FROM RECOMMENDER SYSTEMS TO SPATIO-TEMPORAL DYNAMICS WITH NETWORK SCIENCE

THÈSE N^{O} 7428 (2017)

PRÉSENTÉE LE 31 JANVIER 2017

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE TRAITEMENT DES SIGNAUX 2
PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Kirell Maël BENZI

acceptée sur proposition du jury:

Prof. P. Frossard, président du jury
Prof. P. Vandergheynst, directeur de thèse
Dr S. Anvar, rapporteur
Dr M. Vlachos, rapporteur
Dr O. Verscheure, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2017

Le hasard n'existe pas. Rien ne se produit sans cause.
— Ostad Elahi

Abstract

Networks are data structures that are fundamental for capturing and analyzing complex interactions between objects. While they have been used for decades to solve problems in virtually all scientific fields, their usage for data analysis in real-world practical applications deserves to be further investigated.

In this thesis, we explore multiple aspects of network science and show how the design of new graph-based approaches offers an unprecedented depth for analyzing complex datasets. Through the study of practical applications, we demonstrate how to extract key findings in several domains such as digital humanities, recommender systems, social behavior, neuroscience or knowledge discovery.

First, we propose to define in a concise manner the data science workflow. We present the tools, techniques, and questions that the practitioner needs to have in mind when addressing a new large-scale problem as they are of tremendous importance if one wants to apply network science concepts to real applications.

Based on this foundation chapter, we begin by demonstrating the worth of networks for music recommendation with Genezik, our smart playlist application that adapts to user taste. Using signal processing, machine learning, and graphs, we show how to improve the performance of recommender systems as well as proposing a radically different user experience that has yet to be found in competing systems.

We then move on to the introduction of the causal multilayer graph of activity, a novel graph method dedicated to the analysis of dynamical processes over networks. More than a data structure, we present a data analysis approach that tracks spreading or propagation of events through time in a scalable manner by efficiently combining a network with values associated with its vertices. Used in four different applications, the analysis of spatio-temporal patterns of activity extracted from the causal multilayer graph helps us better understand how rumors spread in social networks or how brain regions interact in resting states for instance.

Finally, we study the browsing behavior of millions of people on Wikipedia and show how to extract contextual patterns of activity that reflect what is collectively remembered from past events. Based on their analysis, we confirm social studies on human behavior and conclude by revealing some of the rules that define human curiosity.

Keywords: network science, data mining, data analytics, knowledge discovery.

Résumé

Les réseaux sont des structures de données fondamentales pour capturer et analyser les interactions complexes entre objets. Bien qu'ils aient été utilisés pendant des décennies dans la majorité des domaines scientifiques, leurs usages pour l'analyse de données dans des applications pratiques méritent d'être étudiés plus en détails.

Dans cette thèse, nous explorons les limites de la science des réseaux et montrons comment le design de nouvelles méthodes de graphes offre une profondeur sans précédent pour analyser des jeux de données complexes. À travers l'étude d'applications pratiques, nous démontrons comment extraire des découvertes clés dans plusieurs domaines comme les humanités digitales, les systèmes de recommandation, l'analyse du comportement, les neurosciences ou encore exploration de connaissances.

En premier lieu, nous proposons de définir d'une manière concise le flux de travaux pour la science des données. Nous présentons les outils, techniques et les questions que le praticien doit garder à l'esprit quand il s'attaque à un nouveau problème à grande échelle car ils sont vitaux si l'on souhaite appliquer les concepts de la science des réseaux à de vraies applications.

En se basant sur ce chapitre de fondation, nous commençons par démontrer l'utilité des réseaux pour la recommandation de musique avec Genezik, notre application de liste de lectures intelligentes qui s'adapte au goût de l'utilisateur. En utilisant le traitement de signal, l'apprentissage machine et les graphes, nous montrons comment améliorer les performances des systèmes de recommandations tout en proposant une expérience utilisateur radicalement différente de ce qui existe dans les systèmes concurrents.

Par la suite, nous introduisons le graphe multicouche causal d'activité, une nouvelle méthode de graphe dédiée à l'analyse des processus dynamiques sur les réseaux. Plus qu'une structure de données, nous présentons une approche d'analyse de données qui traque la propagation des événements dans le temps d'une manière évolutive en combinant efficacement un réseau et les valeurs associées à ses nœuds. Utilisé dans quatre applications différentes, l'analyse des motifs spatio-temporels d'activité extraits du graphe multicouche causal nous aide à mieux comprendre comment les rumeurs se propagent dans les réseaux sociaux ou comment les régions du cerveau interagissent au repos par exemple.

Enfin, nous étudions le comportement de millions de visiteurs sur Wikipedia et montrons comment extraire des motifs d'activité contextuels qui reflètent ce que nous nous remémorons collectivement des événements passés. En se basant sur leurs analyses, nous confirmons les

études sociales sur le comportement et concluons en révélant quelques règles qui façonnent la curiosité humaine.

Mots clefs : science des réseaux, exploration de données, extraction de connaissances

Acknowledgements

When I first entered the LTS2 at EPFL, I didn't know what a Ph.D. was. At the time, it didn't really matter because I was here only for a semester project and firmly believed that I would never venture in academia. Now, at the end of my doctoral journey, I would like to thank all the people that influenced and helped me along the way.

First and foremost, I would like to express my gratitude towards my advisor Prof. Pierre Vandergheynst for supporting me in this endeavor. In addition to offering me the possibility to start this thesis, he also offered me exceptional working conditions. I am grateful for the freedom he gave me as well as his support for my extra academic activities notably my art exhibitions.

The second key person I would like to thank is Alexandre Alahi. It was him who first invited me to visit Switzerland for this initial semester project. As a true friend, he offered much more than supervision. His generosity will always be remembered.
I also would like to especially thank Benjamin Ricaud, aka GBR, my office mate and friend in the "Bureau des Kings". Benjamin was always there in the good days but more importantly in the bad ones. His energy and positive disposition truly helped me when I struggled with my research. He was also there to review this manuscript. We made a great team!

To my close friends in Lausanne, I would like to say "Thank you" for all the time we spent together: to Florian Carrere for his help in building Genezik, to Sasan Yazdani for all our crazy gaming sessions, to Christina Boydev for all our discussions.

I thank all the former and current members of the LTS2 for being great colleagues: Andreas, Xavier, Emmanuel, Rodrigo, Michael, Nauman, Johan. To my trip buddies, Vassilis Kalofolias and Nathanael Perraudin, be sure that our time in Shanghai will be remembered!

Finally, I want to thank my mother, father and my family for their invaluable love and support. To those who believed in me, supported and encouraged me throughout the years to become what I am now, I am eternally grateful. May the Force be with you, always.

Lausanne, 20 December 2016 Kirell Benzi

Contents

Abstract (English/Français) i

Acknowledgements v

List of figures xi

1 Introduction 1
- 1.1 From data to models . . . 1
- 1.2 Motivational example . . . 2
- 1.3 Thesis structure and contributions . . . 4

2 Network science workflow 7
- 2.1 Solving a real-world issue . . . 7
- 2.2 Mining the world . . . 8
  - 2.2.1 Choosing the right tools . . . 8
  - 2.2.2 Finding data into the wild . . . 9
  - 2.2.3 Mining the Star Wars expanded universe . . . 10
- 2.3 Storing data . . . 10
- 2.4 Processing graphs . . . 11
- 2.5 Exploratory data analysis . . . 12
  - 2.5.1 The graph of characters . . . 12
  - 2.5.2 Inference of missing values . . . 13
  - 2.5.3 Network visualization . . . 16
- 2.6 Publishing results . . . 17
- 2.7 Discussion . . . 18

3 Graph-based music recommendation 21
- 3.1 A brief overview of Music Information Retrieval . . . 21
- 3.2 Music recommender systems . . . 22
- 3.3 Song recommendation with non-negative matrix factorization and graph total variation . . . 23
  - 3.3.1 Problem formulation . . . 24
  - 3.3.2 Graph regularization with total variation . . . 25
  - 3.3.3 Primal-dual optimization . . . 26

3.3.4 Recommending songs . . . 27
3.3.5 Graphs of playlists and songs . . . 28
3.3.6 Experimental results . . . 33
3.4 Genezik . . . 36
3.4.1 Genezik architecture . . . 37
3.4.2 Personalized graph of songs . . . 39
3.4.3 Playlist recommendation . . . 45
3.5 Improving models and reproducibility in MIR . . . 49
3.5.1 Introducing the Free Music Archive dataset . . . 50
3.6 Discussion . . . 52

4 Exploring dynamical processes over networks 55
4.1 Tracking dynamic activity patterns . . . 57
4.1.1 The causal multilayer graph . . . 58
4.1.2 Definition . . . 59
4.1.3 Causal multilayer graph of activity . . . 61
4.2 Analyzing dynamic activity patterns . . . 63
4.2.1 Dynamic activated components . . . 63
4.2.2 Clustering the components . . . 64
4.2.3 Analysis of the cluster properties and average activation component . . . 65
4.3 Applications . . . 66
4.3.1 Visualizing crowd movements in a train station . . . 66
4.3.2 Analyzing thousands of collaborative audio playlists . . . 68
4.3.3 Dynamic activation patterns in brain MRI data . . . 74
4.3.4 Rumor spreading on Twitter and dynamic activity of communities . . . 80
4.4 Efficient construction of the causal multilayer graph of activity . . . 86
4.4.1 Implementation details . . . 88
4.4.2 The generalized causal multilayer graph of activity . . . 90
4.5 Discussion . . . 91

5 Social behavior analysis with network science 95
5.1 Structuring social activity on Wikipedia . . . 97
5.1.1 Extracting dynamic activation components . . . 97
5.1.2 Crafting Wiki-souvenirs . . . 98
5.1.3 Deep recollection . . . 100
5.2 Characterization of human curiosity on Wikipedia . . . 101
5.2.1 Media pressure bounds the space of curiosity . . . 102
5.2.2 Curiosity as a mix between lack of information and recreational roaming 102
5.2.3 Gossip and human relations drive curiosity . . . 102
5.3 Emergence of a new kind of collectively-structured network . . . 105
5.4 Discussion . . . 109

6 Discussion **111**
6.1 The causal multilayer graph of activity 111
6.2 Building real-world applications 114
6.3 Diffusing science to the general public 115

Bibliography **135**

List of contributions **136**

Curriculum Vitae **139**

List of Figures

1.1 Data science pipeline in research 2
1.2 The Strength of Collaboration - 2016. Copyright Kirell Benzi 6

2.1 Distribution of the principal species in the Star Wars expanded universe 13
2.2 Total degree of the most connected characters in the Star Wars expanded universe 14
2.3 Character distribution timeline 15
2.4 Visualization of the label propagation algorithm 16
2.5 The Dark Side and the Light - 2015. Copyright Kirell Benzi 19

3.1 Our global song recommender system model 24
3.2 Part of the adjacency matrix of the playlist graph 30
3.3 Features combination to build the song graph 32
3.4 Part of the adjacency matrix of the song graph 33
3.5 MPR for each playlist category on the test set. Our models use the same parameters of Table 3.4. Ambiguous categories such as Rock, Punk have the highest MPR on the test set. Our model outperforms significantly the others methods on those specific categories. 36
3.6 General architecture of the Genezik 38
3.7 Principal node and edge features used in Genezik. 42
3.8 Genezik GUI 44
3.9 Lead Me On, the default playlist recommendation mode in Genezik 47
3.10 When the Music is Good - 2013. Copyright Kirell Benzi. 54

