. We want to visualize patterns that could occur between similar words, from a distance. We take the word "*mint*" from Table 5.1 and plot the activated dimensions of its 50 most and least similar words according to the cosine similarity. Intuitively, similar words to "*mint*" will likely share dimensions related to herbs, cooking, maybe cocktails. Subsequently, if we plot the activation of dimensions, we should see lines appearing, meaning that words that are similar use common dimensions. It is the case for SPINE Figure 5.6(a) although dimensions seemed to be shared even between most dissimilar words. For SINr-NR, we see the same lines appearing, but it is more diffuse for the most dissimilar words. What is even more interesting is the effect of sparsification to 100 activated dimensions per vector. We can see in this case that even with sparsification, words that are the most simi-

lar to "*mint*" still share dimensions. It is more pronounced for `SINr-NR`. We can also see a net reduction of the hue across representations and a potential noise reduction that positively impacts the performances of sparsified models on similarity evaluations.

In Figure 5.7, we present a version of visualizations of Figure 5.6 on which we have zoomed in on the two strongest dimensions for the word "*mint*". We can see that they are shared by most neighbors of the word, while least similar words do not share this dimension.

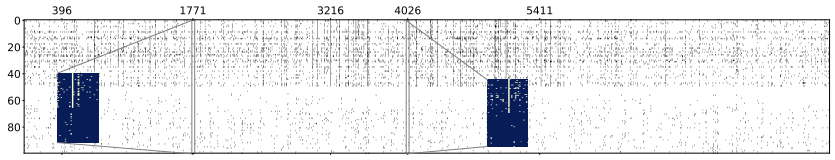


FIGURE 5.7: Visualizing shared dimensions, vectors of 50 closest (top) and most distant (bottom) words to "*mint*" in a `SINr-NR` model trained on `BNC`. The two insets are dimensions on which all 50 closest neighbors have values (white line). Strongest words on dimension 1771: ["*tbsp*", "*oregano*", "*diced*", "*dijon*"]; 4026: ["*rind*", "*juice*", "*lemon*", "*cayenne*"].

If we now focus our interest on the vectors of some words, we can take a look at the dimensions they activate and whether similar words share these dimensions. We visualize in Figures 5.8 and 5.9 the shared dimensions between words that are semantically related. This is a first attempt at exploiting vector-level interpretability.

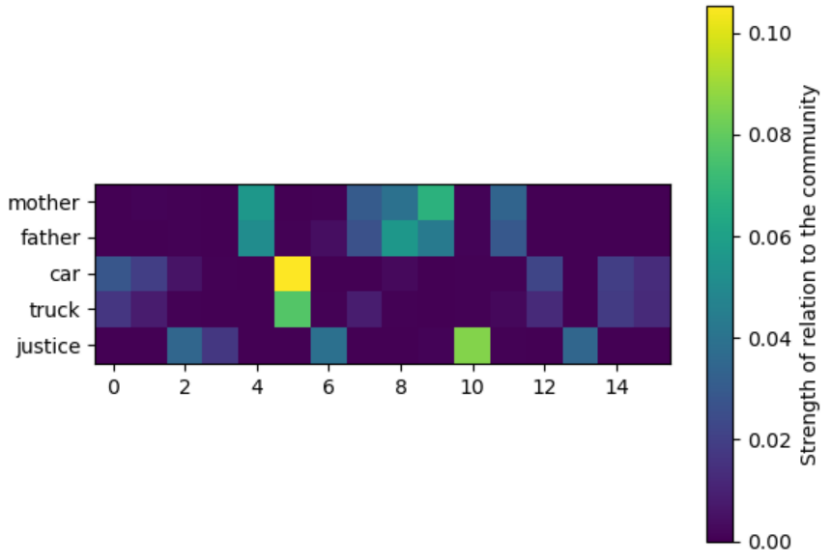


FIGURE 5.8: Common dimensions for five words (vertical axis): *"mother"*, *"father"*, *"car"*, *"truck"*, and *"justice"* in a `SINr-NR` model trained on `BNC`.

We can see here that similar words share common dimensions (*e.g.*, mother/father, car/truck). This echoes with Figure 5.6 where we could see common dimensions shared by similar words. However, there are no common dimensions among the strongest dimensions between words *a priori* less similar (*e.g.*, car/mother). It goes to show that dimensions seem related to topics.

Figure 5.9 presents the shared dimensions for words: "*mint*", "*thyme*", "*insulin*" and "*diabetes*". The first thing we can see is that herbs do not seem

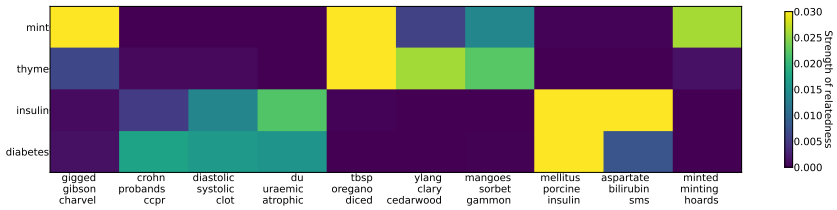


FIGURE 5.9: Common dimensions labeled with their strongest words (horizontal axis) for four words (vertical axis): *"mint"*, *"thyme"*, *"insulin"* and *"diabetes"* in a `SINr-NR` model trained on `BNC`.

to share strong dimensions with diabetes-related terms. Secondly, "*thyme*" and "*mint*" share the same dimensions as in Table 2.2 but also dimensions about herbs and wood sought after for their essential oils (*"ylang"*, *"clary"*, *"cedarwood"*). Dimensions related to *"diabetes"* and *"insulin"* are a mix of cardiovascular terms (*"diastolic"*, *"systolic"*, *"clot"*) and words related to the production of *"insulin"* (*"melitus"*, *"porcine"*, *"insulin"*). We see that similar words share dimensions that are themselves relevant.

We have interpreted some vectors and dimensions showing the potential of interpretable word embedding models to provide insight into the input data and what has been captured by the algorithm. In Appendix B, we provide additional visualizations on vectors extracted on `BNC` with `SINr-NR`.

5.7 SUMMARY

This last chapter focused on the interpretability of word embeddings. The key takeaways from this chapter are the following:

(i) Explainability and interpretability of machine learning algorithms are on the rise with the popularization of algorithmic processing in many domains. With algorithms helping decisions making in high-stake domains such as medicine, justice or finance, understanding the mechanics of algorithms becomes crucial. Explainable models provide means of explaining the process which led to a decision. Interpretable models' internal mechanics can be audited and make sense for humans.

(ii) Interpretable word embedding approaches are mainly focused on dimension interpretability (words with a high value on a dimension are semantically related). Three conditions contribute to interpretability: large number of dimensions, sparsity, and quality of representations.

(iii) `SINr-NR` has on-par performances with `SPINE` and is thus state-of-the-art for dimension interpretability with the *word intrusion* evaluation. `SINr-NR` is also explainable as we can trace back the formation of embeddings to the community structure in the network.

(iv) With the help of additional criteria, interpretability can be extended to understand the structure of vectors with vector-level interpretability. Having a stable model across runs guarantees that interpretations derived from a model will hold true for another model trained with the same algorithm on the same data. Reducing the number of activated dimensions per word (sparsifying vectors) opens the possibility

to analyze the relevance of a reduced set of dimensions representing a word.

(v) Increasing sparsity reduces noise in representations and helps word embedding models achieve better performances on *pairwise word similarity* evaluations. Interpretable word embeddings of similar words display common activated dimensions that make sense with the terms they are part of the representation.

Our evaluation of dimension interpretability of French and our introduction of vector-level interpretability thanks to our sparsification process opened new opportunities in the field of interpretable word embedding models. We talk more in detail about the perspectives of our work on interpretability Chapter 6 and particularly with a human evaluation of vector-level interpretability, an automatized evaluation of interpretability and interpretability in the context of diachronic (temporal) word embeddings.

Part III

EPILOGUE

6

Outlook

6.1 LOOKING BACK

Complex systems contain information that is organically organized and give rise to complex phenomenons. With many interacting elements, it is hard to capture the role and the position of each of a network's component without a representation summarizing this information. That is where graph embedding, and in NLP word embedding, find their purpose. They allow compressing information based on the distributional hypothesis. Graph and word embedding methods share philosophies and techniques but also the drawbacks related to these methods: complexity of computation, and lack of interpretability. This thesis demonstrated how representations of complex systems and more particularly complex networks can be built under three constraints to alleviate the limitations of contemporary methods:

(i) Learn graph and word representations that capture the neighborhood and position of an item within its system sustainably. That is, keeping the compute time and resources of representation learning algorithms as low as possible to reduce energy consumption and allow their execution on various hardware settings.

(ii) Provide interpretable representations that can be audited by humans. A human may interpret how the information is organized solely from the vectors, without needing an external post-hoc model to give explanations. More specifically, in the case of word embeddings, an interpretable model presents dimensions whose strongest words are semantically related. This could be extended to the word itself, where the combination of dimensions should make sense regarding the meaning of a word in the language.

(iii) Maintain high-quality representations with respect to constraints (i) and (ii). Frugality and interpretability should not degrade, or marginally, the quality of embeddings. This quality may be evaluated through intrinsic or extrinsic evaluations.

To support these claims, we introduced a set of methods and evaluations:

- A FRAMEWORK AND TWO IMPLEMENTATIONS. We developed the *Lower Dimension Bipartite Graph Framework* (`LDBGF`) and two implementations that rely on community detection in networks to derive representations: `SINr-NR` and `SINr-MF`. To support the three claims previously announced, we carefully performed quantitative and qualitative on graph and word embedding tasks.

- GRAPH EMBEDDING. We demonstrated the capacities of `SINr-NR` and `SINr-MF` to provide insightful node embeddings and capture network information at different levels. Through *link prediction*, graph features predictions, and node clustering, both in a supervised and unsupervised setting, we showed that our model was able to capture microscopic, mesoscopic and macroscopic-level information.

- WORD EMBEDDING. Large text corpora may be represented using word co-occurrence networks, in which words appearing in the vicinity of each other are connected by an edge. Word embeddings may be extracted from co-occurrence networks using `SINr-NR`. We showed that `SINr-NR` is a solid contender for word embedding, providing representations that can achieve good performances on classical evaluation tasks such as the word similarity task. Furthermore, we demonstrated the benefits of preprocessing data before learning representations, which goes against the current trend of using large quantities of raw data. With preprocessing, we can obtain better results on word similarity with quantities of data smaller by multiple orders of magnitude.

- INTERPRETABILITY. We showed that interpretability is an intrinsic property of `SINr-NR`. First, dimensions of the embedding space are interpretable as attested by a human evaluation of the coherence of dimensions: the *word intrusion* evaluation. We then proceeded to refine the definition of interpretability for word embeddings, introducing new characteristics: vector sparseness, stability of representations, paving the way toward vector-level interpretability. Vector sparseness allows lowering the number of active dimensions of vectors. This is a step toward vector-level interpretability, being able to make sense of the combination of dimensions to build word sense. Stability is desirable in order to guarantee that the interpretations drawn from a model are not spurious. These new constraints to foster interpretability may result in improved performances. Finally, we visualized how dimensions are shared between similar words and how interpretations about word sense and model structure could be drawn from simple plots.

Taken together, these contributions show the versatility of community-based graph and word embeddings to provide interpretable and efficient representations with frugality. However, much remains possible to exploit the potential of community-based representations. Perspectives are numerous as this work is at the crossroads of Complex Networks and NLP.

6.2 LOOKING AHEAD

6.2.1 *Investigating the contribution of sparsity by filtering dimensions*

Our work on word embedding sparsification had one clear objective, lowering the number of active dimensions in vectors to reach a low-enough number of dimensions that would allow vector-level interpretability. While experimenting with the sparsification of representations, we noticed that it allowed to improve performances on *pairwise word similarity* evaluations. We conjectured that this sparsification procedure allowed to remove noise in the representation that impeded the *pairwise word similarity* evaluation. Therefore, representations seem to benefit from a post-processing that removes information from the vectors. To that end, we extend our work on sparsification by focusing on dimension-filtering. Filtering dimensions would allow reducing the overall dimension of representations while filtering information in the embedding matrix. Various questions emerge when it comes to altering the embedding matrix. On what basis should a dimension be removed? Is filtering dimensions the same as filtering communities? We investigated a dimension-filtering approach to filter a `SINr-NR` model: `SINr-filtered` in [Bér+24]. We provide some details about the procedure and results in Appendix A. Filtering rarely and very frequently activated dimensions allows improving the performances of `SINr-NR` models on word similarity evaluations. With such a post-processing, that doesn't add a lot of complexity to the pipeline, `SINr-NR` can compete with pretrained language models on the *pairwise similarity evaluation*. Meanwhile, `SINr-NR` maintains a low compute and requires far less text data to achieve similar performances.

[Bér+24] Béranger et al., "Filtering Communities in Word Co-Occurrence Networks to Foster the Emergence of Meaning"

These results are encouraging for the future of our framework, showing that refinements to the pipeline can be made to come closer to state-of-the-art methods with a fraction of the computation power and data required. In this direction, it would be beneficial to evaluate sparsified models on downstream tasks such as the classification of documents to confirm that performances can be generalized in various contexts, and the benefits of interpretability to understand classification decisions. Furthermore, in `SINr-NR`, very specialized words may be represented by equally specialized communities that are activated by very few words. In this context, sparsifying representations may remove relevant information to represent heavily specialized vocabulary. An evaluation targeted toward specialized lexicon could be beneficial to measure the extent of the impact of sparsification on these words.

6.2.2 *Evaluating dimension-level and vector-level interpretability*

Previously, most of the attention in terms of interpretability of models was toward dimensions. How semantically coherent the dimensions of an embedding model are. However, the *word intrusion* evaluation we presented is costly because it requires human subjects. Thus, proposals to automatize interpretability evaluations were made. Sun et al. [Sun+16] introduce the *DistRatio*, that is supposed to score the interpretability of word embeddings without requiring humans and without being subjective. Similarly to the

[Sun+16] Sun et al. "Sparse word embeddings using l1 regularized online learning"

word intrusion evaluation, tasks including top words on a dimension and an intruder are constructed. Then, if the dimension is interpretable, the top words should be dissimilar from the intruder. They define the *DistRatio* as following:

$$\begin{aligned}
\text{IntraDist}_{\,i} &= \sum_{w_j \in \text{top}_k(i)} \sum_{\substack{w_k \in \text{top}_k(i) \\ w_k \neq w_j}} \frac{\text{dist}\left(w_j, w_k\right)}{k(k-1)} \\
\text{InterDist}_{\,i} &= \sum_{w_j \in \text{ top}_{\,k}(i)} \frac{\text{dist}\left(w_j, w_{b_i}\right)}{k} \\
\text{DistRatio} &= \frac{1}{d} \sum_{i=1}^{d} \frac{\text{InterDist}_{\,i}}{\text{IntraDist}_{\,i}}
\end{aligned} \tag{6.1}$$

The IntraDist measures the average distance between top words on dimension i, given k top words (w_j, w_k). The InterDist measures the distance between the top k words w_j on the dimension and the intruder word w_{b_i} for dimension i. The ratio of these two measures (DistRatio), computed for d dimensions (potentially all dimensions of the model) is supposed to inform on interpretability. The higher the ratio, the farther away the intruder is from the top words, and thus the more interpretable dimensions are. Such an evaluation metric can be useful to assess the interpretability of models. However, it relies on dist that would be the cosine distance between vectors. We use this same distance for *pairwise word similarity* evaluations, to measure the coherence of similarity within the model in comparison with human perception of similarity. However, in this case, a correlation is computed between values. We can wonder whether relying just on a ratio, averaged over many dimensions, can indicate that a model is interpretable. Therefore, using the DistRatio as a first intention evaluation of interpretability can guide interpretable model development. Yet, its relevance would benefit from being accompanied by extensive human evaluations that come to the same conclusions regarding word embedding interpretability.

This thesis introduced the idea of vector-level interpretability for word embeddings. We have shown in Figure 5.8 that shared dimensions between similar words seem to be relevant for the representation of *"mint"* and *"thyme"* however, to confirm that sparsity brings increased vector-level interpretability, one would need to design a human evaluation, as what the *word intrusion* evaluation can provide for dimension interpretability. This evaluation would require to ask humans either if the combination of dimensions is coherent for the representation. This could be done by trying to guess the word from its dimensions. If these dimensions are specific enough to the word, one should be able to find it after several attempts. However, this task remains hard to perform, control and score, showing the sheer difficulty of evaluating at the vector-level beyond an exploratory approach.

6.2.3 *Providing temporal and interpretable representations*

So far, our community-based approach to embeddings is static. Temporal embeddings exist both for networks and words and can provide representations that encompass change through time. The goal is to provide a vec-

tor representation that includes the position of the item represented in the current time period while keeping information about its position in previous time slices. For example, diachronic word embeddings may be useful to follow the evolution of words through time, comparing the representation of *"apple"* before and after the emergence of the technology company. As we have seen throughout this work, building static representations that achieve good performances on evaluation tasks is not easy. Doing so in a temporal context is even more complex. Usually, learning temporal embeddings requires learning one embedding space per temporal slice and later aligning the embedding spaces to have a common representation space in which representations can be compared. Learning a single model can come with significant overhead, as we have seen in our experiments. Multiplying the extraction of representations per the number of time slices mechanically increases runtime and resource consumption. Furthermore, learning one embedding space per time slice is not the only step. After a model has been extracted for a temporal slice, all models need to be aligned so that their representations can be compared. This process is another costly step to obtaining a temporal model.

When it comes to contextual word embedding models, they can provide one representation per token. However, the multiplicity of representations for an item complexifies the temporal representation. To have one vector per type, the multiple representations need to be clustered and aggregated together. Yet, again, this adds another layer of complexity and uncertainty related to clustering and aggregation.

For all these reasons, `SINr-NR` could be a good contender to provide temporal models. First, it is frugal, meaning that the extraction of embeddings takes little time and resources. Second, its representations are interpretable by design, providing interesting insight in the model. Third, the performances of `SINr-NR` are stable across runs and achieve good scores on the word similarity task. Finally, the community structure can help avoid aligning embedding spaces *a posteriori*. Instead, all time slices can be gathered into a single network and communities detected. Then, for each time slice, embeddings may be extracted using the community structure of all time slices, and thus, embeddings are projected in the same latent space structured by the communities. Community-based temporal embeddings is a major perspective of this work that should be investigated. Proving such a model with few parameters, low compute time, resources, and interpretability by design could prove desirable to lexicologists studying the evolution of language through time.

A

SINr-filtered

Chapter 5 investigated interpretability and particularly the contribution of sparsity for vector-level interpretability. In our experiments on gradual sparsification of embeddings, we observed that removing activations from vectors could improve performances on the *parwise word similarity* evaluation. Following these results, in Béranger et al. [Bér+24], we kept on exploring the contributions of sparsity and its potential to improve the quality of representations while reducing the amount of information stored.

[Bér+24] Béranger et al. "Filtering Communities in Word Co-Occurrence Networks to Foster the Emergence of Meaning"

a.1 DISTRIBUTION OF COMMUNITY SIZES AND ACTIVATIONS IN SINR-NR

Sizes of communities at the base of `SINr-NR` tend to follow a power-law distribution. With the γ multi-resolution parameter of `Louvain`, we can detect communities of small sizes. However, when we plot the distribution of community sizes, some remain larger than the rest.

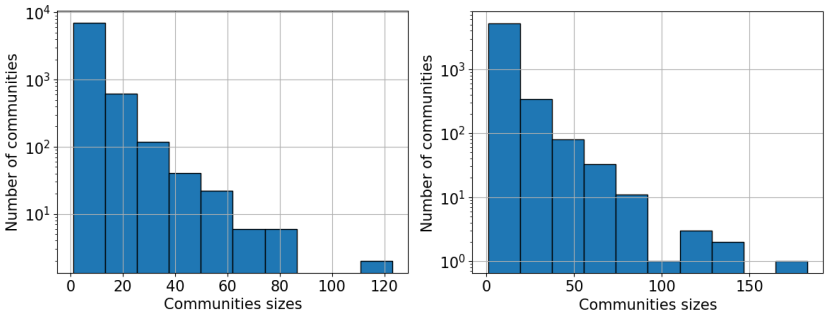


FIGURE A.1: Distribution of community sizes on `BNC` (left) and `UkWac` (right).

Words' occurrences and co-occurrences commonly follow *Zipf's law* (see Chapter 1), which is consistent with a power-law. Some words have significantly more occurrences with the rest of vocabulary, and many words co-occur scarcely with the rest of the vocabulary (only a reduced set of co-occurents). Since communities in `SINr-NR` are composed of words, and community sizes tend to follow a power-law [DBL20], certain communities may be more linked with the rest of the graph than others. Some communities may be scarcely connected with the rest of the graph. In `SINr-NR`, communities determine the dimensions of models. Thus, the distribution of activations (non-zero values in embeddings) may also follow a power-law.

[DBL20] Dao et al., "Community structure: A comparative evaluation of community detection methods"

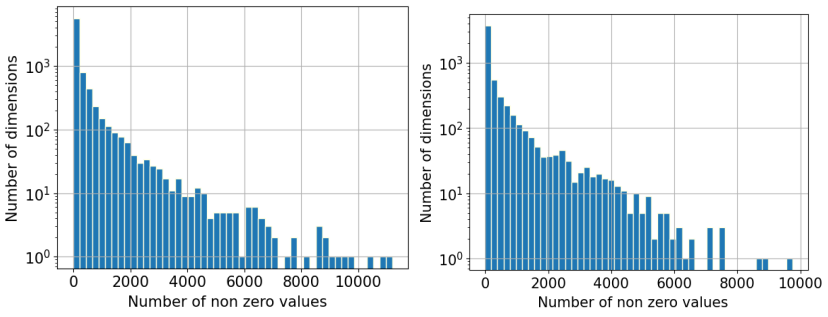


FIGURE A.2: Distribution of the number of activations per dimension on `BNC` (left) and `UkWac` (right).

We see Figure A.2 that the activation of dimensions seems to also follow a power-law. Yet, if we recall our interpretability criteria Chapter 5,

dimensions should be topically consistent. Thus, only a limited number of words should activate one dimension. In the case of these heavily activated dimensions, we can conjecture that they correspond to communities gathering very frequent words that appear in many contexts, and thus they may not be topically consistent.

On the other hand, some dimensions are rarely activated (tail of distribution). They may correspond to very specific communities, not useful to represent most of the vocabulary. If not, they may be noisy dimensions that penalize performance. Yet, these dimensions contribute to the dimension of vectors while being very scarcely activated.

We thus propose to filter-out frequent and rare dimensions and observe the impact of their removal on *pairwise word similarity* results.

a.2 FILTERING DIMENSIONS TO IMPROVE PERFORMANCES

▸ EXPERIMENTAL SETTING

In order to evaluate the impact of dimension removal on *pairwise word similarity* performances, we adopt the same setting as in Chapters 4 and 5. Additionally to `MEN`, `WS353`, and `SCWS`, we chose another dataset: `SimLex`. `SimLex` is split into two subsets `SimLex999` that contains the whole dataset with noun-noun, adjective-adjective, and verb-verb pairs. `SimLex665` contains only the noun-noun pairs. The choice of `SimLex` comes down to its constitution: it did not rely on frequency information, and thus it includes rarer words. Furthermore, the split into two datasets allows seeing if words part-of-speech (noun, adjective, verb, etc.) play a role in word similarity.

We filter dimensions that appear in the head and the tail of the distribution, observing the impact of their removal on *pairwise word similarity*. With this evaluation, we also study the impact of the filtering threshold on performance.

▸ RESULTS

BASELINE. Table A.1, we can see the baseline performances of `SINr-NR` (without dimension filtering), `Word2vec` and `SINr-filtered`, where dimensions have been filtered. With `SINr-filtered`, results are systematically better than `SINr-NR`, catching up with `Word2vec` and coming ahead on `MEN` and `WS353`. If we recall the results Table 4.4, in Bommasani et al. [BDC20], results for `WS353` on a 100B tokens google books corpus where 0.68 for `Word2vec` and 0.73 for `BERT`. `SINr-filtered` achieves higher results than baseline `Word2vec` (0.68) on a significantly larger corpus (`UkWac` has 800M tokens once preprocessed) and comes close to the results of `BERT` (0.73).

[BDC20] Bommasani et al. "Interpreting pretrained contextualized representations via reductions to static embeddings"

FILTERING THE HEAD OF DISTRIBUTION. If we focus on filtering only the head of the distribution, we see that it improves performances. Indeed, Figure A.3, at 12,000 maximum activations per dimension, no filtering is performed. Moving the cursor toward 4,000 activations per vector improves performances for `BNC` and `UkWac`. Between 4,000 and 2,000, performances stagnate and after 2,000, significant information is removed and perfor-

	MEN		WS353		SCWS		SimLex999		SimLex665	
	BNC	UkWac	BNC	UkWac	BNC	UkWac	BNC	UkWac	BNC	UkWac
Word2vec	**.73**	**.75**	.64	.66	**.61**	**.64**	**.28**	**.34**	**.34**	**.37**
SINr-NR	.67	.70	.63	.68	.56	.56	.20	.23	.28	.30
SINr-filtered	.72	**.75**	**.65**	**.70**	.58	.59	.25	.25	.30	.33

TABLE A.1: Summary of the results of competing models and of SINr and its filtered version, SINr-filtered, introduced in [Bér+24].

mances degrade. At 4,000 maximum activations per dimension, SINr-NR catches-up with Word2vec, gaining 5 points on MEN, 2 points for WS353 and WS353. SimLex is a harder dataset due to the rarity of some words and the variety of parts-of-speech. However, we see a similar behavior between the 2,000 and 4,000 mark. Filtering the head of the distribution with a threshold at 4,000 only removes on average 95 dimensions on BNC and 90 on UkWac.

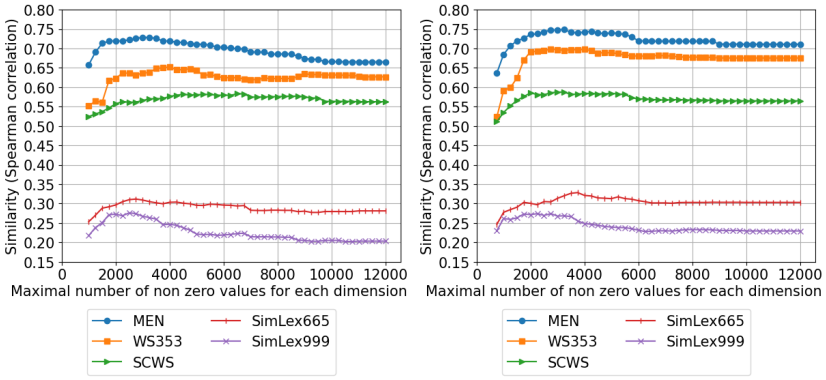


FIGURE A.3: Similarity results according to the maximum number of activations per dimension on BNC (left) and UkWac (right). Filtering frequently activated dimensions.