4.1 Evolution of a congestion pattern in a localized area of a city represented as a graph. 56
4.2 An illustration of the causal multilayer graph. 60
4.3 From signal to binary states. 61
4.4 Construction of the causal multilayer graph of activity H. 62
4.5 Creation of static and dynamic feature vectors. 64
4.6 Segmentation of the west corridor into regions of interests according to the density of pedestrians. 67
4.7 Two examples of average activated components plotted on the same figure. . . 69
4.8 Average activated components representing music moods. 72
4.9 The spatial spread of the different average activated components. 73

4.10 Node distribution among the 7 RSNs for each cluster. 77
4.11 Average activation component of cluster 11, corresponding to the visual RSN. . 79
4.12 Average activation component of cluster 4, the somato-motor RSN 80
4.13 Examples of raw dynamic activation components in the brain. 81
4.14 Graph of the largest activated component from the 10-minute sampling, colored by communities. 85
4.15 Zoom in a region of interest from the Higgs CMG of activity. 86
4.16 Actual implementation of the CMG of activity. 88
4.17 Creation of causal edges from a triplet (A, e, B). 89
4.18 Creation of causal edges in the generalized CMG of activity. 92
4.19 On time - 2014. Copyright Kirell Benzi. 93

5.1 Examples of Wiki-souvenirs extracted from collective activity on Wikipedia. . . 99
5.2 Example of deep recollection for the Ebola Wiki-souvenir. 101
5.3 Germanwings crash Wiki-souvenir and its timeline. 103
5.4 Topic repartition for Wiki-souvenirs. 104
5.5 Wiki-memory and random-memory at the hour scale colored by main topic. . 107
5.6 Evolution of Wiki-memory and random-memory at the hour scale through time. 108
5.7 GUI wireframe of our contextual search engine. 110

6.1 Evolution of the Pokemon video graph. Copyright Kirell Benzi. 114
6.2 Secret Knowledge - 2016. Copyright Kirell Benzi. 116

1 Introduction

Understanding the rules and properties of our universe has always been the goal of science. In this never-ending quest for knowledge, scientists have progressively improved their techniques to observe, explain and predict real-world phenomena.

One of the fundamental tools to address the inherent complexity of our surroundings is the graph. A graph or network is a mathematical object that allows to model relations as links between entities called nodes. Originally rooted in physics, networks can be found in all domains of natural sciences whether to simulate interactions between galaxies or model the propagation of diseases for example. In computer science, graphs are fundamental data structures onto which all modern software that runs the today's world is built. Used in the latest advances in artificial intelligence, networks can power recommender systems as well as face recognition algorithms.

In fact, networks have become so important in the past fifteen years that a new academic field of research named *Network science* is now entirely devoted to their analysis [1, 2]. Drawing on methods of graph theory from mathematics, the study of causal systems from physics, inferential modeling from statistics, social behavior analysis from sociology or data-mining and visualization techniques from computer science, network science is a rich transverse area of research.

1.1 From data to models

The rise of network science is also closely coupled with the data deluge of the past few years. To illustrate this fact, let us take a look at the number of Internet users. In 2015, the number of connected people went beyond 3 billions [3] and every year, the increase of digital information doubles. By 2020, around 40 trillion gigabytes of storage will be required to store these data [4].

This era of "big-data" is transforming society as well as the scientific community. In the case of network science, the data collection process was often considered as a second-class citizen and was only brought up to support theories with strong assumptions on the topology or the

properties of the network. In this thesis, we underline the crucial importance of collecting, storing, retrieving and processing data in network analysis. This approach, from data to models, is progressively maturing from the original buzzword "data science" to a real and well-defined methodology [5, 6, 7] starting from a real-world problem (with associated data) to result in the construction of a particular model as illustrated in Fig.1.1.

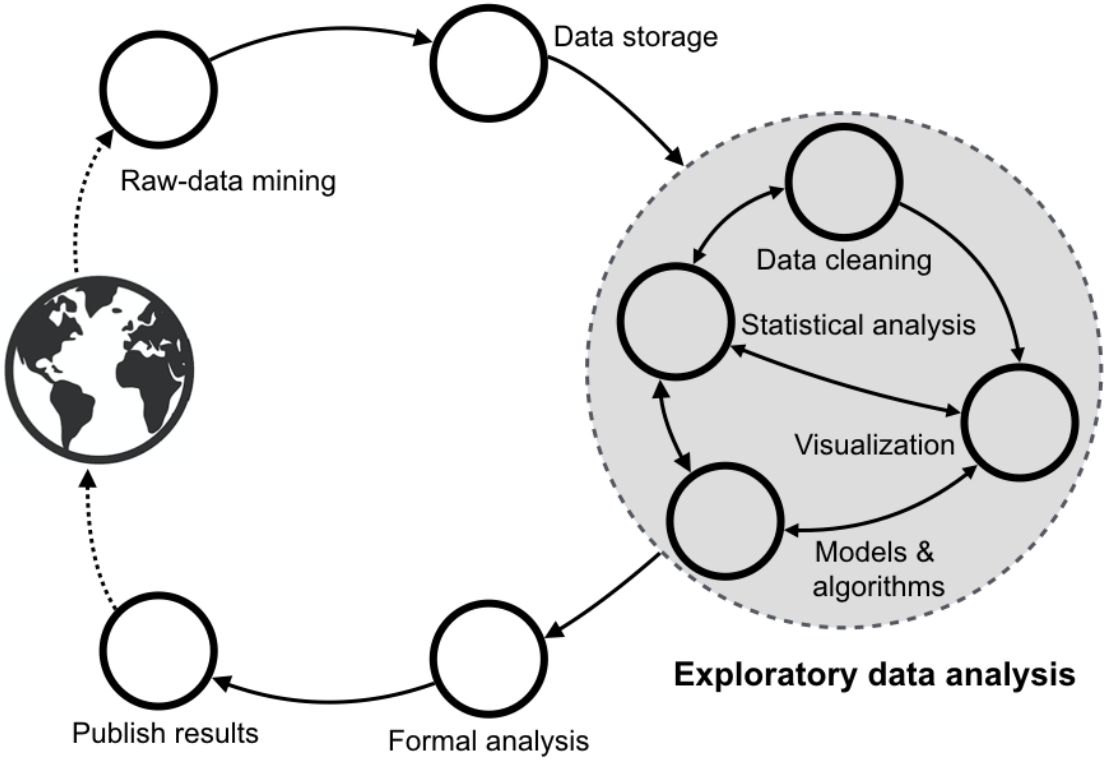


Figure 1.1 – **Data science pipeline for research.** Instead of starting with a general theory and finding applications in the real-world (top-down), the data science approach starts from the data and progressively finds a particular model suited for the problem at hand (bottom-up). The exploratory data analysis helps to tune and refine both algorithms and visualizations to eventually perform the formal analysis. World map design: FreePik.

1.2 Motivational example

To give the reader a better understanding of the data science process and its application to network analysis, let us focus on a simple but real-world example. In 1998, the École Polytechnique Fédérale de Lausanne (EPFL) launched an institutional archive called Infoscience. This database is used to store and display all scientific publications (articles, conference papers, proceedings, books, posters, etc.) as well as patents produced by the laboratories. It could be interesting to characterize how research has evolved for the last 20 years and visualize how researchers interact together. Far from a purely academic standpoint, those insights could be used by the administration of EPFL to improve collaboration in the university and ultimately

strengthen the position of EPFL on a global scale.

The first stage of our pipeline 1.1 is to extract raw data from the public web interface. Because there is no direct way to get the complete (already normalized) database, we have to query the EPFL service in an automated manner and parse the archive records. Note that our description is intentionally left without too many details as it is the object of chapter 2 and could distract the reader from getting the main picture. Because the size of the whole database is significant (around 123,000 records in 2016), it is necessary to think about how to optimally store and retrieve the records for the exploratory data analysis phase that comes next. As we start playing with the data and observe how the records are structured, we notice that parts of the data are ill-formed and need to be filtered out (statistical analysis and data cleaning phase).

Now that our database is consolidated and cleaned, we can start thinking on how to reveal the collaboration between researchers. In other words, what algorithms or models could best fit the problem at hand. We know that a database record at least indicates the name of the work, the list of co-authors and the published date. A network of co-publications could reveal how researchers have interacted together through time.

In this undirected network $G = (V_G, E_G, W_G)$, authors (scientist, professor, Ph.D. student, etc.) are the nodes (vertices) of the graph with $|V_G| = N$ the total number of authors. The set of links $E_G = \{w_{ij} \,|\, i, j \in V_G,\, w_{ij} \in W_G\}$, also called edges, connect authors with a given weight w_{ij} representing how many papers two researchers have co-published.

To quickly gain some intuition on the structure of the network, we often rely on its visualization in a two or three-dimensional space. Here, the choice of the layout algorithm and the tuning of its parameters is a necessary step to address the complexity. Fortunately, some tools have been developed to facilitate this process and obtain both informative and aesthetically pleasing results as shown in Fig. 1.2.

Shaping the data in the form of a network opens many perspectives as we can draw on more than fifty years of graph theory and computer science algorithms on networks to make the data talk. For instance, it is now possible to find who is the most central point of the network or which are the communities of researchers. Is there any collaboration between laboratories or does everyone only co-publish with colleagues of the same group? Is the network connected? If so, it would mean that there is a path from all researchers in any field of research to all others researchers. All these questions and hypotheses once answered pave the way for a new batch of interrogations and ultimately broaden our understanding of how the academic world works, from data to science.

Despite their power, existing methods are sometimes too limited to model underlying phenomena, calling for the development of new graph-based methods. For instance, if we focus on time series that we can associate with vertices of a graph, most of the works adopt an exclusive approach: either study the values or analyze the graph. In the following chapters, we will demonstrate, *au contraire*, the added value of combining signal and graph to uncover

previously hidden interactions in the data.

1.3 Thesis structure and contributions

In this thesis, we explore each aspect of network science by developing models, methods, and visualizations for practical applications. Starting with the concrete implementation of an audio playlist recommender system, we move on to the study of dynamical processes over networks under the prism of our data-driven multilayer graph model [8]. Additionally, one of the side goals of this manuscript is to advocate for the acknowledgement of dataset creation: potential discoveries from these datasets can outweigh by far the pain of gathering data. The thesis is structured as follows.

Chapter 2: We formalize the data science workflow for network analysis. This chapter forms a concise data-mining guide for anyone willing to learn how to extract, store, process and visualize data to reproduce this work and go beyond it. It also serves as a foundation chapter introducing all the techniques that are used later on in the manuscript to create and analyze graphs. We illustrate each step by focusing on the extraction and analysis of one of the largest fictional universes ever created: the Star Wars expanded universe dataset. More than just an example, we show how a real historical corpus could benefit from the inference of its missing values by combining networks and machine learning.

Chapter 3: We develop the inner workings of Genezik, a hybrid audio playlist recommendation application [9] based on graphs. Using both content-based features as well as collaborative filtering, the Genezik model builds on implicit and explicit user feedback to adapt to the global taste of users as well as delivering extremely personal recommendations. We show how its unique architecture combining personalized user graph with a state-of-the-art song recommendation engine [10], offers a finer and richer user experience than the competition.