FILTERING THE TAIL OF THE DISTRIBUTION. Filtering the tail of the distribution has a different effect on performance. Figure A.4, filtering rare dimensions does not allow gaining performances. Yet, filtering dimensions with fewer than 500 activations does not cause significant loss. However, it allows reducing the number of dimensions by 5: 6,600 to 1,200 for BNC, and 5,700 to 1,100 for UkWac. It is a positive result that goes in the direction of reducing the memory footprint of models.

IMPACT OF FILTERING ON DISTRIBUTION OF COMMUNITIES. Since dimensions are related to communities, their removal means that the community is not used to represent words. Therefore, we could wonder whether these frequently and rarely activated dimensions correspond to large and small communities. Figure A.5, filtering frequently activated dimensions (4,000 threshold) mostly removes larger communities (over 150 words on UkWac, 60 to 80 on BNC). When filtering rarely activated dimensions (500 activations), smaller communities are removed, although most of them remain. It also impacts larger communities (60 words on BNC, 50 words on UkWac). Thus, filtering based on dimensions is different than filtering based on community sizes.

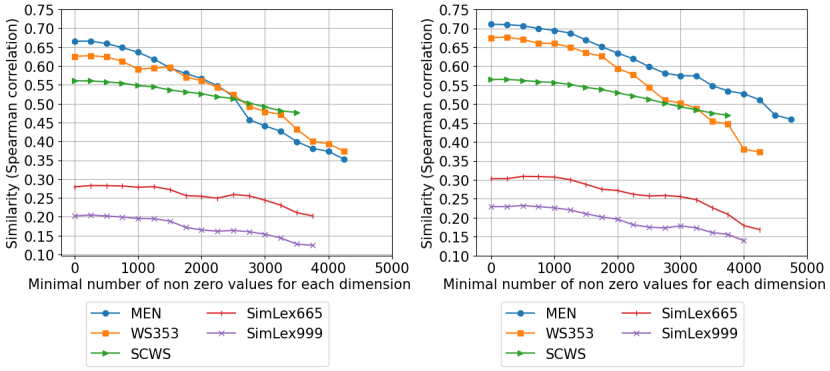


FIGURE A.4: Similarity results according to the minimum number of activations per dimension on on BNC (left) and UkWac (right). Filtering rarely activated dimensions.

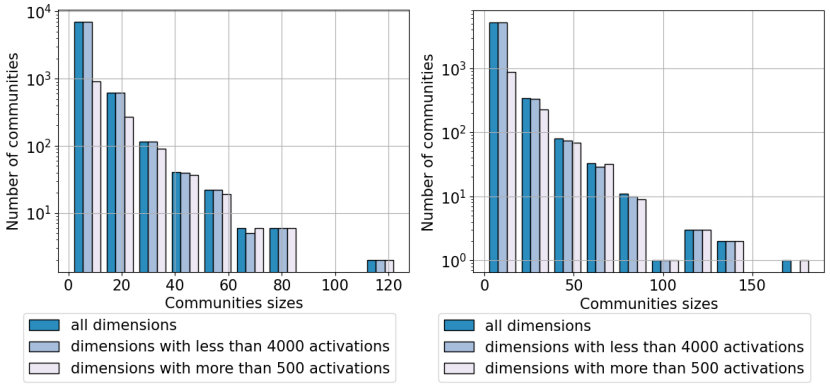


FIGURE A.5: Effects of filtering on community distribution on BNC (left) and UkWac (right).

a.3 SUMMARY

[Bér+24] Béranger et al. "Filtering Communities in Word Co-Occurrence Networks to Foster the Emergence of Meaning"

[GPD23b] Guillot et al. "Sparser is better: one step closer to word embedding interpretability"

In summary, our experiment in Béranger et al. [Bér+24] have extended our work on sparsification. When our setting in Guillot et al. [GPD23b] was to sparsify each vector and only keeping the *top-k* activations. In [Bér+24] we filter dimensions based on the number of words that activate them. Sparsification in these two settings has one purpose, simplifying the model for increased interpretability and performance. The main findings of Béranger et al. [Bér+24] are:

(i) Filtering most frequently activated dimensions helps gaining performances and coming closer to `Word2vec`. `SINr-filtered` can even achieve better results than `BERT` with a fraction of the training data.

(ii) Filtering rare dimensions can help reduce model size by 5 while preserving performances.

Thus, filtering dimensions with `SINr-filtered` is a beneficial post-processing step to improve the performances of `SINr-NR`. It would be interesting to extend this study to other types of corpora, especially domain-specific corpora. Communities of smaller sizes could be observed, and filtering rarer dimensions could have a different impact. Furthermore, evaluating the interpretability of `SINr-filtered` with a human evaluation would allow as-

sessing whether it improves interpretability both at the dimension-level and the vector-level.

B

SINr library: visualizing vectors

Learning and interpreting word embeddings with SINr

This notebook aims at demonstrating how to learn word embeddings with SINr and use the library to interpret representations. In the first part of this demonstration, we describe the steps required to learn a SINr model. In the second part of our demonstration, we investigate the representations of "silver" whose representation seems heavily influenced by metals, colors and also by the fact that it is a material used to build medals. Our second analysis revolves around "wine" and spirits. We will see that the representation of "wine" is a mix of its grape variety and measures of quantities. Finally, when investigating the representation of different spirits, we can see that they share dimensions related to wine but also to cocktails.

I. Learning a model

1. After installing SINr (available through `pip`), the first step to learning SINr embeddings is to load (or) extract a word co-occurrence matrix from a large corpus (here BNC). This co-occurrence matrix can be extracted with SINr (co-occurrence extraction is not demonstrated for conciseness purposes).

```
!pip install sinr
import sinr
model = sinr.load_from_cooc_pkl("matrix_bnc.pk")
```

```
2024-04-25 17:36:14,374 - load_from_cooc_pkl - INFO - Building Graph.
2024-04-25 17:36:14,375 - load_pkl_text - INFO - Loading cooccurrence matr
ix and dictionary.
2024-04-25 17:36:14,505 - load_pkl_text - INFO - Finished loading data.
2024-04-25 17:36:25,542 - load_from_cooc_pkl - INFO - Finished building gr
aph.
```

2. Using the word co-occurrence matrix, communities (Louvain) can be detected. The γ (multi-resolution) parameter allows prevent Louvain from providing large communities.

```
import sinr.graph_embeddings as ge
communities = model.detect_communities(gamma=30)
```

```
2024-04-25 17:36:30,518 - detect_communities - INFO - Detecting communitie
s.
```

```
Gamma for louvain : 30
Communities detected in 4.01269 [s]
solution properties:
-------------------  ------------
# communities        4759
min community size      1
max community size    271
avg. community size    11.7411
modularity              0.0398516
-------------------  ------------
2024-04-25 17:36:34,733 - detect_communities - INFO - Finished detecting c
ommunities.
```

3. Once communities have been detected, SINr can leverage the community structure to weight each word vector.

```
model.extract_embeddings(communities)
```

```
2024-04-25 17:36:39,987 - extract_embeddings - INFO - Extracting embedding
s.
2024-04-25 17:36:39,989 - extract_embeddings - INFO - Applying NFM.
2024-04-25 17:36:39,990 - get_nfm_embeddings - INFO - Starting NFM
2024-04-25 17:38:34,248 - extract_embeddings - INFO - NFM successfully app
lied.
2024-04-25 17:38:34,250 - extract_embeddings - INFO - Finished extracting
embeddings.
```

4. Once embeddings have been extracted, a `SINrVectors` is used to store the output. `SINrVectors` object come with many function to audit and explore the representation of words.

```
sinr_vectors = ge.InterpretableWordsModelBuilder(model, "oanc",
n_jobs=8, n_neighbors=15).build()
```

II. Interpreting and visualizing dimensions and communities

A. Investigating "silver"

1. Most similar words (neighbors)

```
sinr_vectors.most_similar("silver")
```

```
{'object ': 'silver',
 'neighbors ': [('gold', 0.92),
  ('bronze', 0.77),
  ('medal', 0.76),
  ('bullion', 0.76),
  ('coins', 0.75),
  ('hoard', 0.74),
  ('bracelet', 0.73),
  ('emboss', 0.73),
  ('ingot', 0.73),
  ('bangle', 0.73),
  ('filigree', 0.72),
  ('debasement', 0.71),
  ('brooch', 0.71),
  ('necklace', 0.71)]}
```

2. Community of "silver" and members (descriptors)

```
sinr_vectors.get_dimension_descriptors("silver", topk=8)
```

```
{'dimension': 1894, 'descriptors': [(0.17, 'medallist'), (0.12, 'medal'),
(0.11, 'debasement'), (0.11, 'the_royal_mint'), (0.11, 'cufflink'), (0.1,
'ingot'), (0.1, 'bullion'), (0.1, 'commonwealth_games')]}
```

3. Strongest dimensions associated with "silver" and members of the corresponding community (descriptors)

```
sinr_vectors.get_obj_descriptors("silver", topk_dim=3,topk_val=5)
```

```
[{'dimension': 1894,
  'value': True,
  'descriptors': [(0.17, 'medallist'),
   (0.12, 'medal'),
   (0.11, 'debasement'),
   (0.11, 'the_royal_mint'),
   (0.11, 'cufflink')]},
 {'dimension': 1893,
  'value': True,
  'descriptors': [(0.34, 'cyan'),
   (0.22, 'magenta'),
   (0.2, 'viridian'),
   (0.19, 'ultramarine'),
   (0.16, 'sienna')]},
 {'dimension': 1473,
  'value': True,
  'descriptors': [(0.18, 'arsenical'),
   (0.12, 'antimony'),
   (0.11, 'zinc'),
   (0.11, 'molybdenum'),
   (0.11, 'nickel')]}]
```

4. Words with the strongest values (stereotypes) on the dimension of "silver"

```
```

```
sinr_vectors.get_dimension_stereotypes("silver", topk=5)
```

```
{'dimension': 1894, 'stereotypes': [(0.22, 'nine_carat'), (0.17, 'medallis
t'), (0.12, 'steve_redgrave'), (0.12, 'medal'), (0.11, 'the_royal_mint')]}
```

B. Investigating "wine" and other spirits

1. Community of "wine" and members (descriptors)

```
sinr_vectors.get_dimension_stereotypes("wine", topk=10)
```

```
{'dimension': 1504, 'stereotypes': [(0.26, 'sauvignon'), (0.24, 'cabernet'
), (0.22, 'pinot'), (0.2, 'pinot_noir'), (0.19, 'chardonnay'), (0.17, 'the
_montagne_de_reims'), (0.17, 'aube'), (0.16, 'uncork'), (0.15, 'riesling')
, (0.15, 'marne')]}
```

2. Three words that have high values on the 5 dimensions that are useful to describe "wine"

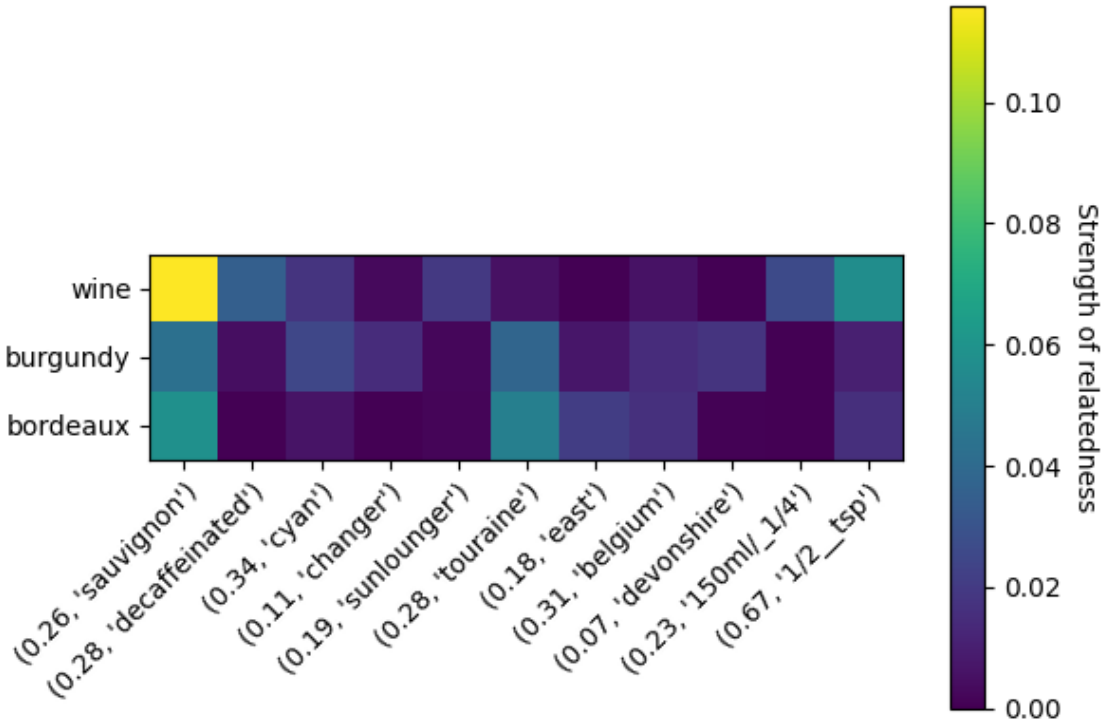
```
sinr_vectors.get_obj_stereotypes("wine", topk_dim=4, topk_val=3)
```

```
[{'dimension': 1504,
  'value': True,
  'stereotypes': [(0.26, 'sauvignon'), (0.24, 'cabernet'), (0.22, 'pinot')
]},
 {'dimension': 60,
  'value': True,
  'stereotypes': [(0.67, '1/2__tsp'), (0.6, '5ml/tsp'), (0.48, '1_tsp')]},
 {'dimension': 2784,
  'value': True,
  'stereotypes': [(0.28, 'decaffeinated'), (0.15, 'cup'), (0.14, 'cola')]}
,
 {'dimension': 182,
  'value': True,
  'stereotypes': [(0.23, '150ml/_1/4'),
   (0.23, '300ml/_1/2'),
   (0.2, 'kaliber')]}]
```