Chapter 4: Building on the possible improvements of our recommender system, we introduce a novel data-driven multilayer graph model [8]. Specifically designed to track activity patterns in dynamical processes over large networks, it handles billions of data points on a single commodity server. We further apply this model to several real-world examples to underline its merits. First, we use the causal multilayer graph of activity to analyze thousands of collaborative audio playlists as a means to answer some of the challenges brought by media recommendation. Second, we apply it to track crowd movements in a train station. By analyzing recurring patterns of people walking in the train station corridors, we show how our method can help predict congestion and take appropriate measures to improve the pedestrian flow. Third, we investigate the activity of the main anatomical regions of the brain in resting state recordings. Here, our approach recovers and confirms the existence of resting state networks, revealing how brain regions interact through time. Finally, we study how rumors spread by performing a multi-scale analysis of Twitter feeds around the discovery of the Higgs boson. We recover the results from the literature and find evidence that communities of users appear in a dynamical manner.

Chapter 5: Building on the social implications of the analysis of Twitter, we further refine our causal multilayer graph model to extract browsing patterns on Wikipedia over a period of seven months. With the combination of a semantic database, we show how to craft "Wiki-souvenirs" i.e. rich, contextualized and timestamped patterns of activity representative of events happening in the world in real-time. The analysis of these Wiki-souvenirs helps us characterize human curiosity in a significant manner as we process billions of visits on pages of the online encyclopedia. We conclude this chapter by witnessing the emergence of a new kind of collectively-structured network based on Wiki-souvenirs with deep sociological implications about how humans search, organize and share information.

We complete this thesis in chapter 6 with a summary of our contributions and the possible evolutions of the proposed models. We also dedicate a section to advocating for the diffusion of science in the general public. We show how art and science intertwine to achieve this goal.

We summarize our main contributions in the following.

Main contributions:

- a concise data science guide for network analysis introducing a standardized methodology as well as tools and techniques for practitioners;
- a consumer-grade implementation and distribution of two graphs-based audio recommender systems advancing the state-of-the-art in both performance and user experience [9, 10];
- the gathering of a new large-scale legal dataset of raw audio files and features solving copyright issues for the development of applications in the field of Music Information Retrieval (MIR);
- a novel multilayer graph model that encodes both the activity and the structure of a network in a scalable manner [8];
- the collection of four different datasets and their analysis using our multilayer graph model with the notable introduction of data visualizations that allow the apprehension of the dynamical activity on the network [8, 11];
- a detailed large-scale spatio-temporal analysis of Wikipedia revealing fundamental mechanisms behind human curiosity;
- the introduction of a new class of networks, collectively structured by the activity of users on the web;
- the diffusion of scientific methodologies in a pedagogic manner using art to raise awareness and interest for research in general [12].

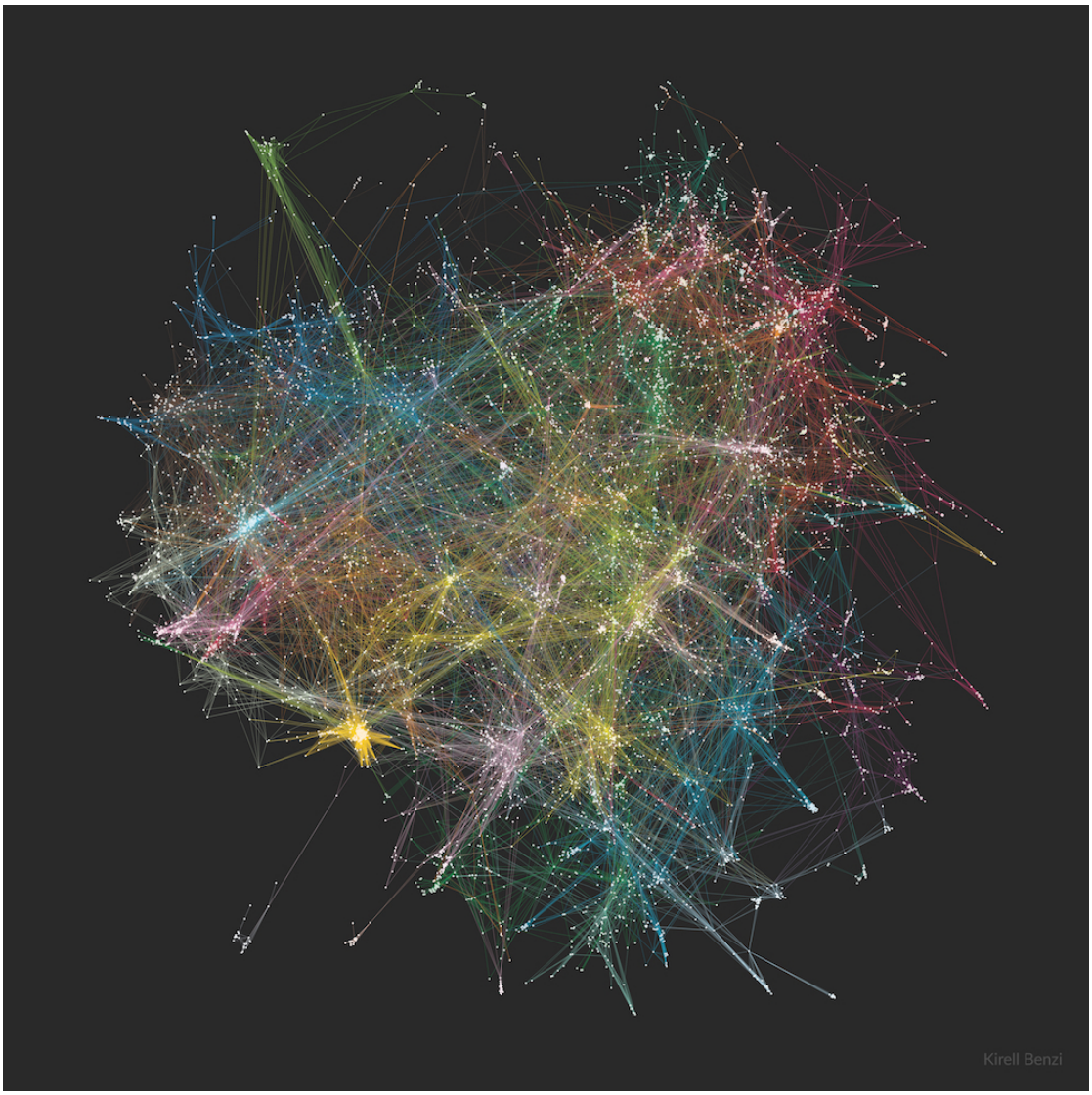


Figure 1.2 – **The Strength of Collaboration - 2016**. This network depicts the scientific collaboration at EPFL. Based on Infoscience, the online archive of publications on the campus, authors are connected if they have co-published a paper. Started in 1998, Infoscience now holds more than 123,000 records regrouping conference articles, scientific journals, and even patents. On the graph, authors from the same area of research constitute distinct colored communities. Indeed, it is more natural to publish with colleagues in your team than with people from completely different fields. Nevertheless, the network is well connected, showing that the exchange of ideas among fellow members of the university and across domains is widespread.

2 Network science workflow

Building on the motivational example of the introduction, we here propose a practical guide to tackling the many challenges of data science for network analysis by showing some of the questions the practitioner needs to have in mind and give some answers on how to solve them. More generally, our approach is in the continuity of the open science movement, pushing science forward by showing how to ease the reproducibility of results, enforcing good code practices and encouraging the distribution of datasets [13].

Here, we thus briefly describe some of the tools and techniques to mine and distribute real-world datasets. We also demonstrate "in-vivo" the strength of networks by analyzing the characters of a large fictional corpus of documents: the Star Wars expanded universe.

2.1 Solving a real-world issue

Star Wars is indubitably the most popular franchise in the Western world. Created in 1977 by George Lucas, the epic space opera films have progressively expanded to unique sets of, books, video games, comic books, short-stories and merchandising with more than 90,546 unique items cataloged in 2013 [14].

From an academic point of view, a network analysis of a fictional universe is not unprecedented [15, 16, 17] as it brings new insights on the relationships between characters or their affiliation to a community. More generally, the usage of science and computing open unique perspectives that help apprehend humanities from a digitalized and automated perspective [18, 19, 20]. This revolution is now impacting all the academic fields related to the study of human culture such as literature, history, philosophy, art or musicology [21, 22, 23].

Unfortunately, one of the key limitation in this field of digital humanities, apart from resistance to change [24], is the access to digitalized information. Indeed, the availability of ancient manuscripts is scarce, and their digitalization is a problem that has yet to be solved as it draws on many different scientific fields such as physics [25], chemistry [26] as well as pattern recognition [27] and automated text analysis [28]. The advantages of studying an online and

regularly updated set of documents thus alleviate concerns about the digitalization and help us focus on the actual analysis of the characters, presented in 2.5.

2.2 Mining the world

Now that the context is set, we can start with the first step of our data science process 1.1 and explain how to extract raw data from the world. The interested reader can dive further into the subject by studying the book of Moens *et al.* [29].

2.2.1 Choosing the right tools

Doing data science is a synonym of spending an incredible amount of time extracting and cleaning the data. The actual science part only comes at the end when the data is put in the right format for the models or algorithms one needs to apply. The toolchain is thus paramount as it influences the viability of the realization of the practitioner's idea.

The strength of a programming language is often defined by the number of well-maintained libraries around it. In the field of data science, the Python programming language [30] is a clear winner [5, 6, 31] as it offers a very rich ecosystem of tools, libraries or tutorials to get started while being general-purpose and high-level. Its readability and direct interface to the C programming language [32] combines both the advantages of being easily understandable by others without sacrificing the performance of well-optimized numerical libraries such as Numpy, Scipy, Scikit-learn or Pandas [33, 34, 35, 31].

Even though the readability of code can be enforced by a programming language like Python, all ultimately depends on the skills of the programmer. This point is critical as many scientists often disregard programming as a mean to achieve an idea neglecting the basics of coding and core reuse. We argue here that programming should not be treated as a second-class citizen and that one should be able to reproduce the original experiments precisely to push science forward.

Fortunately, with the advent of technology, an astonishing quantity of material can be found to hone one's skills in programming, algorithms and software engineering. A good starter could consists of taking several Massive Open Online Courses (MOOC) [36] on Coursera[1] complemented with books on design patterns [37], software engineering [38] and functional programming [39].

Often neglected because of the effort needed to set it correctly, code documentation is also an important part of reproducibility. In an ideal world, every nontrivial function should be documented. However, in practice, a good compromise can be made by documenting nontrivial parts of the code and providing walk-through examples to reproduce the work. Note

[1] https://www.coursera.org/

that because bad documentation is much worse than no documentation at all, one should be careful while refactoring parts of the code as the project grows.

Combining all these aspects, researchers and software developers of the Juypter project[2] developed the Jupyter Notebook [33, 40]. Born from the lack of reproducibility, the now-standard notebook is an interactive "collaborative, shareable, publishable, and reproducible" document that contains code, documentation, and images related to a given work. Designed around a read-eval-print loop (REPL), its introduction is driving the adoption of Python in the scientific community by revolutionizing the data science pipeline. Mastering the notebook thus appears as a must for practitioners in the field of data science and research in general.

Finally, we wish to underline the importance of code versioning for the scientific methodology as a whole. More than just a backup of our code, code versioning is the first step to achieving reproducibility and collaboration with others. Basic proficiency in tools like Git [41] or SVN are mandatory to let others contribute, reuse and possibly modify the code of our experiments.