3. Visualizing common dimensions between "wine", "burgundy" and "bordeaux"

```
from sinr import viz
sinr_viz = viz.SINrViz(sinr_vectors)
sinr_viz.compare_stereotypes(["wine", "burgundy", "bordeaux"])
```

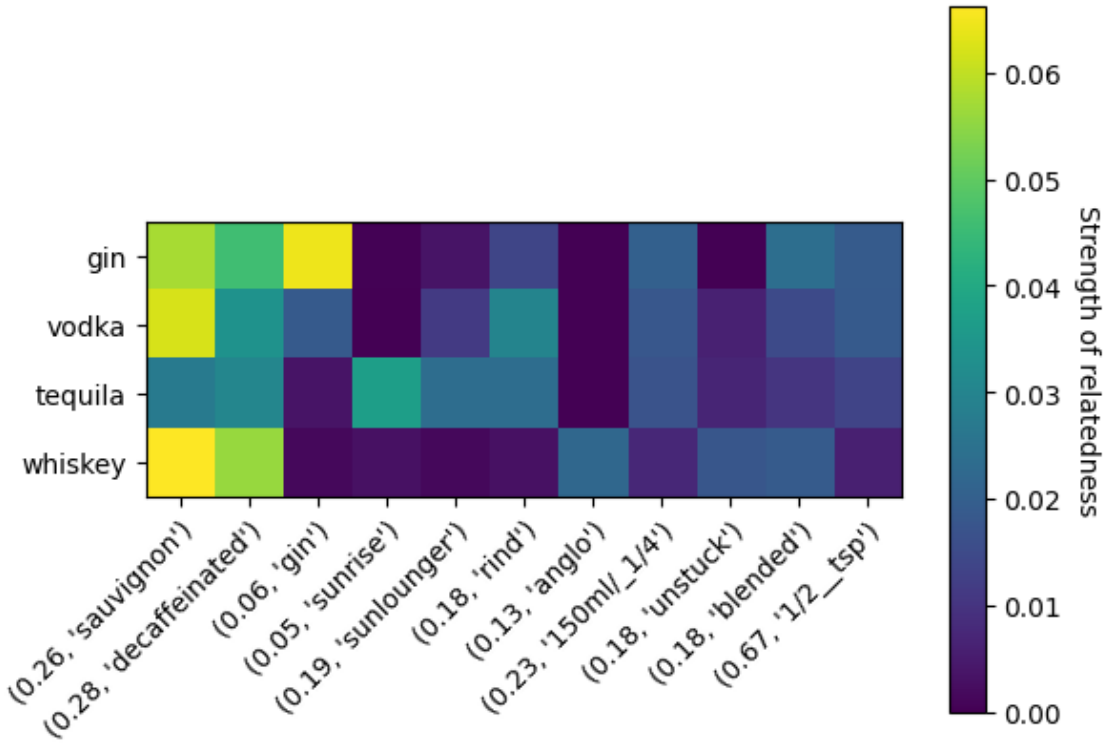
```
{1504, 2784, 1893, 202, 2379, 139, 74, 2219, 2479, 182, 60}
```



3. Visualizing common dimensions between spirits

```
sinr_viz.compare_stereotypes(["gin", "vodka", "tequila", "whiskey"])
```

```
{1504, 2784, 1444, 1480, 2379, 2354, 1682, 182, 2295, 2520, 60}
```



References

[Aar13] Scott Aaronson. “Why Philosophers Should Care about Computational Complexity”. In: *Computability*. MIT Press, 2013, pp. 261–328. DOI: 10.7551/mitpress/8009.003.0011 (cit. on p. 10).

[AA03] Lada A Adamic and Eytan Adar. “Friends and neighbors on the Web”. In: *Social Networks* 25.3 (2003), pp. 211–230. DOI: 10.1016/S0378-8733(03)00009-1 (cit. on p. 74).

[Ale03] Popescul Alexandrin. “Statistical relational learning for link prediction”. In: *IJCAI'03 Workshop on Learning Statistical Models from Relational Data*. 2003. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=67dc9b7c747436f9c8e74813d0ad4098428b2647 (cit. on p. 73).

[AP05] Abdulrahman Almuhareb and Massimo Poesio. “Concept learning and categorization from the web”. In: *proceedings of the annual meeting of the Cognitive Science society*. Vol. 27. 2005. URL: https://escholarship.org/content/qt8gh9h462/qt8gh9h462.pdf (cit. on p. 100).

[Als+19] Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. “Publicly Available Clinical BERT Embeddings”. In: *Proceedings of the 2nd Clinical Natural Language Processing Workshop*. Association for Computational Linguistics, 2019, pp. 72–78. DOI: 10.18653/v1/W19-1909 (cit. on p. 33).

[Ang+16] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. “Machine Bias”. In: *Propublica* (2016). URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing (cit. on p. xxi).

[Ani+23] Rohan Anil et al. *PaLM 2 Technical Report*. 2023. arXiv: 2305.10403 [cs.CL]. URL: https://arxiv.org/abs/2305.10403 (cit. on p. xx).

[Ant71] Jac M. Anthonisse. “The rush in a directed graph”. In: *Journal of Computational Physics* (1971), pp. 1–10. URL: https://api.semanticscholar.org/CorpusID:118421505 (cit. on p. 52).

[Ari24] Aristotle. *Metaphysics*. Oxford University Press, 1924 (cit. on p. 8).

[ABG21] Nora Assouli, Khelifa Benahmed, and Brahim Gasbaoui. “How to predict crime informatics-inspired approach from link prediction”. In: *Physica A: Statistical Mechanics and its Applications* 570 (2021). DOI: 10.1016/j.physa.2021.125795 (cit. on p. 73).

[AZL22] Eric Austin, Osmar R. Zaane, and Christine Largeron. “Community Topic: Topic Model Inference by Consecutive Word Community Discovery”. In: *Proceedings of the 29th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, 2022, pp. 971–983. URL: https://aclanthology.org/2022.coling-1.81 (cit. on p. 55).

[Bak18] Amir Bakarov. “A Survey of Word Embeddings Evaluation Methods”. In: *arXiv preprint arXiv:1801.09536* (2018). URL: http://arxiv.org/abs/1801.09536 (cit. on p. 99).

[BA99] Albert László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *Science* (1999). DOI: 10.1126/science.286.5439.509 (cit. on pp. 17, 18, 74).

[BP16] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, 2016. URL: http://barabasi.com/networksciencebook/ (cit. on p. 13).

[Bar+09] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. "The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora". In: *LREC'09* 43.3 (2009), pp. 209–226. URL: http://www.jstor.org/stable/27743614 (cit. on p. 105).

[BDK14] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors". In: *ACL'14*. 2014, pp. 238–247. DOI: 10.3115/v1/P14-1023 (cit. on p. 98).

[BL11] Marco Baroni and Alessandro Lenci. "How we BLESSed distributional semantic evaluation". In: *GEometrical Models of Natural Language Semantics* (2011), pp. 1–10. URL: http://dl.acm.org/citation.cfm?id=2140490.2140491 (cit. on p. 100).

[BN03] Mikhail Belkin and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396. DOI: 10.1162/089976603321780317 (cit. on p. 34).

[BK20] Emily M. Bender and Alexander Koller. "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data". In: *ACL'20*. ACL'20, 2020, pp. 5185–5198. DOI: 10.18653/v1/2020.acl-main.463 (cit. on p. 27).

[Bér+24] Anna Béranger, Nicolas Dugué, Simon Guillot, and Thibault Prouteau. "Filtering Communities in Word Co-Occurrence Networks to Foster the Emergence of Meaning". In: *Complex Networks & Their Applications XII*. Springer Nature Switzerland, 2024, pp. 377–388. DOI: 10.1007/978-3-031-53468-3_32 (cit. on pp. 103, 137, 143, 145, 146).

[BG07] Indrajit Bhattacharya and Lise Getoor. "Collective Entity Resolution in Relational Data". In: *ACM Trans. Knowl. Discov. Data* 1.1 (2007), 5–es. DOI: 10.1145/1217299.1217304 (cit. on pp. 71, 72).

[Bho+20] Ayan Kumar Bhowmick, Koushik Meneni, Maximilien Danisch, Jean-Loup Guillaume, and Bivas Mitra. "LouvainNE: Hierarchical Louvain Method for High Quality and Scalable Network Embedding". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. ACM, 2020, pp. 43–51. DOI: 10.1145/3336191.3371800 (cit. on pp. 66, 68, 69, 71, 81).

[Bil92] Enki Bilal. *Froid équateur*. Humanoïdes associés, 1992 (cit. on p. 72).

[Blo+08] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008. DOI: 10.1088/1742-5468/2008/10/P10008 (cit. on pp. 48, 52).

[Boj+17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* (2017). DOI: 10.1162/tacl_a_00051 (cit. on p. 31).

[BDC20] Rishi Bommasani, Kelly Davis, and Claire Cardie. "Interpreting pretrained contextualized representations via reductions to static embeddings". In: *ACL'20*. 2020, pp. 4758–4781. DOI: 10.18653/v1/2020.acl-main.431 (cit. on pp. 107, 144).

[Bon+17] Stephen Bonner, John Brennan, Ibad Kureshi, Georgios Theodoropoulos, Andrew Stephen McGough, and Boguslaw Obara. "Evaluating the quality of graph embeddings via topological feature reconstruction". In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 2691–2700. DOI: 10.1109/BigData.2017.8258232 (cit. on pp. 71, 78–81).

[Bou+23] Zakaria Bouhoun, Theo Guerrois, Xianli Li, Mouna Baker, Nassara Elhadji Ille Gado, Emir Roumili, Francesco Vitillo, Lies Benmiloud Bechet, and Robert Plana. "Information Retrieval Using Domain Adapted Language Models: Application to Resume Documents for HR Recruitment Assistance". In: *International Conference on Computational Science and Its Applications*. Springer. 2023, pp. 440–457 (cit. on p. xx).

[BP98] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual Web search engine". In: *Computer Networks and ISDN Systems* 30.1-7 (1998), pp. 107–117. DOI: 10.1016/S0169-7552(98)00110-X (cit. on p. 78).

[BGV19] Robin Brochier, Adrien Guille, and Julien Velcin. "Global Vectors for Node Representations". In: *WWW'19*. WWW '19. San Francisco, CA, USA: Association for Computing Machinery, 2019, pp. 2587–2593. DOI: 10.1145/3308558.3313595 (cit. on p. 36).

[Bro+21] David A Broniatowski et al. "Psychological foundations of explainability and interpretability in artificial intelligence". In: *NIST, Tech. Rep* (2021). DOI: 10.6028/NIST.IR.8367 (cit. on pp. 114, 115).

[Bro+20] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf (cit. on p. xx).

[BTB14] E. Bruni, N. K. Tran, and M. Baroni. "Multimodal Distributional Semantics". In: *Journal of Artificial Intelligence Research* 49 (2014), pp. 1–47. DOI: 10.1613/jair.4135 (cit. on p. 99).

[CS01] Ramon Ferrer I. Cancho and Richard V. Solé. "The small world of human language". In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 268.1482 (2001), pp. 2261–2265. DOI: 10.1098/rspb.2001.1800 (cit. on pp. 16, 18).

[CLX15] Shaosheng Cao, Wei Lu, and Qiongkai Xu. "GraRep: Learning Graph Representations with Global Structural Information". In: *CIKM*. 2015, pp. 891–900. DOI: 10.1145/2806416.2806512 (cit. on p. 66).

[CLX16] Shaosheng Cao, Wei Lu, and Qiongkai Xu. "Deep neural networks for learning graph representations". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 2016. DOI: 10.1609/aaai.v30i1.10179 (cit. on p. 66).

[CCP18] Tanmoy Chakraborty, Zhe Cui, and Noseong Park. "Metadata vs. Ground-truth: A Myth behind the Evolution of Community Detection Methods". In: *WWW'18*. ACM Press, 2018, pp. 45–46. DOI: 10.1145/3184558.3186921 (cit. on p. 21).

[Cha+09] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-graber, and David Blei. "Reading Tea Leaves: How Humans Interpret Topic Models". In: *NIPS'09*. Vol. 22. Curran Associates, Inc., 2009. URL: https://proceedings.neurips.cc/paper/2009/hash/f92586a25bb3145facd64ab20fd554ff-Abstract.html (cit. on pp. 53, 120).