2.2.2 Finding data into the wild

The preferred way to access data is to use the Application Programming Interface (API)[3] of a web service if it exists. An API is a standardized [42] and documented interface for anyone willing to access the data from a program. The queried data is often returned in the form of JSON data [43], a structured text format, easily readable with any programming language. Another advantage of using APIs is to leverage the community of programmers around them. Most of the time, the data provider itself release a driver for a mainstream programming language (such as Python) to ease the pain of writing a well-designed and evolutive library for the service. Because the time to access data is always greatly diminished when using APIs, one should always seek or ask if they exist before using any other method. It also encourages a fair-use of the bandwidth and enforces the limit rates or legal conditions imposed to users by the service. Fortunately, most of the data providers have special plans for academics that allow accessing the data freely.

If no API exists or if the desired data is not directly exposed, it is necessary to "scrape" web pages to gather the information we seek. The process of scraping consists of reading the source of a web page in the HTML format and extract from the markup tags the desired information. Once again, one should always try to leverage existing libraries [44] to perform this task and focus on the matter at hand instead of re-inventing a suboptimal piece of code to do so. Another advantage of using a dedicated library is to perform automatic retrial on errors, throttling web requests or luring web servers into thinking that our program acts like a human being. With experience, we have noticed that simpler and naive approaches of web scraping without the implementation of these core principles fail, stopping the data analysis before it even started.

[2] http://jupyter.org/

[3] https://en.wikipedia.org/wiki/Application_programming_interface

2.2.3 Mining the Star Wars expanded universe

To gather the content of Star Wars EU, we need to find a reliable source of information containing the maximum of information about the franchise and its individuals. Fortunately, the for-profit company behind the online encyclopedia Wikipedia hosts a dedicated Wiki[4] about Star Wars EU containing more than 130,135 pages. While an API exists to get the content of the Wiki in a structured manner, there is no direct access to only retrieve the biography of all the characters of the universe.

The proposed solution thus relies on the usage of a web scraper (a bot) that fetches all the categories indexing the characters such as *Category:Individuals*. First, from this root category, we create a network of sub-categories by recursively fetching all sub-categories. Note that this network of categories is formally defined as a forest [45] (a disconnected set of trees) because the loose structure of the Wiki created by fans does not systematically index all the individuals under the main root category. Then, for each node of the category network, our robot retrieves the associated biography web page.

2.3 Storing data

Mining the web or any services is very similar to mining ore from a gold mine: once we extract nuggets from the earth we need to consider what are the best solutions to store, protect and refine it. In our case, there are three storage strategies:

Keep all in memory: While it saves the hassle of actually managing data, it is very impractical as it requires to request the service each time we add a modification to our model or algorithm. It is also not robust in case of power failure or memory corruption.

Flat files: Saving text data to flat files is usually done when the content is completely unstructured, such as log files or when the data is so big that requires a distributed file system such as HDFS [46]. One can also directly append a row to a CSV (comma separated values) file, but it has some disadvantages. First, it is computationally heavy and very unpractical to update an existing row of a large file in case the data changes. Data must only be appended to this end of the file leading to duplicated values if the program runs again with the same file. More importantly, adding columns to a CSV is problematic has we need to update all previous rows to account for the newly added column. We should thus prefer the CSV file format when exporting structured data and not used it as primary storage.

Another solution is to store data in large binary files using a standard data format such as HDF5 [47]. Mostly employed in the scientific community it is a viable solution for storage and exchange of data. However, it still lacks the flexibility and durability of a database notably for concurrent access, replication, and security.

[4] http://starwars.wikia.com/

Databases: Using a database to store information probably is the best solution when fetching data from an API as it provides a standardized interface to store and retrieve the data while letting to others the burden of managing the data (a research field in itself). Choosing the right database for a particular problem is an important decision pondering on multiple factors such as the size of the dataset, the shape of the data, the number of concurrent users, the existing infrastructure or the kind of workload one needs to process. Reviewing the different systems (RDBMS, key-value, document, column family, graph or time-series) is out of the scope of this work, but the interested reader can refer to [48, 49, 50, 51] to have a better idea of the pro and cons of each model.

As a general guideline for data science practitioners, the evolution of the data should be paramount. Whether the source API changes or if we add new features during the analysis of the data, it is mandatory to have a flexible data store that can accommodate our experiments. We thus recommend a well-maintained, documented open-source NoSQL database as the first choice as it does not require a formal table structure with predefined columns as one could find in traditional SQL data stores such as MySQL or SQLite and easily scales to millions of records. Some of these new-generation databases such as RethinkDB[5] or the multi-paradigm ArangoDB[6] can also directly ingest JSON formatted data, considerably simplifying the process of storing data from an API.

To store our Star Wars dataset, the usage of a graph database, like Neo4j[7], fits our requirements perfectly as it is designed to store and retrieve graphs. It also offers a rich query language to manipulate and traverse graphs which constitute the core of our analysis[8].

Note that despite all their advantages, databases are not optimal to store large binary files such as audio tracks or large pictures. A standard technique is to reference their path on the filesystem in a database record and to keep a separate folder hierarchy to store them. Moreover, to optimize the I/O (input-output) speed, one should create a hierarchy of folders containing parts of the files instead of storing everything in the same folder.

2.4 Processing graphs

Now that we have seen how to store data, we need to be able to create graphs and manipulate them in an efficient manner before moving on to the exploratory analysis. Once again, the size of the dataset dictates the choice of tools. For large-scale datasets, the two principal open-source rivals are Apache GraphX [52] and Turi GraphLab Create [53]. Both provide an API in Python and are part of a larger set of tools dedicated to parallelized machine learning. We refer the reader to chapter 4 and chapter 5 for the usage of GraphLab on large datasets.

[5] https://www.rethinkdb.com/
[6] https://www.arangodb.com/
[7] https://www.neo4j.com/
[8] The Star Wars dataset will be released upon acceptance of the thesis

When the graph size is in below the order of millions of object, we rely on the graph-tool Python library [54] to manipulate the graph in memory (depending on the available RAM). Packed with many algorithms related to traversals, flows, topology or community detection, it is also one of the fastest available thanks to its parallel implementation on top of the Boost Graph Library [55]; well-known by C++ developers for its speed and quality of implementation. Note that the almost standard Networkx package [56] can work for small graphs as it is very flexible and packed with numerous algorithms. However, its pure Python implementation is its greatest weakness as the memory usage skyrockets, and the execution speed is rather slow.

In the case the graph is not naturally given by the data, we build it by comparing the features associated with nodes together. As we will see in the following chapters, knowing what to take as features and compare them in an optimal manner is challenging and also dependent on the application. To be scalable, most of the methods rely on approximations of the k nearest neighbor graph (ANN) that can be computed very efficiently with NMSLib [57] or Annoy[9]. To be concise here, we invite the reader to read the survey of the field by Wang in [58].

2.5 Exploratory data analysis

We now have defined the pipeline to extract, store and process data from the Star Wars wiki. To illustrate the exploratory data analysis loop in Fig. 1.1, we first perform a brief statistical analysis of the corpus while detailing how to create the network of characters. In a second part, we demonstrate the worth of network analysis and machine learning for the digital humanities by showing how we can utilize the network to infer missing values for some of the characters and enrich the dataset with new content.

2.5.1 The graph of characters

To create the graph of characters, we parse the HTML markup and look for other internal wiki links. If one of these links points to another character in our database, we connect them in the graph of characters. Let us suppose that we are currently parsing the biography of Darth Vader (the most connected character of the Star Wars universe), for each of the names appearing in his biography we create a link from the node Darth Vader to the node representing the other character. The number of edges in this directed graph is thus dependent on the exhaustivity and completeness of the biography as well as the community effort to create links between characters.

At the time of the analysis, the graph of characters contained 19,613 identified characters for 66,425 links. Also, the timeline on which characters live spans over 37,000 years. While it is not possible to claim that the Star Wars EU is the biggest fictional universe ever created due to

[9] https://github.com/spotify/annoy

the lack of comparison, it is the most popular [14]. Going deeper in our analysis, we can have a look at the repartition of the different species depicted in the EU, see Fig. 2.1. Surprisingly, and despite one could think, alien species represent less than 23% of the data, the remaining 77% are humans. The distribution of Male / Female is also far from being equal with 12,129 males for 3,515 females, the rest of the characters are undetermined or unknown.

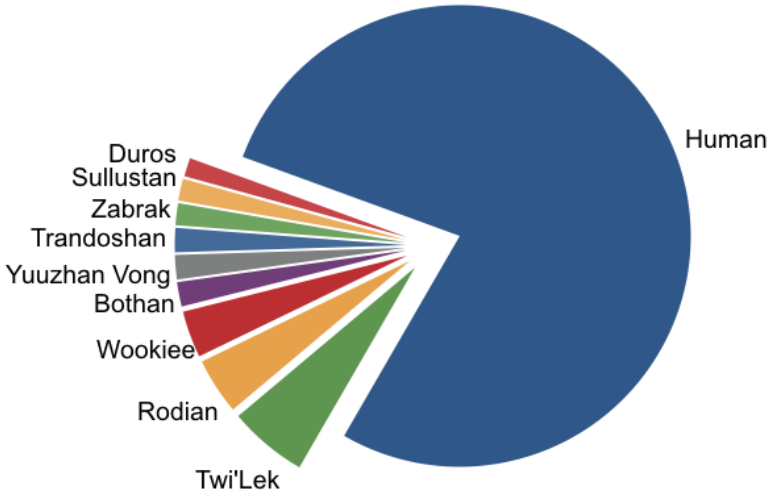


Figure 2.1 – **Distribution of the principal species in the Star Wars expanded universe**. Humans are predominant in the galaxy, they represent 77% of all species.

Looking at the distribution of the nodes with the highest total degree, the sum of inbound and outbound links, we observe that the most connected characters are predominantly found in the movies as shown in Fig. 2.2. This fact is explained by the amount of material accumulated for almost forty years (books, video games, etc.) around the movies. A notable exception is Revan, a human character living thousands of years before the first film. His storyline was mostly developed in the massive multiplayer online role-playing game (MMORPG) Star Wars: The Old Republic.

2.5.2 Inference of missing values

An interesting question brought by the exploration of this dataset is the inference of missing values. Building on our previous statistical analysis, we notice that despite its richness, the Star Wars wiki is also far from being complete as it requires an in-depth knowledge of the fictional world to add content. Moreover, the universe is continuously being updated as Disney (owner of Star Wars) releases new material.