[CZG08] Jiyang Chen, Osmar R. Zaïane, and Randy Goebel. "An Unsupervised Approach to Cluster Web Search Results Based on Word Sense Communities". In: *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 1. 2008, pp. 725–729. DOI: 10.1109/WIIAT.2008.24 (cit. on p. 49).

[CG16] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *SIGKDD'16*. ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785 (cit. on p. 83).

[CMN08] Aaron Clauset, Cristopher Moore, and M. E. J. Newman. "Hierarchical structure and the prediction of missing links in networks". In: *Nature* 453.7191 (2008), pp. 98–101. DOI: 10.1038/nature06830 (cit. on p. 76).

[CL14] E. Côme and P. Latouche. *Model selection and clustering in stochastic block models with the exact integrated complete data likelihood*. 2014. URL: http://arxiv.org/abs/1303.2962 (cit. on p. 46).

[CD19] Victor Connes and Nicolas Dugué. "Apprentissage de plongements lexicaux par une approche réseaux complexes". In: *TALN'19*. 2019. URL: https://hal.science/hal-02408156 (cit. on pp. 49, 51, 57).

[Con07] BNC Consortium. *British National Corpus, XML edition*. 2007. URL: http://hdl.handle.net/20.500.12024/2554 (cit. on p. 105).

[DBL20] Vinh Loc Dao, Cécile Bothorel, and Philippe Lenca. "Community structure: A comparative evaluation of community detection methods". In: *Network Science* 8.1 (2020), pp. 1–41. DOI: 10.1017/nws.2019.59 (cit. on p. 143).

[DBL17] Vinh-Loc Dao, Cécile Bothorel, and Philippe Lenca. "Community detection methods can discover better structural clusters than ground-truth communities". In: *ASoNAM'17*. ACM, 2017, pp. 395–400. DOI: 10.1145/3110025.3110053 (cit. on p. 21).

[De 23] Alex De Vries. "The growing energy footprint of artificial intelligence". In: *Joule* 7.10 (2023), pp. 2191–2194. DOI: 10.1016/j.joule.2023.09.004 (cit. on pp. xx, xxi, 38).

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (2019), pp. 4171–4186. DOI: 10.18653/v1/N19-1423 (cit. on p. 33).

[DH73] W. E. Donath and A. J. Hoffman. "Lower Bounds for the Partitioning of Graphs". In: *IBM Journal of Research and Development* 17.5 (1973), pp. 420–425. DOI: 10.1147/rd.175.0420 (cit. on p. 46).

[DM01] Sergey N. Dorogovtsev and José Fernando F. Mendes. "Language as an evolving word web". In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 268.1485 (2001), pp. 2603–2606. DOI: 10.1098/rspb.2001.1824 (cit. on p. 18).

[DF18] Julia Dressel and Hany Farid. "The accuracy, fairness, and limits of predicting recidivism". In: *Science Advances* 4.1 (2018), pp. 1–6. DOI: 10.1126/sciadv.aao5580 (cit. on pp. xxi, 39).

[DLP19] Nicolas Dugué, Jean-Charles Lamirel, and Anthony Perez. "Bringing a Feature Selection Metric from Machine Learning to Complex Networks". In: *Complex Networks and Their Applications VII*. Springer International Publishing, 2019, pp. 107–118. DOI: 10.1007/978-3-030-05414-4_9 (cit. on p. 52).

[DP22] Nicolas Dugué and Anthony Perez. "Direction matters in complex networks: A theoretical and applied study for greedy modularity optimization". In: *Physica A: Statistical Mechanics and its Applications* 603 (2022), p. 127798. DOI: 10.1016/j.physa.2022.127798 (cit. on p. 46).

[Dum+88] Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, and Richard Harshman. "Using latent semantic analysis to improve access to textual information". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1988, pp. 281–285. DOI: 10.1145/57167.57214 (cit. on p. 29).

[EP23] F. Eggenberger and G. Pólya. "Über die Statistik verketteter Vorgänge". In: *ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik* 3.4 (1923), pp. 279–289. DOI: 10.1002/zamm.19230030407 (cit. on p. 17).

[ER58] Paul L. Erdos and Alfréd Rényi. "On random graphs. I." In: *Publicationes Mathematicae Debrecen* (1958). URL: https://api.semanticscholar.org/CorpusID:253789267 (cit. on p. 14).

[EGP66] Paul Erdös, A. W. Goodman, and Louis Pósa. "The Representation of a Graph by Set Intersections". In: *Canadian Journal of Mathematics* 18 (1966), pp. 106–112. DOI: 10.4153/CJM-1966-014-3 (cit. on p. 44).

[ESS08] ESSLLI. *Shared Tasks from the ESSLLI 2008 Workshop*. 2008. URL: http://wordspace.collocations.de/doku.php/data:esslli2008:start (cit. on p. 101).

[Est23] Ernesto Estrada. "What is a Complex System, After All?" In: *Foundations of Science* (2023). DOI: 10.1007/s10699-023-09917-w (cit. on p. 7).

[EC16] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)*. 2016 (cit. on p. 117).

[Far+15] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. "Sparse Overcomplete Word Vector Representations". In: *ACL-JCNLP'15*. Association for Computational Linguistics, 2015, pp. 1491–1500. DOI: 10.3115/v1/P15-1144 (cit. on pp. 119, 120, 126, 129).

[Fie73] Miroslav Fiedler. "Algebraic connectivity of graphs". In: *Czechoslovak Mathematical Journal* 23.2 (1973), pp. 298–305. DOI: 10.21136/CMJ.1973.101168 (cit. on p. 46).

[Fin+01] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. "Placing search in context: The concept revisited". In: *WWW* (2001), pp. 406–414. DOI: 10.1145/371920.372094 (cit. on p. 100).

[Fir57] J R Firth. "A synopsis of linguistic theory 1930-55." In: *The Philological Society* (1957) (cit. on pp. 26–28, 49).

[FB07] Santo Fortunato and Marc Barthélemy. "Resolution limit in community detection". In: *Proceedings of the National Academy of Sciences* 104.1 (2007), pp. 36–41. DOI: 10.1073/pnas.0605965104 (cit. on pp. 21, 48).

[Fre77] Linton C. Freeman. "A Set of Measures of Centrality Based on Betweenness". In: *Sociometry* 40.1 (1977), pp. 35–41. URL: http://www.jstor.org/stable/3033543 (cit. on p. 52).

[Fre78] Linton C. Freeman. "Centrality in social networks conceptual clarification". In: *Social Networks* 1.3 (1978), pp. 215–239. DOI: 10.2307/3033543 (cit. on p. 79).

[Gar+18] Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. "Word embeddings quantify 100 years of gender and ethnic stereotypes". In: *Proceedings of the National Academy of Sciences* 115.16 (2018), pp. 3635–3644. DOI: 10.1073/pnas.1720347115 (cit. on p. 123).

[Gar+01] P. Garrard, M. A. Lambon Ralph, J. R. Hodges, and K. Patterson. "Prototypicality, distinctiveness, and intercorrelation: Analyses of the semantic attributes of living and nonliving concepts". In: *Cognitive Neuropsychology* 18.2 (2001), pp. 125–174. DOI: 10.1080/02643290042000053 (cit. on p. 126).

[Gef+17] Alexandre Gefen, Mark Andrew Algee-Hewitt, David McClure, Frédéric Glorieux, Marianne Reboul, JD Porter, and Marine Riguet. "Vector based measure of semantic shifts across different cultural corpora as a proxy to comparative history of ideas". In: *JADH'17* (2017), p. 12. URL: https://hal.science/hal-03168025 (cit. on p. 123).

[Gel95] Murray Gell-Mann. "What is complexity? *Remarks on simplicity and complexity by the Nobel Prize-winning author of* The Quark and the Jaguar". In: *Complexity* 1.1 (1995), pp. 16–19. DOI: 10.1002/cplx.6130010105 (cit. on p. 7).

[Geo05] GeoNames. *All Cities with a population > 1000*. 2005. URL: https://public.opendatasoft.com/explore/dataset/geonames-all-cities-with-a-population-1000/ (cit. on pp. 17, 18).

[GN02] Michelle Girvan and Mark EJ Newman. “Community structure in social and biological networks”. In: *Proceedings of the national academy of sciences* 99.12 (2002), pp. 7821–7826. DOI: 10.1073/pnas.122653799 (cit. on pp. 20, 45, 46).

[GR03] Debra S. Goldberg and Frederick P. Roth. “Assessing experimentally derived interactions in a small world”. In: *Proceedings of the National Academy of Sciences* 100.8 (2003), pp. 4372–4376. DOI: 10.1073/pnas.0735871100 (cit. on p. 72).

[GO13] Yoav Goldberg and Jon Orwant. “A Dataset of Syntactic-Ngrams over Time from a Very Large Corpus of English Books”. In: (2013), pp. 241–247. URL: https://aclanthology.org/S13-1035 (cit. on p. 49).

[GV13] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013 (cit. on p. 46).

[Gou10] S.J. Gould. *The Panda’s Thumb: More Reflections in Natural History*. W. W. Norton, 2010. URL: https://books.google.fr/books?id=z0XY7Rg_lOwC (cit. on p. 6).

[GL16] Aditya Grover and Jure Leskovec. “node2vec: Scalable Feature Learning for Networks”. In: *SIGKDD’16*. ACM, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754 (cit. on p. 36).

[GL06] Jean-Loup Guillaume and Matthieu Latapy. “Bipartite Graphs as Models of Complex Networks”. In: *Physica A: Statistical Mechanics and its Applications* 371.2 (2006), pp. 795–813. DOI: 10.1016/j.physa.2006.04.047 (cit. on p. 43).

[GPD23a] Simon Guillot, Thibault Prouteau, and Nicolas Dugue. “De l’interprétabilité des dimensions à l’interprétabilité du vecteur : parcimonie et stabilité”. In: *CORIA-TALN 2023*. ATALA, 2023, pp. 83–91. URL: https://aclanthology.org/2023.jeptalnrecital-international.10 (cit. on p. 57).

[GPD23b] Simon Guillot, Thibault Prouteau, and Nicolas Dugué. “Sparser is better: one step closer to word embedding interpretability”. In: *International Conference of Computational Semantics 2023 (IWCS)*. Vol. IWCS’23. 2023, pp. 106–115. URL: https://hal.science/hal-04321407 (cit. on pp. 57, 146).

[GA05] Roger Guimera and Luís A. Nunes Amaral. “Cartography of complex networks: modules and universal roles”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.02 (2005), P02001. DOI: 10.1088/1742-5468/2005/02/P02001 (cit. on p. 52).

[HYL17] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *NIPS’17* 30 (2017). DOI: 10.48550/arXiv.1706.02216 (cit. on p. 36).

[HLJ16] William L Hamilton, Jure Leskovec, and Dan Jurafsky. “Cultural shift or linguistic drift? comparing two computational measures of semantic change”. In: *EMNLP*. 2016, p. 2116. DOI: 10.18653/v1/D16-1229 (cit. on p. 123).

[Har15] Bernard E. Harcourt. “Risk as a Proxy for Race: The Dangers of Risk Assessment”. In: *Federal Sentencing Reporter* 27.4 (2015), pp. 237–243. DOI: 10.1525/fsr.2015.27.4.237. eprint: https://online.ucpress.edu/fsr/article-pdf/27/4/237/135063/fsr_2015_27_4_237.pdf (cit. on p. xxi).

[Har54] Zellig S. Harris. “Distributional Structure”. In: *WORD* 10.2-3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520 (cit. on p. 27).

[HZ11] Mohammad Al Hasan and Mohammed J. Zaki. “A Survey of Link Prediction in Social Networks”. In: *Social Network Data Analytics*. Springer US, 2011, pp. 243–275. DOI: 10.1007/978-1-4419-8462-3_9 (cit. on p. 73).

[Hom50] George C. Homans. *The human group*. Harcourt, Brace, 1950 (cit. on p. 45).

[Hop87] Paul Hopper. “Emergent Grammar”. In: *Annual Meeting of the Berkeley Linguistics Society* 13 (1987), p. 139. DOI: 10.3765/bls.v13i0.1834 (cit. on p. 8).

[Hua+12] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. “Improving word representations via global context and multiple word prototypes”. In: *ACL'12*. 2012. URL: https://aclanthology.org/P12-1092 (cit. on p. 100).

[HLC05] Zan Huang, Xin Li, and Hsinchun Chen. “Link prediction approach to collaborative filtering”. In: *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*. ACM, 2005, pp. 141–142. DOI: 10.1145/1065385.1065415 (cit. on p. 72).

[Huf52] David A Huffman. “A Method for the Construction of Minimum-Redundancy Codes”. In: *Proceedings of the I.R.E.* (1952). DOI: 10.1109/JRPROC.1952.273898 (cit. on p. 47).

[ISO20] ISO Central Secretary. *Software and systems engineering – Software testing – Part 1: 11: Guidelines on the testing of AI-based systems*. Software testing ISO/IEC TR 29119-11:2020. International Organization for Standardization, 2020. URL: https://www.iso.org/standard/62711.html (cit. on pp. 114, 115).

[Jac01] Paul Jaccard. “Étude comparative de la distribution florale dans une portion des Alpes et du Jura”. In: *Bulletin de la Société Vaudoise des Sciences Naturelles* (1901), pp. 547–579. DOI: 10.5169/SEALS-266450 (cit. on p. 74).

[JW02] Glen Jeh and Jennifer Widom. “SimRank: A Measure of Structural-Context Similarity”. In: *SIGKDD'02*. KDD '02. Association for Computing Machinery, 2002, pp. 538–543. DOI: 10.1145/775047.775126 (cit. on pp. 69, 75).

[Kar72] Richard M. Karp. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*. Springer US, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9 (cit. on p. 44).

[Kat53] Leo Katz. “A new status index derived from sociometric analysis”. In: *Psychometrika* 18.1 (1953), pp. 39–43. DOI: 10.1007/BF02289026 (cit. on p. 75).

[Kel94] Rudi Keller. *On language change: The invisible hand in language*. Psychology Press, 1994 (cit. on p. 9).

[Kre02] Valdis E Krebs. “Mapping networks of terrorist cells”. In: *Connections* 24.3 (2002), pp. 43–52. URL: https://api.semanticscholar.org/CorpusID:16168595 (cit. on p. 73).

[Kum+20] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. “Link prediction techniques, applications, and performance: A survey”. In: *Physica A: Statistical Mechanics and its Applications* 553 (2020), p. 124289. DOI: 10.1016/j.physa.2020.124289 (cit. on pp. 46, 73, 76, 83).

[Kun13] Jérôme Kunegis. “KONECT – The Koblenz Network Collection”. In: *Proceedings of the International Conference on World Wide Web Companion*. 2013, pp. 1343–1350. URL: http://dl.acm.org/citation.cfm?id=2488173 (cit. on pp. 17, 18).

[KLB09] Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. “The Slashdot Zoo: Mining a Social Network with Negative Edges”. In: *WWW'09*. 2009, pp. 741–750. DOI: 10.1145/1526709.1526809 (cit. on p. 76).

[Kuo+13] Tsung-Ting Kuo, Rui Yan, Yu-Yang Huang, Perng-Hwa Kung, and Shou-De Lin. “Unsupervised link prediction using aggregative statistics on heterogeneous social networks”. In: *SIGKDD'13*. ACM, 2013, pp. 775–783. DOI: 10.1145/2487575.2487614 (cit. on p. 76).

[LLW13] James Ladyman, James Lambert, and Karoline Wiesner. “What is a complex system?” In: *European Journal for Philosophy of Science* 3.1 (2013), pp. 33–67. DOI: 10.1007/s13194-012-0056-8 (cit. on p. 7).

[LFR08] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. “Benchmark graphs for testing community detection algorithms”. In: *Physical review E* 78.4 (2008), p. 046110. DOI: 10.1103/PhysRevE.78.046110 (cit. on p. 80).

[Lan+10] Andrea Lancichinetti, Mikko Kivelä, Jari Saramäki, and Santo Fortunato. “Characterizing the Community Structure of Complex Networks”. In: *PLoS ONE* 5.8 (2010), pp. 1–8. DOI: 10.1371/journal.pone.0011976 (cit. on p. 52).

[Lan+11] Andrea Lancichinetti, Filippo Radicchi, Jose’ Javier Ramasco, and Santo Fortunato. “Finding statistically significant communities in networks”. In: *PLoS ONE* 6.4 (2011), e18961. DOI: 10.1371/journal.pone.0018961 (cit. on p. 46).

[LGI21] Loïc Lannelongue, Jason Grealey, and Michael Inouye. “Green Algorithms: Quantifying the Carbon Footprint of Computation”. In: *Advanced Science* 8.12 (2021), p. 2100707. DOI: 10.1002/advs.202100707 (cit. on p. 38).

[LNB14] Jey Han Lau, David Newman, and Timothy Baldwin. “Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality”. In: *EACL’14*. Association for Computational Linguistics, 2014, pp. 530–539. DOI: 10.3115/v1/E14-1056 (cit. on p. 123).

[LC14] Conrad Lee and Pádraig Cunningham. “Community detection: effective evaluation on large social networks”. In: *Journal of Complex Networks* 2.1 (2014), pp. 19–37. DOI: 10.1093/comnet/cnt012 (cit. on p. 89).

[LG14] Omer Levy and Yoav Goldberg. “Neural Word Embedding as Implicit Matrix Factorization”. In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf (cit. on pp. 32, 50).

[LGD15] Omer Levy, Yoav Goldberg, and Ido Dagan. “Improving Distributional Similarity with Lessons Learned from Word Embeddings”. In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 211–225. DOI: 10.1162/tacl_a_00134 (cit. on pp. 30, 50).

[Lib05] David Liben-Nowell. “An Algorithmic Approach to Social Networks”. PhD Thesis. Massachusetts Institute of Technology, 2005. DOI: 1721.1/33861 (cit. on pp. 75, 77).

[LK03] David Liben-Nowell and Jon Kleinberg. “The Link Prediction Problem for Social Networks”. In: *CIKM*. CIKM ’03. New Orleans, LA, USA: Association for Computing Machinery, 2003, pp. 556–559. DOI: 10.1145/956863.956972 (cit. on pp. 72, 73).

[LK07] David Liben-Nowell and Jon Kleinberg. “The link-prediction problem for social networks”. In: *Journal of the American Society for Information Science and Technology* 58.7 (2007), pp. 1019–1031. DOI: 10.1002/asi.20591 (cit. on pp. 71, 73).

[LLC10] Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. “New Perspectives and Methods in Link Prediction”. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’10. Association for Computing Machinery, 2010, pp. 243–252. DOI: 10.1145/1835804.1835837 (cit. on p. 75).

[LL10] Weiping Liu and Linyuan Lu. “Link Prediction Based on Local Random Walk”. In: *Europhysics Letters* 89.5 (2010), p. 58007. DOI: 10.1209/0295-5075/89/58007 (cit. on p. 75).

[Liu+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692 (cit. on p. 33).

[LZ11] Linyuan Lu and Tao Zhou. "Link Prediction in Complex Networks: A Survey". In: *Physica A: Statistical Mechanics and its Applications* 390.6 (2011), pp. 1150–1170. DOI: 10.1016/j.physa.2010.11.027 (cit. on pp. 72, 73).

[LJZ09] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. "Similarity index based on local paths for link prediction of complex networks". In: *Physical Review E* 80.4 (2009), p. 046122. DOI: 10.1103/PhysRevE.80.046122 (cit. on pp. 75, 78).

[LVL23] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. "Estimating the carbon footprint of bloom, a 176b parameter language model". In: *Journal of Machine Learning Research* 24.253 (2023), pp. 1–15. DOI: 10.48550/arXiv.2211.02001 (cit. on p. xx).

[Lut22] Quentin Lutz. "Graph-based contributions to machine-learning". PhD thesis. Institut polytechnique de Paris, 2022 (cit. on p. 55).

[Mak+21] Ilya Makarov, Dmitrii Kiselev, Nikita Nikitinsky, and Lovro Subelj. "Survey on graph embeddings and their applications to machine learning problems on graphs". In: *PeerJ Computer Science* 7 (2021), e357. DOI: 10.7717/peerj-cs.357 (cit. on pp. 76, 83).

[Mar+20] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benot Sagot. "CamemBERT: a Tasty French Language Model". In: *ACL'20*. Association for Computational Linguistics, 2020, pp. 7203–7219. DOI: 10.18653/v1/2020.acl-main.645 (cit. on p. 33).

[MBC17] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. "A Survey of Link Prediction in Complex Networks". In: *ACM Computing Surveys* 49.4 (2017), pp. 1–33. DOI: 10.1145/3012704 (cit. on pp. 73, 75).

[Mat+05] Irina Matveeva, Gina Anne Levow, Ayman Farahat, and Christiaan Royer. "Term representation with generalized latent semantic analysis". In: *RANLP'05* 2005-Janua (2005), pp. 308–315. DOI: 10.1075/cilt.292.08mat (cit. on p. 30).

[McR+05] Ken McRae, George S. Cree, Mark S. Seidenberg, and Chris Mcnorgan. "Semantic feature production norms for a large set of living and nonliving things". In: *Behavior Research Methods* 37.4 (2005), p. 547. DOI: 10.3758/BF03192726 (cit. on p. 126).

[Mer88] Robert K. Merton. "The Matthew Effect in Science, II: Cumulative Advantage and the Symbolism of Intellectual Property". In: *Isis* 79.4 (1988), pp. 606–623. DOI: 10.1086/354848 (cit. on p. 17).

[Mik+13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space". In: *ICLR'13*. 2013. DOI: 10.48550/arXiv.1301.3781 (cit. on pp. 31, 102).

[Mil56] George A. Miller. "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information". In: *The Psychological Review* 63.2 (1956), pp. 81–97. URL: http://www.musanim.com/miller1956/ (cit. on p. 126).

[Mil57] George A. Miller. "Some Effects of Intermittent Silence". In: *The American Journal of Psychology* 70.2 (1957), p. 311. DOI: 10.2307/1419346 (cit. on p. 9).

[Mil95] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (1995), pp. 39–41. DOI: 10.1145/219717.219748 (cit. on p. 100).

[Mit09] Melanie Mitchell. *Complexity: a guided tour*. Oxford University Press, 2009 (cit. on p. 9).

[Mor92] Edgar Morin. “From the concept of system to the paradigm of complexity”. In: *Journal of Social and Evolutionary Systems* 15.4 (1992), pp. 371–385. DOI: https://doi.org/10.1016/1061-7361(92)90024-8 (cit. on p. 7).

[Mos21] Daniel A. Moses. “Deep learning applied to automatic disease detection using chest X-rays”. In: *Journal of Medical Imaging and Radiation Oncology* 65.5 (2021), pp. 498–517. DOI: https://doi.org/10.1111/1754-9485.13273. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1754-9485.13273 (cit. on p. xx).

[MTM12] Brian Murphy, Partha Pratim Talukdar, and Tom Mitchell. “Learning effective and interpretable semantic models using non-negative sparse embedding”. In: *COLING’12: Technical Papers*. 2012. URL: https://aclanthology.org/C12-1118.pdf (cit. on pp. 118–120, 125, 126, 129).

[Nan11] Keith Suderman Nancy Ide Randi Reppen. *The Open ANC (OANC)*. 2011. URL: https://hdl.handle.net/11403/sldr000770/v2 (cit. on p. 104).

[Név+22] Aurélie Névéol, Yoann Dupont, Julien Bezançon, and Karën Fort. “French CrowS-Pairs: Extending a challenge dataset for measuring social bias in masked language models to a language other than English”. In: *ACL’22 (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022, pp. 8521–8531. DOI: 10.18653/v1/2022.acl-long.583 (cit. on p. xxi).

[New11] M. E. J. Newman. “Resource Letter CS1: Complex Systems”. In: *American Journal of Physics* 79.8 (2011), pp. 800–810. DOI: 10.1119/1.3590372 (cit. on pp. 6, 7).

[New13] M. E. J. Newman. “Spectral methods for network community detection and graph partitioning”. In: *Physical Review E* 88.4 (2013), p. 042822. DOI: 10.1103/PhysRevE.88.042822 (cit. on p. 46).

[NG04] M. E. J. Newman and M. Girvan. “Finding and evaluating community structure in networks”. In: *Physical Review E* 69.2 (2004), p. 026113. DOI: 10.1103/PhysRevE.69.026113 (cit. on pp. 46, 48).

[New06] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the National Academy of Sciences* 103.23 (2006), pp. 8577–8582. DOI: 10.1073/pnas.0601602103 (cit. on p. 21).

[Obe+19] Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. “Dissecting racial bias in an algorithm used to manage the health of populations”. In: *Science* 366.6464 (2019), pp. 447–453. DOI: 10.1126/science.aax2342 (cit. on p. xxi).

[OSR19] Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. “Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures”. In: Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019. Leibniz-Institut für Deutsche Sprache, 2019, pp. 9–16. DOI: 10.14618/ids-pub-9021 (cit. on p. 33).

[OST57] Charles E. Osgood, George J. Suci, and Percy H. Tannenbaum. *The measurement of meaning*. The measurement of meaning. Univer. Illinois Press, 1957 (cit. on p. 98).

[Ou+16] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. “Asymmetric Transitivity Preserving Graph Embedding”. In: *SIGKDD’16*. ACM, 2016, pp. 1105–1114. DOI: 10.1145/2939672.2939751 (cit. on p. 83).

[Pal+05] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. “Uncovering the overlapping community structure of complex networks in nature and society”. In: *Nature* 435.7043 (2005), pp. 814–818. DOI: 10.1038/nature03607 (cit. on p. 49).

[Pat+21] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. "Carbon Emissions and Large Neural Network Training". In: (2021), p. 22. DOI: 10.48550/arXiv.2104.10350 (cit. on p. 38).

[Pav23] Ellie Pavlick. "Symbols and grounding in large language models". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 381.2251 (2023), p. 20220041. DOI: 10.1098/rsta.2022.0041. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2022.0041 (cit. on p. 27).

[PLC17] Leto Peel, Daniel B Larremore, and Aaron Clauset. "The ground truth about metadata and community detection in networks". In: *Science advances* 3.5 (2017), e1602548. DOI: 10.1126/sciadv.1602548 (cit. on p. 89).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global vectors for word representation". In: *EMNLP'14*. 2014. DOI: 10.3115/v1/d14-1162 (cit. on p. 30).

[Per+17] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. "Don't Walk, Skip! Online Learning of Multi-scale Network Embeddings". In: *ASONAM*. 2017, pp. 258–265. DOI: 10.1145/3110025.3110086 (cit. on pp. 36, 66).