A practical, more advanced, application of network analysis and machine learning could nevertheless help fill missing values of the corpus, de facto expanding our collective knowledge of Star Wars. After an automated submission to the wiki, the effort of verifying and correcting the mistakes of the algorithm could also be crowd-sourced among editors using a dedicated

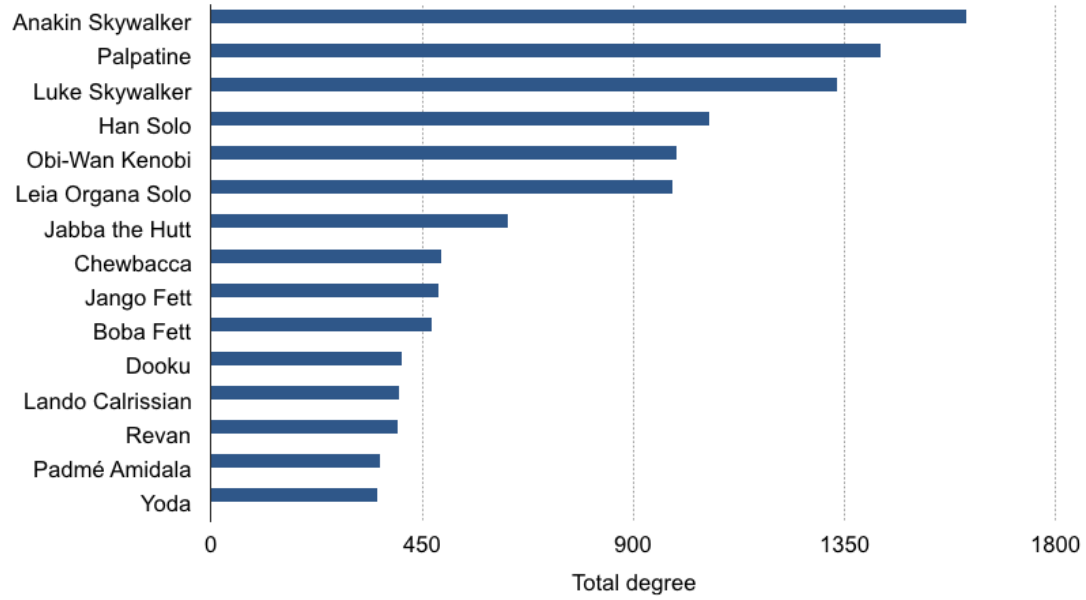


Figure 2.2 – **Total degree of the most connected characters in the Star Wars expanded universe**. With no surprise, the most connected characters are well-known with 14 out of 15 characters in the movies. The most iconic character, Anakin Skywalker / Darth Vader, is the most connected closely followed by the evil Emperor Palpatine. The outlier in this set is Revan, a very famous character from the Old Republic Era, thousands of years before the timeline of the movies.

service such as Amazon Mechanical Turk[10].

To illustrate the idea, we choose to focus on the different Star Wars eras, see Fig. 2.3, and try infer where in the history of Star Wars unlabeled character belong. To do so, we use a slightly modified version of the label propagation algorithm defined in [59].

The general idea of the algorithm is to use the connections between characters to iteratively push until convergence the era of known characters to their undetermined neighbors according to their propability of being influenced by that node. To prevent the originally known set of values of being diluted by the iterative pass of the algorithm, they continuously propagate their value without being influenced by their neighbors. The final label of an undetermined node thus depends on the topology of the graph and the initial distribution of labels as presented in Fig. 2.4.

To ensure the convergence of the algorithm we need to view our graph of characters as undirected. Otherwise, some undetermined nodes with no inbound edges could not be reached by the diffusion of labels. More formally, we define the graph of characters as $G = (V_G, E_G, W_G)$ with $|V_G| = N$, the set of nodes, and E_G the set of edges and $w_{ij} \in W_G$ a map from $E_G \mapsto \mathcal{R}$ representing the edge weights. The affinity matrix or probabilistic transition matrix T

[10] https://www.mturk.com/mturk/welcome

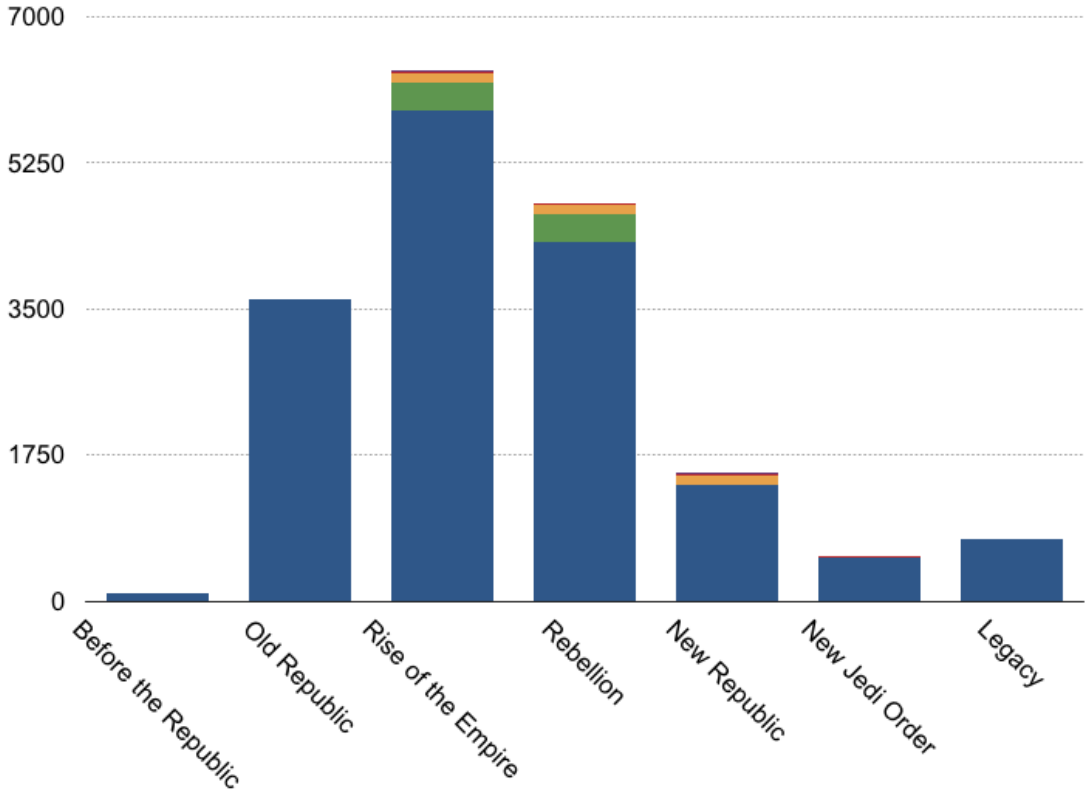


Figure 2.3 – **Character distribution timeline**. Spanning over 37,000 years, the Star Wars expanded universe in divided into seven major eras. The different colors on top of the main bars show characters living across different eras. The Rise of the Empire corresponds to the first three movies, the Rebellion era to the fourth, fifth, and sixth episodes. Note that all that happens after the Rebellion era is no longer considered canon since Disney bought the Star Wars franchise. The avid fan can still find the non-canon material under the term Legends in the wiki.

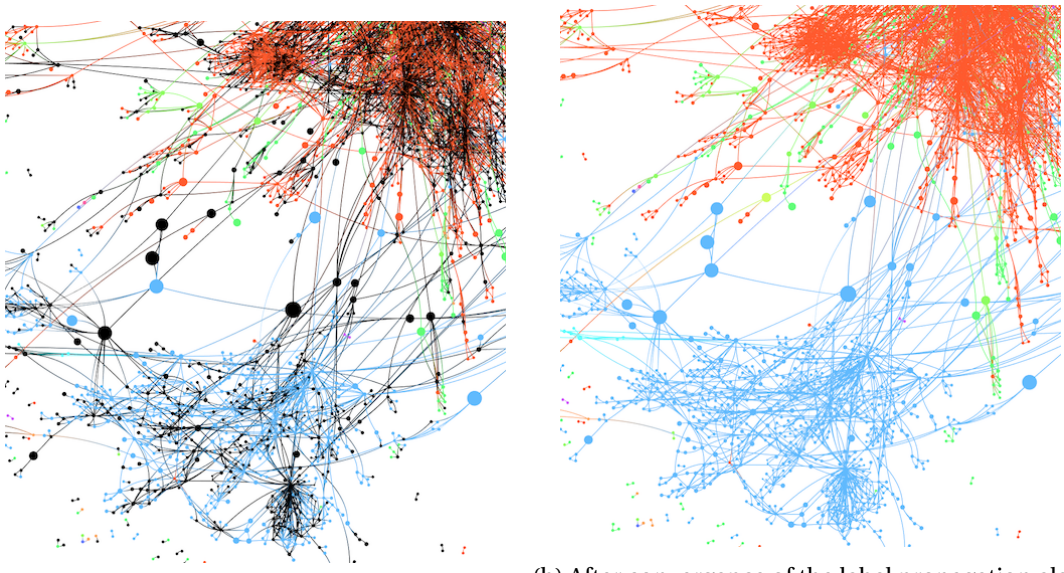
as defined in [59] can be written as the dot product of the inverted degree matrix D with the weighted adjacency matrix W:

$$T = D^{-1}W, \qquad T_{ij} = \frac{w_{ij}}{D_{ii}}, \qquad D_{ii} = \sum_{j=1}^{n} w_{ij}. \tag{2.1}$$

We also define a $N \times C$ label matrix Y where C is the total number of possible eras and where the row Y_i represents the probability of a node $v_i \in V_G$ to belong to an era.

Then for each step s of the algorithm until convergence, we just diffuse the labels $Y^{s+1} = TY$, row-normalize Y to keep a distribution of probability of belonging to a given era and clamp the labeled data to ensure that known labels are not being erased by the algorithm.

In our analysis, the algorithm is trained on 60% of the data while we keep the remaining 40% of the known labels for the test set. We obtain an accuracy of 80% after averaging the results over 10 runs. To improve our predictions of missing eras, we could sort periods and set rules specifically tuned for the Star Wars corpus. Indeed, some of the characters live hundreds of years and also appear to some individuals as ghosts thousands of years later. Because the graph naturally encodes the connection between the ectoplasm and the character, it here influences the outcome of the label propagation algorithm negatively.



(a) Before the label propagation algorithm.

(b) After convergence of the label propagation algorithm.

Figure 2.4 – **Visualization of the label propagation algorithm on the graph of Star Wars characters.** In 2.4a black nodes represent missing values. Red nodes belong to the Rise of the Empire Era, blue nodes to the Rebellion era, green ones to both eras. In 2.4b black nodes are replaced by the best compromise of known eras using the topology of the graph.

2.5.3 Network visualization

An essential part of data science and more particularly network science is data visualization (data-vis or data-viz). More than just displaying the results in an understandable fashion, data visualization is a field of research in itself in between mathematics, computer science, design, and art. For complex networks, it often rimes with finding a good low-dimensional embedding that reveals the structure of an attractive part of the network. While it may seem easy with small networks with the usage of graphical user interface specially designed to layout graphs such as Gephi [60], choosing what to represent, the correct layout algorithm as well as tuning its parameters can be a daunting task especially when the network is big (over 100,000 nodes).

Another factor to consider is the complexity and the implementation (CPU, GPU) of the layout algorithm. To be viable and scale to relatively large size, layout algorithms have to approximate the placement of nodes or use a multilevel scheme instead of computing an

exact solution [61, 62, 63, 64]. Once we fix the position of the nodes, other graphical properties can also be tuned such as coloring the main communities of the network [65] or changing the size of nodes according to their degree or PageRank values [66].

From theory to practice, our last data-viz of this chapter consists of an artwork in Fig 2.5 of Star Wars characters showing the conflict between good (blue) and evil (red) in the universe. The most connected nodes are displayed in the center of the image, and the size of the nodes is directly proportional to their number of connections. The two most influential characters of the whole galaxy in the center are Anakin Skywalker in blue and the Emperor Palpatine in red.

Finally, an important consideration that gives depth to a network visualization is its interactivity and availability online. A review of the most popular interactive graph drawing libraries, or more generally visualization frameworks for the web, can be found in Table.1 of [67]. All written in ECMAscript (Javascript), the lingua-franca of the web for client-side interactions, they require a basic proficiency in this programming language to be used. In this regard, we invite the curious reader to see various interactive graph visualizations in action on the author's personal website[11].

2.6 Publishing results

The last part of our data science pipeline for research consists of publishing the results of our analysis, see Fig. 1.1. Publishing results involve two aspects: writing a paper, journal, patent, technical report describing the methodology and releasing data associated with the study if any.