[PAS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: Online Learning of Social Representations". In: *SIGKDD'14* (2014), pp. 701–710. DOI: 10.1145/2623330.2623732 (cit. on pp. 35, 66, 71, 81).

[Pet+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations". In: *NAACL'18*. Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202 (cit. on p. 33).

[PP59] Lloyd Peterson and Margaret Jean Peterson. "Short-Term Retention of Individual Verbal Items". In: *Journal of Experimental Psychology* 58.3 (1959), p. 193. DOI: 10.1037/h0049234 (cit. on p. 126).

[Pia+23] Simone Piaggesi, Megha Khosla, André Panisson, and Avishek Anand. *DINE: Dimensional Interpretability of Node Embeddings*. 2023. DOI: 10.48550/arXiv.2310.01162 (cit. on p. 119).

[Pie20] Bénédicte Pierrejean. "Qualitative Evaluation of Word Embeddings: Investigating the Instability in Neural-Based Models". PhD thesis. Université Toulouse 2 - Jean Jaurès, 2020 (cit. on pp. 123, 124).

[PL05] Pascal Pons and Matthieu Latapy. "Computing Communities in Large Networks Using Random Walks". In: *Computer and Information Sciences - ISCIS*. Springer Berlin Heidelberg, 2005, pp. 284–293. DOI: 10.1007/11569596_31 (cit. on p. 47).

[Pri76] Derek De Solla Price. "A general theory of bibliometric and other cumulative advantage processes". In: *Journal of the American Society for Information Science* 27.5 (1976), pp. 292–306. DOI: 10.1002/asi.4630270505 (cit. on pp. 17, 74).

[Pro+21] Thibault Prouteau, Victor Connes, Nicolas Dugué, Anthony Perez, Jean-Charles Lamirel, Nathalie Camelin, and Sylvain Meignier. "SINr: Fast Computing of Sparse Interpretable Node Representations is not a Sin!" In: *IDA'21*. Lecture Notes in Computer Science. Springer International Publishing, 2021, pp. 325–337. DOI: 10.1007/978-3-030-74251-5_26 (cit. on pp. 52, 53, 55, 57).

[PDG24] Thibault Prouteau, Nicolas Dugue, and Simon Guillot. "From Communities to Interpretable Network and Word Embedding: a Unified Approach". In: *Journal of Complex Networks (submitted, major revision)* (2024) (cit. on pp. 54, 55, 57).

[Pro+22] Thibault Prouteau, Nicolas Dugué, Nathalie Camelin, and Sylvain Meignier. "Are Embedding Spaces Interpretable? Results of an Intrusion Detection Evaluation on a Large French Corpus". In: *LREC'22*. European Language Resources Association, 2022, pp. 4414–4419. URL: https://aclanthology.org/2022.lrec-1.469 (cit. on pp. 57, 120, 129).

[Pro+23] Thibault Prouteau, Nicolas Dugué, Simon Guillot, and Anthony Perez. "SINr: a python package to train interpretable word and graph embeddings". In: *FRCCS'23*. 2023, p. 215. DOI: 10.5281/zenodo.7957531 (cit. on p. 57).

[QBK06] Yanjun Qi, Ziv Bar-Joseph, and Judith Klein-Seetharaman. "Evaluation of different biological data and computational classification methods for use in protein interaction prediction". In: *Proteins: Structure, Function, and Bioinformatics* 63.3 (2006), pp. 490–500. DOI: 10.1002/prot.20865 (cit. on p. 72).

[RAK07] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. "Near linear time algorithm to detect community structures in large-scale networks". In: *Physical Review E* (2007). DOI: 10.1103/PhysRevE.76.036106 (cit. on pp. 47, 52).

[Raj+22] Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol. "AI in health and medicine". In: *Nature medicine* 28.1 (2022), pp. 31–38. DOI: 10.1038/s41591-021-01614-0 (cit. on p. xxi).

[RB06] Jörg Reichardt and Stefan Bornholdt. "Statistical mechanics of community detection". In: *Phys. Rev. E* 74 (1 2006), p. 016110. DOI: 10.1103/PhysRevE.74.016110 (cit. on p. 49).

[Ric27] Stuart A. Rice. "The Identification of Blocs in Small Political Bodies". In: *American Political Science Review* 21.3 (1927), pp. 619–627. DOI: 10.2307/1945514 (cit. on p. 45).

[RKR20] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. "A Primer in BERTology: What We Know About How BERT Works". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 842–866. DOI: 10.1162/tacl_a_00349 (cit. on pp. 38, 116).

[Roz+19] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. "GEMSEC: Graph Embedding with Self Clustering". In: *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*. ACM. 2019, pp. 65–72. DOI: 10.1145/3341161.3342890 (cit. on p. 66).

[RKS20] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. "Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs". In: *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. ACM. 2020, pp. 3125–3132. DOI: 10.1145/3340531.3412757 (cit. on pp. 58, 67).

[RG65] Herbert Rubenstein and John B. Goodenough. "Contextual correlates of synonymy". In: *Communications of the ACM* 8.10 (1965), pp. 627–633. DOI: 10.1145/365628.365657 (cit. on p. 98).

[Rud19] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature machine intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x (cit. on pp. 114–117).

[SG17] Fatemeh Salehi Rizi and Michael Granitzer. "Properties of Vector Embeddings in Social Networks". In: *Algorithms* 10.4 (2017), p. 109. DOI: 10.3390/a10040109 (cit. on pp. 71, 79, 80, 85, 87).

[Sam+23] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. *From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference*. 2023. URL: http://arxiv.org/abs/2310.03003 (cit. on pp. 34, 38).

[Sca+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605 (cit. on p. xx).

[Sch92] H. Schütze. "Dimensions of meaning". In: *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*. Supercomputing '92. IEEE Computer Society Press, 1992, pp. 787–796. DOI: 10.1109/SUPERC.1992.236684 (cit. on p. 29).

[SUF00] Benno Schwikowski, Peter Uetz, and Stanley Fields. "A network of proteinprotein interactions in yeast". In: *Nature Biotechnology* 18.12 (2000), pp. 1257–1261. DOI: 10.1038/82360 (cit. on p. 72).

[SPS20] Or Sharir, Barak Peleg, and Yoav Shoham. "The Cost of Training NLP Models: A Concise Overview". In: *CoRR* abs/2004.08900 (2020). arXiv: 2004.08900. URL: https://arxiv.org/abs/2004.08900 (cit. on p. 34).

[SMF18] Jamin Shin, Andrea Madotto, and Pascale Fung. "Interpreting word embeddings with eigenvector analysis". In: *NIPS'18, IRASL workshop*. 2018, pp. 73–81. URL: https://openreview.net/forum?id=rJfJiR5ooX (cit. on pp. 38, 116).

[Sim55] Herbert A. Simon. "On a Class of Skew Distribution Functions". In: *Biometrika* 42.3-4 (1955), pp. 425–440. DOI: 10.1093/biomet/42.3-4.425 (cit. on p. 17).

[Sim62] Herbert A. Simon. "The Architecture of Complexity". In: *Proceedings of the American Philosophical Society* 106.6 (1962), pp. 467–482. URL: http://www.jstor.org/stable/985254 (cit. on p. 8).

[SS91] John Sinclair and Les Sinclair. *Corpus, Concordance, Collocation*. Oxford University Press, 1991. 204 pp. (cit. on pp. 26, 27).

[SCV19] Aakash Sinha, Rémy Cazabet, and Rémi Vaudaine. "Systematic Biases in Link Prediction: Comparing Heuristic and Graph Embedding Based Methods". In: *Complex Networks and Their Applications VII*. Vol. 812. Springer International Publishing, 2019, pp. 81–93. DOI: 10.1007/978-3-030-05411-3_7 (cit. on pp. 34, 52, 66, 74, 84).

[Sta+18] Natalie Stanley, Thomas Bonacci, Roland Kwitt, Marc Niethammer, and Peter J. Mucha. *Stochastic Block Models with Multiple Continuous Attributes*. 2018. URL: http://arxiv.org/abs/1803.02726 (cit. on p. 76).

[SGM19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP". In: arXiv, 2019. URL: http://arxiv.org/abs/1906.02243 (cit. on pp. xx, 34, 38).

[Su+19] Zhendong Su, Kaijun Ren, Ruoyun Zhang, and Suo-Yi Tan. "Disrupting Terrorist Networks Based on Link Prediction: A Case Study of the 911 Hijackers Network". In: *IEEE Access* 7 (2019), pp. 61689–61696. DOI: 10.1109/ACCESS.2019.2915938 (cit. on p. 73).

[Sub+18] Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. "SPINE: SParse Interpretable Neural Embeddings". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17433 (cit. on pp. 101, 102, 119, 120, 122, 129).

[Sun+16] Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. "Sparse word embeddings using l1 regularized online learning". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16. New York, New York, USA: AAAI Press, 2016, pp. 2915–2921. DOI: 10.5555/3060832.3061029 (cit. on pp. 123, 137).

[Tan+21] Aditya Tandon, Aiiad Albeshri, Vijey Thayananthan, Wadee Alhalabi, Filippo Radicchi, and Santo Fortunato. "Community detection in networks using graph embeddings". In: *Physical Review E* 103.2 (2021), p. 022316. DOI: 10.1103/PhysRevE.103.022316 (cit. on pp. 71, 80).

[Tan+15] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. "LINE: Large-scale Information Network Embedding". In: *WWW*. 2015, pp. 1067–1077. DOI: 10.1145/2736277.2741093 (cit. on p. 66).

[Tas+03] Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. "Link prediction in relational data". In: *Advances in neural information processing systems* 16 (2003). URL: https://proceedings.neurips.cc/paper_files/paper/2003/file/1e0a84051e6a4a7381473328f43c4884-Paper.pdf (cit. on p. 73).

[TSL00] Joshua B Tenenbaum, Vin de Silva, and John C Langford. "A global geometric framework for nonlinear dimensionality reduction". In: *science* 290.5500 (2000), pp. 2319–2323. DOI: 10.1126/science.290.5500.2319 (cit. on p. 34).

[TGH17] Julien Tissier, Christophe Gravier, and Amaury Habrard. "Dict2vec: Learning word embeddings using lexical dictionaries". In: *EMNLP'17*. 2017, pp. 254–263. DOI: 10.18653/v1/D17-1024 (cit. on p. 31).

[Tka+20] Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. *Label Studio: Data labeling software*. 2020. URL: https://github.com/heartexlabs/label-studio (cit. on p. 121).

[TFP06] Hanghang Tong, Christos Faloutsos, and Jia-yu Pan. "Fast Random Walk with Restart and Its Applications". In: *ICDM'06)*. IEEE, 2006, pp. 613–622. DOI: 10.1109/ICDM.2006.70 (cit. on p. 75).

[TM69] Jeffrey Travers and Stanley Milgram. "An Experimental Study of the Small World Problem". In: *Sociometry* 32.4 (1969), pp. 425–443. URL: http://www.jstor.org/stable/2786545 (cit. on p. 13).

[Tsi+18] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. "VERSE: Versatile Graph Embeddings from Similarity Measures". In: *WWW'18* (2018), pp. 539–548. DOI: 10.1145/3178876.3186120 (cit. on p. 69).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, ukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *NIPS'17*. Vol. 2017-December. Neural information processing systems foundation, 2017, pp. 5999–6009. URL: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (cit. on p. 33).

[Wan+20] Changping Wang, Chaokun Wang, Zheng Wang, Xiaojun Ye, and Philip S. Yu. "Edge2vec: Edge-based Social Network Embedding". In: *ACM Transactions on Knowledge Discovery in Data* 14.4 (2020). DOI: 10.1145/3391298 (cit. on p. 66).

[WSP07] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. "Local Probabilistic Models for Link Prediction". In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 322–331. DOI: 10.1109/ICDM.2007.108 (cit. on p. 76).

[WCZ16] Daixin Wang, Peng Cui, and Wenwu Zhu. "Structural Deep Network Embedding". In: *SIGKDD'16*. 2016, pp. 1225–1234. DOI: 10.1145/2939672.2939753 (cit. on p. 66).

[Wan+19] Hanchen Wang, Nina Grgic-Hlaca, Preethi Lahoti, Krishna P. Gummadi, and Adrian Weller. "An Empirical Study on Learning Fairness Metrics for COMPAS Data with Human Supervision". In: NIPS'19 (2019). URL: http://arxiv.org/abs/1910.10255 (cit. on p. 39).

[Wan+15] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. "Link prediction in social networks: the state-of-the-art". In: *Science China Information Sciences* 58.1 (2015), pp. 1–38. DOI: 10.1007/s11432-014-5237-y (cit. on p. 76).

[Wan+17] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. "Community Preserving Network Embedding". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 31.1 (2017). DOI: 10.1609/aaai.v31i1.10488 (cit. on p. 66).

[WS98] Duncan J Watts and Steven H Strogatz. “Collective dynamics of small-world networks”. In: 393 (1998), p. 3. DOI: 10.1038/30918 (cit. on pp. 13–15).

[WJ55] Robert S. Weiss and Eugene Jacobson. “A Method for the Analysis of the Structure of Complex Organizations”. In: *American Sociological Review* 20.6 (1955), p. 661. DOI: 10.2307/2088670 (cit. on p. 45).

[Wil16] Sinead A. Williamson. “Nonparametric Network Models for Link Prediction”. In: *Journal of Machine Learning Research* 17.202 (2016), pp. 1–21. URL: http://jmlr.org/papers/v17/16-032.html (cit. on p. 76).

[XC08] Jennifer Xu and Hsinchun Chen. “The topology of dark networks”. In: *Communications of the ACM* 51.10 (2008), pp. 58–65. DOI: 10.1145/1400181.1400198 (cit. on p. 73).

[Yan+18] Xiaoran Yan, Lucas G. S. Jeub, Alessandro Flammini, Filippo Radicchi, and Santo Fortunato. “Weight thresholding on complex networks”. In: *Physical Review E* 98.4 (2018), p. 042304. DOI: 10.1103/PhysRevE.98.042304 (cit. on pp. 50, 51).

[Zha+17] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. “Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints”. In: *EMNLP’17*. Association for Computational Linguistics, 2017, pp. 2979–2989. DOI: 10.18653/v1/D17-1323 (cit. on p. xxi).

[ZLZ09] Tao Zhou, Linyuan Lu, and Yi-Cheng Zhang. “Predicting Missing Links via Local Information”. In: *The European Physical Journal B* 71.4 (2009), pp. 623–630. DOI: 10.1140/EPJB/E2009-00335-8 (cit. on p. 74).

[Zip35] G. K. Zipf. *The psycho-biology of language*. Houghton, Mifflin, 1935 (cit. on pp. 9, 17, 18).

[Zip49] George Kingsley Zipf. *Human behavior and the principle of least effort*. Addison-Wesley Press, 1949 (cit. on p. 17).

COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Le Mans Université

Title: Graphs, Words, and Communities: Converging Paths to Interpretability with a Frugal Embedding Framework

Keywords: graph embedding, word embedding, interpretability, frugality, complex networks

Abstract: Representation learning with word and graph embedding models allows distributed representations of information that can in turn be used in input of machine learning algorithms. Through the last two decades, the tasks of embedding graphs nodes and words have shifted from matrix factorization approaches that could be trained in a matter of minutes to large models requiring ever larger quantities of training data and sometimes weeks on large hardware architectures. However, in a context of global warming where sustainability is a critical concern, we ought to look back to previous approaches and consider their performances with regard to resources consumption. Furthermore, with the growing involvement of embeddings in sensitive machine learning applications (judiciary system, health), the need for more interpretable and explainable representations has manifested. To foster efficient representation learning and interpretability, this thesis introduces Lower Dimension Bipartite Graph Framework (`LDBGF`), a node embedding framework able to embed with the same pipeline graph data and text from large corpora represented as co-occurrence networks. Within this framework, we introduce two implementations (`SINr-NR`, `SINr-MF`) that leverage community detection in networks to uncover a latent embedding space where items (nodes/-words) are represented according to their links to communities. We show that `SINr-NR` and `SINr-MF` can compete with similar embedding approaches on tasks such as predicting missing links in networks (link prediction) or node features (degree centrality, PageRank score). Regarding word embeddings, we show that `SINr-NR` is a good contender to represent words via word co-occurrence networks. Finally, we demonstrate the interpretability of `SINr-NR` on multiple aspects. First with a human evaluation that shows that `SINr-NR` s dimensions are to some extent interpretable. Secondly, by investigating sparsity of vectors, and how having fewer dimensions may allow interpreting how the dimensions combine and allow sense to emerge.

Titre : Graphes, mots et communautés : chemins convergents pour l'interprétabilité des représentations par une approche de plongements frugale

Mot clés : plongements de graphes, plongements de mots, interprétabilité, frugalité, réseaux complexes

Résumé : L'apprentissage de représentations au travers des méthodes de plongements de mots (*word embedding*) et de graphes (*graph embedding*) permet des représentations distribuées de l'information. Ces représentations peuvent à leur tour être utilisées en entrée d'algorithmes d'apprentissage automatique. Au cours des deux dernières décennies, les tâches de plongement de nœuds et de mots sont passées d'approches par factorisation matricielle qui pouvaient être réalisées en quelques minutes à de grands modèles nécessitant des quantités toujours plus importantes de données dapprentissage et parfois des semaines sur de grandes architectures matérielles. Toutefois, dans un contexte de réchauffement climatique où la durabilité est une préoccupation essentielle, il peut être souhaitable de revenir à des méthodes plus frugales comme elles pouvaient l'être par le passé. En outre, avec l'implication croissante des plongements dans des applications sensibles de l'apprentissage automatique (système judiciaire, santé), le besoin de représentations plus interprétables et explicables s'est manifesté. Pour favoriser l'apprentissage de représentations efficaces et l'interprétabilité, cette thèse présente *Lower Dimension Bipartite Graph Framework* (`LDBGF`), un framework de plongements de nœuds capable d'extraire une représentation vectorielle avec le même pipeline de traitement, à condition que les données proviennent d'un graphe ou de texte issu de grands corpus représentés sous forme de réseaux de cooccurrence. Dans ce cadre, nous présentons deux implémentations (`SINr-NR`, `SINr-MF`) qui tirent parti de la détection des communautés dans les réseaux pour découvrir un espace latent dans lequel les éléments (nœuds/mots) sont représentés en fonction de leurs liens avec les communautés. Nous montrons que `SINr-NR` et `SINr-MF` peuvent rivaliser avec des approches de plongements concurrentes sur des tâches telles que la prédiction des liens manquants dans les réseaux (*link prediction*) ou certaines caractéristiques des nœuds (centralité de degré, score *PageRank*). En ce qui concerne les plongements de mots, nous montrons que `SINr-NR` est un bon candidat pour représenter les mots en utilisant les réseaux de cooccurrences de mots. Enfin, nous démontrons l'interprétabilité de `SINr-NR` sur plusieurs aspects. Tout d'abord, une évaluation humaine montre que les dimensions de `SINr-NR` sont dans une certaine mesure interprétables. Ensuite, nous étudions la parcimonie des vecteurs. Notamment, combien un nombre réduit de dimensions peut permettre d'interpréter comment ces dernières se combinent et permettent de dégager un sens.

Chapitre 1 : Systèmes et réseaux complexes

Les systèmes complexes sont présents dans beaucoup d'aspects de nos vies quotidiennes. Nous sommes nous-mêmes des systèmes biologiques complexes composés de cellules, tissus et protéines. Nous évoluons également dans des systèmes sociaux. Lorsque nous utilisons les transports en commun, ou lorsque nous participons à une réunion en visioconférence, nous interagissons également avec et au travers de systèmes complexes. Leur omniprésence dans notre quotidien en fait une source inépuisable pour explorer et mieux comprendre comment ils se forment, se développent et fonctionnent. Par leur multiplicité, ces systèmes sont étudiés du point de vue et avec les méthodes de divers domaines tels que la physique, l'écologie, les

neurosciences ou ici l'informatique. En ce qui concerne les travaux présentés dans ce manuscrit, nous allons nous concentrer sur le sous-domaine et les structures de réseaux complexes dans lesquels interagissent des objets pour donner une organisation à plus grande échelle. Ce chapitre a pour objectifs de faire un tour d'horizon des systèmes complexes et des diverses méthodes d'étude de ces derniers. Nous présentons alors leurs principales caractéristiques avant de porter notre attention sur les réseaux complexes. Nous définirons leurs principales caractéristiques et comment ils reflètent celles énoncées pour les systèmes complexes.

Chapitre 2 : Du réseau au vecteur : apprentissage de représentations

Exploiter la richesse des connaissances offertes par les réseaux et leur topologie est complexe sans faire appel à des méthodes tierces pour la représentation de leur contenu. Ces méthodes, communément appelées méthodes de plongements (*embeddings*) permettent d'obtenir des représentations compactes pouvant être utilisées en entrée d'algorithmes d'apprentissage automatique. Les plongements de graphes ont pour objectif de fournir ces représentations sous forme de vecteurs. Cependant, les méthodes de plongements ne sont pas restreintes aux réseaux, elles sont légion lorsqu'il s'agit de représenter des données, notamment du texte au travers de ce que l'on appelle plongements de mots. Les plongements de graphes et de mots partagent un objectif commun : fournir une représentation compacte des données tout en préservant les similarités et dissimilarités entre ces données dans l'espace d'origine. Puisque cet objectif est commun aux méthodes de plongements, ces dernières reposent aussi sur une hypothèse commune, l'hypothèse distributionnelle. Aussi, les méthodes employées présentent de grandes similitudes qui témoignent d'une parenté entre plongements de graphes et de mots que nous allons mettre en lumière dans ce chapitre. Néanmoins, les méthodes de plongements ne sont pas sans défauts, notamment en ce qui concerne l'interprétabilité des représentations et leur complexité de calcul. La course vers des modèles de plongements toujours plus grands résulte en une croissance du besoin en ressources de calcul et de stockage ainsi qu'en une complexification de leur apprentissage. De plus, le manque d'interprétabilité associé aux modèles "boîte noire" signifie qu'il est difficile d'inspecter leur structure et de comprendre comment la représentation des données est formée. Afin de répondre à ces problèmes, nous avons développé une méthode de plongements capable de dériver des représentations interprétables, à la fois à partir de réseaux et de textes, en peu de temps et avec peu de ressources de calcul. Nous introduisons le cadre théorique `LDBGF` utilisant les cliques d'un réseau pour découvrir une projection bipartie permettant de compresser le graphe. Nous décrivons ensuite deux implémentations inspirées par `LDBGF`: `SINr-NR`, qui repose sur la détection de communautés, et `SINr-MF` qui effectue une factorisation de la matrice d'adjacence d'un réseau.

Chapitre 3 : Représenter les réseaux : plongements de graphes

Les plongements de graphes ont pour objectif de capturer la topologie d'un réseau et de la résumer dans une représentation vectorielle compacte. Les méthodes permettant d'obtenir des plongements de graphes s'appuient sur différentes techniques, pour capturer, aussi précisément que possible cette topologie. Cependant, ces méthodes ont parfois pour inconvénient de nécessiter de longs temps d'apprentissage lorsqu'elles sont confrontées à de grands réseaux. De plus, toutes les méthodes ne capturent pas l'architecture des réseaux avec le même succès, certains ont de bonnes performances sur des tâches bien spécifiques. Ce chapitre à pour but d'évaluer les capacités de `SINr-NR` et `SINr-MF` à produire des représentations pertinentes. Leurs performances sont com-

parées à celles d'autres algorithmes permettant d'obtenir des plongements de graphes, et ce sur des tâches variées. Nos expériences attraient à évaluer la qualité des plongements de graphes produits au travers de: la *prédiction de liens*, la prédiction de caractéristiques des noeuds et du graphe (degré, coefficient de clustering, *PageRank*) et à former des groupes de noeuds à partir de leurs représentations. Nous montrons que `SINr-NR` et `SINr-MF` sont polyvalents et qu'ils peuvent fournir des plongements de graphes de qualité. `SINr-NR` a l'avantage de fournir ces représentations avec un temps de calcul réduit.

Chapitre 4 : Réprésenter les textes : plongements de mots

Les plongements de mots ont pour objectif de capturer la complexité du langage et fournir des représentations pertinentes du sens des mots sous forme de vecteurs. Dans la décennie passée, de nombreuses nouvelles méthodes ont été développées pour apprendre des plongements de mots. Cependant, certaines limites liées à ces méthodes subsistent. Leur développement s'est concentré sur l'obtention de modèles performants. Cependant, dans le contexte de sobriété énergétique, imposé par le réchauffement climatique, le temps d'exécution est un aspect des méthodes d'apprentissage automatique qu'il est nécessaire de prendre en compte. Or, les grands modèles de langage (LLM) développés ces dernières années mobilisent de grandes masses de données qui impliquent de longs temps d'apprentissage. Par ailleurs,la plupart des méthodes d'apprentissage de plongements de mots fournissent des représentations opaques peu auditables par les humains. Or, ce manque de transparence des modèles est un problème lorsque ceux-ci sont utilisés pour des applications sensibles. En effet, les plongements de mots sont souvent la première brique des chaînes de traitement en NLP. Ce chapitre évalue `SINr-NR` au regard des limites des modèles précédents et notamment en termes de temps d'exécution et de performance. Pour ce faire, `SINr-NR` est évalué aux côtés d'approches concurrentes. Nous mesurons notamment le temps d'exécution des méthodes et leurs capacités à fournir une représentation fidèle des mots en préservant leur similarité dans l'espace de plongements. Nous montrons qu'en plus d'être un modèle performant pour produire des plongements de graphes, `SINr-NR` est adapté pour l'apprentissage de plongements de mots.

Chapitre 5 : Interpréter les représentations

L'explicabilité et l'interprétabilité sont des sujets qui prennent de l'ampleur dans les domaines de l'apprentissage automatique. Rendre les décisions explicables et les systèmes interprétables peut être utile pour prévenir les biais potentiels présents dans les systèmes et augmenter la confiance dans ces derniers. Des modèles interprétables ont été développés pour l'apprentissage de plongements de mots. Leur avantage principal est de ne pas recourir à d'autres informations que celles présentes dans le modèle pour interpréter les représentations. Ce chapitre s'intéresse à l'interprétabilité de tels modèles et spécifiquement à l'interprétabilité de `SINr-NR`. Nous montrons que `SINr-NR` est au niveau des méthodes de plongements de mots interprétables à l'état de l'art. Dans un second temps, nous étendons la définition d'interprétabilité pour les plongements de mots. Cela ouvre de nouvelles perspectives pour les modèles interprétables et pour comprendre comment l'information s'y organise. Enfin, nous montrons, au travers de plusieurs visualisations, l'interprétabilité des modèles et donnons un aperçu de leur structure interne.