Let us first focus on the writing. For hundreds of years, this process has not changed, despite the fact that means of communication between authors have evolved. In short, authors write a paper that they submit to a conference or journal. The journal is in charge of finding suitable scientists to review the given work. The publication in the journal depends on the opinion of the reviewers while the final word is left to the editor. To access a published work, one needs to pay. As a result, most of the researchers rely on the deal made by their academic institution to access the work of others.

Very recently, the open science movement forced some of the conferences and journal editors to propose a free access to the written content of an article given that the authors pay for it. Leaving aside the ethical considerations of these practices, another mean of publishing could only consist of deposing a non-peer reviewed version of one's work on popular preprint repository such as ArXiv[12]. However because it has not been rated by the community, the quality of the work might not be sufficient to be claimed as a sound scientific paper.

Another solution suited to our open science movement consists of the writing of a piece of

[11] http://kirellbenzi.com
[12] https://arxiv.org/

work collaboratively and accepting registered or anonymous comments to improve the quality of the work in a transparent manner as proposed by Authorea[13], Overleaf[14] or Gitbook[15]. However, note that considering the current system, only senior scientists can change the publishing process as junior researchers desperately need to prove their worth and thus must conform to the existing mold to succeed.

Moving on to the release of data associated with one's work, we can envision several options depending on the size of the dataset. The simplest solution is to host the data in-house on a Network Attached Storage (NAS) for instance and provide a public download link alongside the paper. Modern solutions could also consist of using dedicated services such as the Open Science Framework[16] to host files related to an experiment. For networks, one can upload to notable repositories such as SNAP [68] or the novel Network Repository [69]. Finally, if the dataset is enormous (over 200 Go), a good solution to save bandwidth and reduce costs consists of hosting the data on Amazon S3[17]. In case the dataset is public and approved by Amazon, it can be downloaded at no cost, thus increasing the chance of usage of the dataset.

2.7 Discussion

In this chapter, we have presented what we consider essential knowledge to push the field of network science forward. Used extensively in the thesis, our formalized methodology to extract, store, analyze and distribute graphs was here illustrated by the study of the Star Wars corpus.

Throughout this example, we have shown that the combination of statistical analysis, machine learning, graph theory and data visualization can lead to the discovery of new insights on complex datasets with applications in various domains such as the digital humanities. Additionally, we have also demonstrated the importance of software engineering for network scientists and data scientists in general, insisting on the choice of tools, the worth of formal training and code reuse.

In the following work, we build on this foundation chapter and continue our exploration of network science by proposing scalable graph-based models that we apply to novel datasets. Our goal is to demonstrate that graph methods are indeed applicable to a broad class of problems starting in the next chapter with recommender systems.

[13] https://www.authorea.com/
[14] https://www.overleaf.com
[15] https://github.com/GitbookIO/gitbook
[16] https://osf.io/
[17] https://aws.amazon.com/public-data-sets/

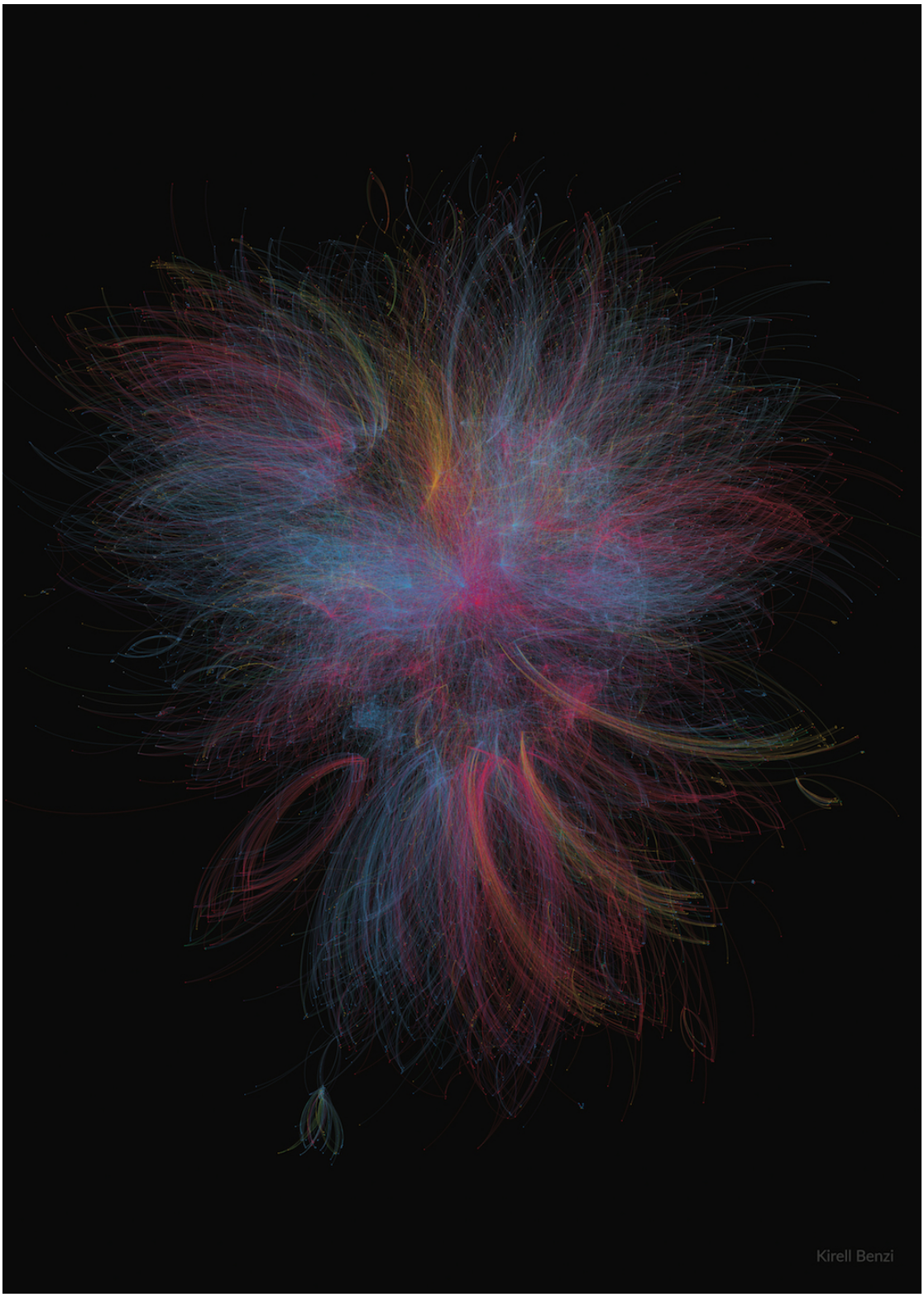


Figure 2.5 – **The Dark Side and the Light - 2015. Copyright Kirell Benzi.** Visualization of the Star Wars expanded universe network. Represented in blue, Jedis, the Republic and the Rebellion fight against the Empire and the Siths in red. The yellow nodes expose criminals and bounty hunters that are, unsurprisingly, mostly connected to evil beings.

3 Graph-based music recommendation

In this chapter, we demonstrate the worth of networks in recommendation systems by studying the making of Genezik, our hybrid playlist recommender application. After an overview of the field of Music Information Retrieval (MIR) and more specifically recommender systems, we present the peculiar hybrid architecture of our application composed of a cloud-based song recommender system [10] using networks combined with an adaptive playlist recommender engine distributed on each client [9] also based on graphs.

We first describe the global (cloud-based) song recommender system model and its performance in a precise and reproducible manner in Sec. 3.3. We dedicate the next part to the implementation of the Genezik client application and its original playlist recommender engine, see Sec. 3.4. This last point reveals some of the challenges that need to be solved when switching from a research context to a final distributed product.

Finally, we conclude this chapter by the introduction of the Free Music Archive dataset in Sec. 3.5. This new dataset solves some of the issues introduced by the development of large scale recommender systems and will hopefully help the MIR community develop new models on a reproducible, free and legal dataset.

3.1 A brief overview of Music Information Retrieval

The MIR community is a relatively small but very active community of researchers devoted to the automatized analysis of music using signal processing techniques as well as machine learning and musicology. Application-oriented, MIR researchers have developed interests in the automatic categorization of music such as genre [70, 71] or mood detection [72, 73]. These techniques are also often used in music recommender systems, another topic of MIR covered in more details in Sec. 3.2, as they principally rely on the extraction of features from the audio signal.

Another interesting application of MIR is track separation and instrument recognition. Heavily drawing on signal processing possible applications include instrumental melody extrac-

tion [74], solo detection [75] or vocal isolation [76, 77].

Building on the other fields of MIR, music transcription aims at converting musical tracks into symbols such as a MIDI file or a score. This process involves detecting instruments, the duration of each sound or the rhythmic structure of the audio recording as explained in [78].

The most innovative and challenging field of MIR is music composition/generation. Here, algorithms are developed to learn the structure of a musical genre to imitate its style using genetic algorithms [79] or recurrent neural networks [80] for instance. Music synthesis, the process of creation of computer generated sound in an automated manner, is also an interesting subdomain of MIR [81, 82].

3.2 Music recommender systems

Recommender systems (rec-sys) are one of the active fields of research in machine learning. As a result, the literature on the subject is incredibly vast and ranges from mathematical frameworks to real-world implementations. For an excellent overview of the domain, we point the reader to these three complete surveys [83, 84, 85]. In our case, we specifically focus on the recommendation of music, itself divided into two categories: song recommenders and playlist recommendation (playlisters). The goal of any of these type of systems is to return the "best" recommendations that would adapt to user preferences, but in the case of playlisters, the system also has to optimize the ordering as well as the coherence as a whole.

Defining what constitutes the best recommendations for music is extremely challenging as it is entirely subjective and thus hard to assess [86, 87, 88]. For instance, let us consider a concerto of W.A. Mozart and an electro-house remix with the same melody. We could consider it as musically close to the original song by an algorithm where the similarity criteria only focuses on the melody. However, if we were to choose the rhythm as the main similarity criteria, the outcome would be entirely different underlying the importance of the choice of features and metric to compare songs.

These content-based (CB) algorithms [89, 90, 91], working with features extracted from the audio signal and metadata, are opposed to collaborative filtering (CF) algorithms, solely dependent on community-provided ratings [92, 93, 94, 95]. The last approach for playlist recommendation tries to find similar patterns of song transitions (using, for instance, the LDA technique) and try to reproduce them for the recommendation, see [94, 91]. For more information on music recommender systems, we invite the reader to review the surveys of Celma [96], Bonnin *et al.* [97] and Schedl *et al.* [98].

On the one hand, CB algorithms offer smooth musical transitions and greater heterogeneity but are dependent on the choice of a good metric while being computationally expensive. On the other hand, CF algorithms do not need to extract musical features and follow the ratings, popularity, and appreciation of songs by users. However, they lack diversity and most

importantly cannot order songs, crucial when recommending playlists. Another drawback of CF is that one cannot "bootstrap" a new user: if there are not enough ratings nothing can be recommended. Finally, the last approach called frequent pattern mining in [97], tries to mimic human playlist generation but fails at suggesting new transitions never seen in the training set.

We clearly see that the three solutions have advantages and drawbacks, we thus suspect that the best recommender systems can only come from the merging of both content-based and collaborative filtering models with the inclusion of frequent pattern mining to learn from existing playlists. These hybrid recommender systems have been used more specifically in the MIR community to overcome the drawbacks of each models [97]. Our music recommendation application Genezik, described in the following sections, belongs to the list of these hybrid systems as it virtually employs all hybridization strategies reported in Table. 3.1.

Hybridization method	**Description**
Weighted	Scores from several recommendation techniques are combined
Switching	System switches between models depending on the situation
Mixed	Recommendation from different rec-sys are presented together
Feature combination	Features from several sources are mixed into one single rec-sys
Feature augmentation	The output of one model is used as an input feature to another one
Cascade	One rec-sys refines recommendations given by another
Meta-level	One model serves as input to the other

Table 3.1 – **Hybridization strategies for audio recommender systems.**

3.3 Song recommendation with non-negative matrix factorization and graph total variation

We first start our journey into the application of network science to music by describing the model of our global song recommender system [10]. Note that this work was done in collaboration with Vassilis Kalofolias and Xavier Bresson for the problem formulation.

Our recommender system is modeled as a matrix completion problem that benefits from collaborative filtering through Non-negative Matrix Factorization (NMF) and content-based filtering via Total Variation (TV) on graphs as illustrated in Fig. 3.1. The graphs encode both playlist proximity information and song similarity, using a rich combination of audio, meta-data and social features distributively collected from users of the system. As we demonstrate, our hybrid recommendation system is very versatile and incorporates several well-known methods while outperforming them. Particularly, we show on real-world data that our model overcomes w.r.t. two evaluation metrics the recommendation of models solely based on low-rank prior of the rating matrix, graph-based information or a combination of both.

The choice of low-rank matrix factorization techniques [99] is motivated by their effectiveness for collaborative filtering. Indeed, they were amongst the winners of the famous Netflix Prize,

involving explicit user ratings as input. Similar techniques were soon used to solve implicit feedback problems, where item preferences were implied for example by the actions of a user [100, 101]. Recently, the utility of networks was once again demonstrated with the introduction of graph regularization to enhance the quality of matrix completion problems [102, 103, 104].

We base our method on solving a succession of well-posed convex optimization problems based on proximal splitting methods [105]. The structure implied by the similarity information between songs and between playlists is imposed by penalizing the TV energy [106] of the factors on their corresponding graphs. While graph regularization has been previously suggested to enhance NMF [104], we show that using TV instead of Tikhonov regularization yields better results regarding recommendation quality.

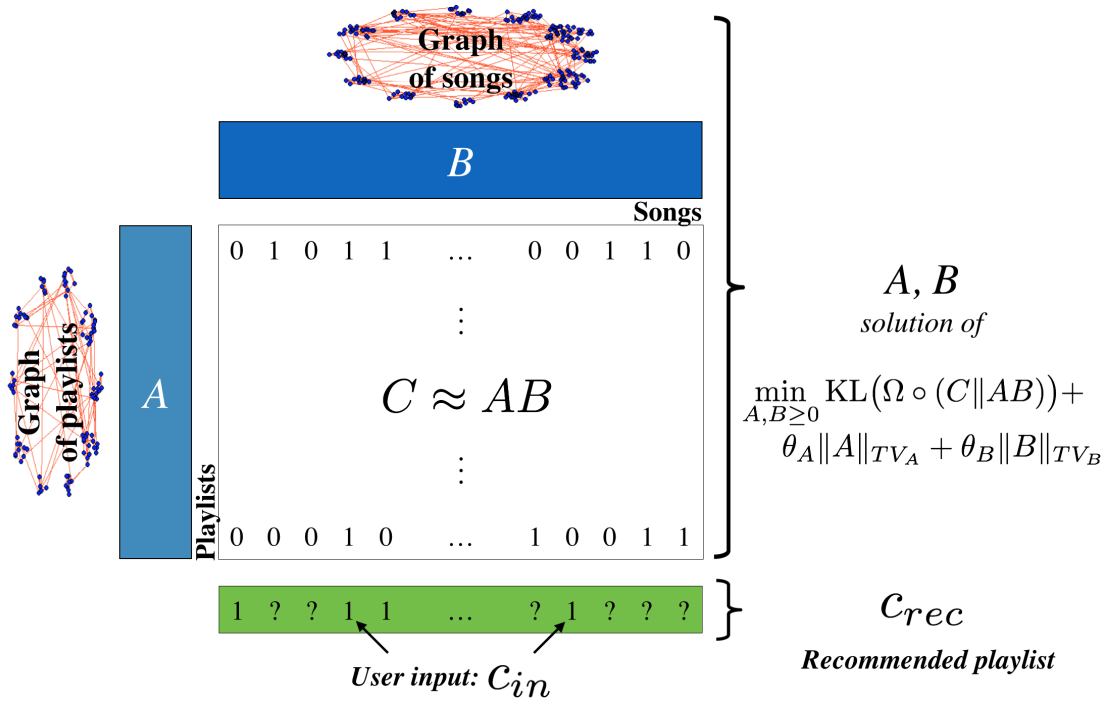


Figure 3.1 – **The architecture of our global song recommender system.**

3.3.1 Problem formulation

Suppose we are given n playlists, each containing some of m songs. We define matrix $C \in \{0,1\}^{n\times m}$ as in [94, 101], that has a value $C_{ij} = 1$ if playlist i contains song j, 0 otherwise. We also define a weight mask $\Omega \in \{\varepsilon, 1\}^{n\times m}$ that has a "confidence" value $\Omega_{ij} = 1$ one if the entry C_{ij} is 1, and a small value ε, otherwise (we use $\varepsilon = 0.1$). This follows the example of implicit feedback problems [100], since a zero in matrix C does not mean that the corresponding song is irrelevant to the playlist, but that it is less probably relevant.

The goal of the training step is to find an approximate low-rank representation $AB \approx C$, where $A \in \mathbb{R}_+^{n\times r}$, $B \in \mathbb{R}_+^{r\times m}$ non-negative and with small r. This problem is known as Non-Negative

Matrix Factorization (NMF) and has drawn much attention after the original work [107].

The advantage of NMF over other factorization techniques is that the approximation is only based on adding factors, a property explained as *learning the parts of objects* [107], in this case, the playlists. NMF comes to the cost of being NP-hard [108], so sophisticated regularization is necessary to find a good local minimum. In our problem, we use outside information given by the songs and playlists graphs to give structure to the factors A and B. Our model is formulated as

$$\min_{A,B\geq 0} \mathrm{KL}\big(\Omega \circ (C \| AB)\big) + \theta_A \|A\|_{TV_A} + \theta_B \|B\|_{TV_B}, \tag{3.1}$$

where $\circ$ is the point-wise multiplication operator and $\theta_A, \theta_B \in \mathbb{R}_+$. We use a weighted Kullback-Leibler (KL) divergence as a distance measure between C and AB, that has been shown to be more accurate than the Frobenius norm for various NMF settings [109]. The second term is the TV of the rows of A on the playlists graph, so penalizing it promotes piecewise constant signals [106]. Similarly with the third term for columns of B.

Overall, the model (3.1) makes use of similar patterns of playlists (1^{st} term), proximity between playlists (2^{nd} term) and songs (3^{rd} term) while leveraging the works of [103, 109] and extending them to graphs using the TV semi-norm.

3.3.2 Graph regularization with total variation

In our NMF-based recommender, each playlist i is represented in a low-dimensional space by a row A_i of the matrix A. In order to learn better low-rank representations A_i of the playlists, we also impose the pairwise similarities of the playlists $w^A_{ii'}$ on their corresponding low-rank representations. We can see this from the definition of the TV regularization term, $\|A\|_{TV_A} = \frac{1}{2}\sum_i \sum_{i'\sim i} w^A_{ii'} \|A_i - A_{i'}\|_1$. Hence, when two playlists i, i' are similar then they are also well-connected on the graph and the weight of the edge connecting these two playlists $w^A_{ii'}$ is large (here $w^A_{ii'} \approx 1$). Moreover, any large distance between the corresponding low-dimensional representation vectors $(A_i, A_{i'})$ is penalized, forcing $(A_i, A_{i'})$ to stay close in the low-dimensional space. In a similar way, each song j is represented in a low-dimensional space by a column B_j of the matrix B. If two songs (j, j') are close ($w^B_{jj'} \approx 1$), so will be $(B_j, B_{j'})$ with the graph regularization $\|B\|_{TV_B}$.

A similar idea has been used in [104] by incorporating the graph information through Tikhonov regularization, i.e. with the Dirichlet energy term $\frac{1}{2}\sum_i \sum_{i'\sim i} w^A_{ii'} \|A_i - A_{i'}\|_2^2$. However, the latter promotes smooth changes between the columns of A, while the graph TV term penalization promotes piecewise constant signals with potentially sharp transitions between columns A_i and $A_{i'}$. This is advantageous in applications where well-separated classes are sought, for example in clustering [110] or in our recommendation system where similar playlists might belong to different categories.

As we demonstrate in Sec. 3.3.6, the use of the graphs of songs and playlists improve significantly the recommendations. Besides, the results are better when the more forgiving TV term is used instead of Tikhonov regularization.

3.3.3 Primal-dual optimization

The optimization problem (3.1) is globally non-convex but separately convex w.r.t. A and B. A standard strategy is thus to optimize B for fixed A, then optimize A for fixed B, and repeat until convergence. We describe here the proposed optimization algorithm w.r.t. B for fixed A based on [111, 105, 109]. The same algorithm is also applied to A for fixed B. Let us rewrite the problem (3.1) as:

$$\min_{B \geq 0} \; F(AB) + G(K_B B), \tag{3.2}$$

where

$$F(AB) = \mathrm{KL}\big(\Omega \circ (C \| AB)\big) = \tag{3.3}$$
$$\sum_{i=1}^{m} \sum_{j=1}^{n} \Big(-\Omega_{ij} C_{ij} \Big(\log \frac{(AB)_{ij}}{C_{ij}} + 1 \Big) + \Omega_{ij} (AB)_{ij} \Big),$$
$$G(K_B B) = \theta_B \| B \|_{TV_B} = \theta_B \| K_B B \|_1, \tag{3.4}$$

where $K_B \in \mathbb{R}^{n_e \times m}$ is the graph gradient operator [110], with n_e being the number of edges in the graph of B. Using the conjugate functions $F^\star$ and $G^\star$ of F and G, (3.2) is equivalent to the saddle-point problem:

$$\min_{B \geq 0} \max_{Y_1, Y_2} \; \mathrm{tr}((AB)^T \cdot Y_1) - F^\star(Y_1) + \mathrm{tr}((KB^T)^T \cdot Y_2) - G^\star(Y_2), \tag{3.5}$$

where $Y_1 \in \mathbb{R}^{n \times m}$, $Y_2 \in \mathbb{R}^{n_e \times r}$. Let us now introduce the proximal terms and the time steps

$\sigma_1, \sigma_2, \tau_1, \tau_2$:

$$\begin{aligned}&\min_{B\geq 0}\max_{Y_1,Y_2}\ \mathrm{tr}((AB)^T\cdot Y_1) - F^\star(Y_1) + \\ &\mathrm{tr}((KB^T)^T\cdot Y_2) - G^\star(Y_2) + \frac{\tau_1+\tau_2}{2\tau_1\tau_2}\|B-B^k\|_F^2 \\ &-\frac{1}{2\sigma_1}\|Y_1-Y_1^k\|_F^2 - \frac{1}{2\sigma_2}\|Y_2-Y_2^k\|_F^2. \end{aligned} \tag{3.6}$$

The iterative scheme is thus for $k \geq 0$:

$$Y_1^{k+1} = \mathrm{prox}_{\sigma_1 F^\star}(Y_1^k + \sigma_1 AB^k), \tag{3.7}$$
$$Y_2^{k+1} = \mathrm{prox}_{\sigma_2 G^\star}(Y_2^k + \sigma_2 K_B B^k), \tag{3.8}$$
$$B^{k+1} = (B^k - \tau_1 A^T Y_1^{k+1} - \tau_2 (K_B^T Y_2^{k+1})^T)_+, \tag{3.9}$$

where prox is the proximal operator [105] and $(\cdot)_+ = \max(\cdot, 0)$. For our problem we choose the standard Arrow-Hurwicz time steps $\sigma_1 = \tau_1 = 1/\|A\|$ and $\sigma_2 = \tau_2 = 1/\|K\|$, where here $\|\cdot\|$ is the operator norm.

The proximal solutions (3.7) and (3.8) are given by:

$$\begin{aligned}\mathrm{prox}_{\sigma_1 F^\star}(Y) &= \frac{1}{2}\left(Y + \Omega - \sqrt{(Y-\Omega)^2 + 4\sigma_1 \Omega \circ C}\right) \\ \mathrm{prox}_{\sigma_2 G^\star}(Y) &= Y - \mathrm{shrink}(Y, \theta_B/\sigma_2), \end{aligned} \tag{3.10}$$

where shrink is the soft shrinkage operator [112].

Note that the same algorithm could be used for Tikhonov regularization, i.e. replacing $\|K_B B\|_1$ by $G(K_B B) = \frac{\theta_B}{2}\|K_B B\|_2^2$ by just changing the first proximal (3.10) to $\mathrm{prox}_{\sigma_2 G^\star}(Y) = \frac{\theta_B}{\sigma_2+\theta_B} Y$. In [104] this regularization is used along with a symmetric version of the KL divergence, however, the latter has no analytic solution, unlike the one we use in this work. As a result, their objective function does not fit an efficient primal-dual optimization scheme like the one we propose. We thus choose to keep the nonsymmetric KL model, denoted as GNMF in this paper, in order to compare the TV versus Tikhonov regularization.

3.3.4 Recommending songs

Once we have learned matrices A and B by solving (3.1), we wish to recommend a new playlist c_{rec} given a few songs c_{in} (see Fig. 3.1). Preoccupied with a real-world usage of our system, we craft of fast recommender function that we detail in the following.

Given the songs c_{in}, we first find a good representation of the query on the learned low-rank space of playlists by solving a regularized least squares problem:
$a_{in} = \arg\min_{a \in \mathbb{R}^{1\times r}} \|\Omega_{in} \circ (c_{in} - aB)\|_2^2 + \varepsilon \|a\|_2^2$. The latter enjoys an analytic solution $a_{in} = (B^T \Omega_{in} B + \varepsilon I)^{-1} (B^T \Omega_{in} c_{in})$ that is cheap to compute as r is small (we use $\varepsilon = 0.01$).

The recommended playlist can benefit from the playlists that have similar representations as the one of the query, thus we use the weighted sum $a_{rec} = \sum_{i=1}^{n} w_i A_i / \sum_{i=1}^{n} w_i$ as the representation of the recommended playlist in the low dimensional space. Here the weights w_i are defined as $w_i = e^{-\|a_{in} - A_i\|_2^2 / \sigma^2}$ and depend on the distance of a_{in} from other playlists representations, while $\sigma = \text{mean}_i(\{\|a_{in} - A_i\|_2\}_{i=1}^{n})/4$. The final recommended playlist uses the low-rank representation a_{rec}:

$$c_{rec} = a_{rec} B. \tag{3.11}$$

Note finally that the recommended playlist c_{rec} is not binary, but with continued values that serve as song rankings.

3.3.5 Graphs of playlists and songs

Now that we presented the model of our song recommender system let us take a closer look at the graph of playlists and graph of songs. Because Genezik is a closed-source application, we do not use the gathered playlists and songs to show the performance of our model. Instead, for greater validation and reproducibility, we present the results of our model on the freely available Art of the Mix dataset gather by McFee et al. in [113].

The Art of the Mix is a collaborative website[1] where users can share their mixes. A mix is a special kind of playlist inspired by DJs where songs are supposedly coherent altogether by following a user-defined theme. Besides, the length of a mix rarely exceeds the number of songs that can fit on an audio CD (around 15 songs). In this work, we restrict the word playlist to mix and use both of them interchangeably.

In [113], the authors crawled the site for all playlists from 1998 to 2011 resulting in a dataset Aotm2011 of 101,343 playlists. A playlist is composed of a timestamp, a creator, a playlist category and a list of plain-text song titles and artist names. To extract features from these songs, the authors used a full-text search on the Million Song Dataset (MSD) [114] containing Echonest features of one million popular tracks. The resulting dataset is composed of 98,359 songs. The MSD has proven essential in the last five years for the Music Information Retrieval (MIR) community, see Sec. 3.5. However, this dataset is now outdated in the sense that the

[1] http://www.artofthemix.com

MSD features cannot be ID matched to the Echonest[2] features with their latest API (API v4). Besides, the generation of real playlists from popular streaming services such as Spotify, Deezer or Rdio has been made difficult by the lack of mapping between MSD IDs and those services.

A contribution of this work is to overcome the issues mentioned above by building a new crawler using Echonest, LastFm, and Musixmatch APIs to extract the most up-to-date version of the Aotm2011 dataset. This new dataset contains more than 270,000 songs with all Echonest features, LastFm terms associated with each song as well as the N-grams (here $N = 5$) of their lyrics.

This dataset, available at https://lts2.epfl.ch/datasets/fma/, is the perfect candidate to simulate the outcome of our local playlist recommender system as it constituted of playlists solely made by humans. It is thus perfectly adapted to the taste of mix creators, the goal we want to achieve with our system.

Playlists graph

The playlists graph naturally encodes pairwise similarities between playlists. The set of nodes of this graph is the set of playlists, and the edge weight provides the proximity between two playlists. A large weight (here $w^A_{ii'} \approx 1$) implies a strong proximity between the playlists. In this work, the edge weight of the playlists graph uses both "outside" information, i.e. the meta-data, and "inside" information, i.e. the songs that form the playlists. As meta-data, we use the predefined Art of the Mix playlist categories [113] onto which users label their mixes. The edge weight of the playlists graph is thus defined as follows:

$$w^A_{ii'} = \gamma_1 \delta_{cat\{i\}=cat\{i'\}} + \gamma_2 \, \mathrm{sim}_{\cos}(C_i, C_{i'}), \tag{3.12}$$

where cat stands for playlist category, C_i is the i^{th} row of matrix C and
$\mathrm{sim}_{\cos}(p,q) = p^\top q/(\|p\|.\|q\|)$ is the cosine similarity distance between the vectors of the songs of the two playlists. In our case, the cosine similarity is the ratio between the songs in common and the square root of the product of the lengths of the two playlists. The two positive parameters γ_1, γ_2 with $\gamma_1 + \gamma_2 = 1$ allow weighting the importance of the playlist labels against their element-wise similarity. To control the edge density in each category and to give more flexibility to our recommendation model, we keep a random subset of 20% of the edges between nodes of the same category. As we find experimentally, $\gamma_2 = 0.3$ constitutes a good compromise, see Sec. 3.3.6.

We measure the quality of the playlist graph by partitioning the graph using the standard

[2] http://the.echonest.com/

Louvain's method [115]. The number of partitions is automatically given by the modularity dendrogram which is cut where the modularity is maximal. The graph used in Sec. 3.3.6 has a modularity of 0.63 when using the cosine similarity ($\gamma_2 = 0$) only as shown in Fig. 3.2. If we add the metadata information by connecting 20% of all playlist pairs within each category with $\gamma_2 = 0.3$, the modularity increases to 0.82.

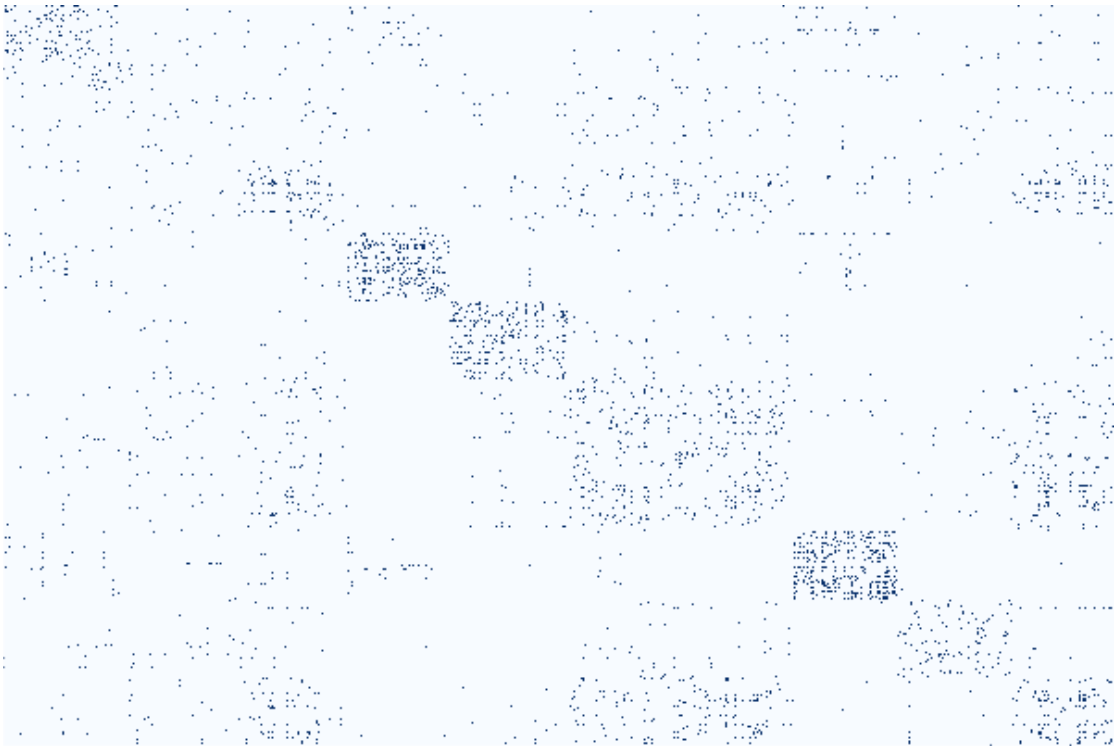


Figure 3.2 – **Part of the adjacency matrix of the playlist graph.** Nodes are here ordered according to their playlist category and connected using cosine similarity. We see that the graph is already well-structured and very sparse without using the category metadata.

Songs graph

The second graph used in our model is the graph of song similarity. To create the edges of the song network, we first extract features from the audio signal that we combine with metadata information and social features for the track.

Let us now describe the features used to create the graph of songs in details; the reader can review them more concisely in Tab. 3.2. To be very clear and reproducible, we present our model with the usage of Echonest features, freely available for research purposes. Due to its commercial nature, Echonest features cannot be used in a closed-source application that could generate profit (Genezik). The audio features employed in the actual implementation of Genezik are similar to the ones presented here, see 3.4.

Audio features. We first compute the Temporal Echonest Features (TEF) described in [116]

High Level Features	
acousticness	Acoustic or electric?
valence	Is the song positive or negative?
energy	How energetic is the song?
liveness	Is it a "live" recording?
speechiness	How many spoken words?
danceability	Is the song danceable?
tempo	Normalized BPM.
instrumentalness	Is the song instrumental?
Social Features	
artist discovery	How unexpectedly popular is the artist?
artist familiarity	How familiar is the artist?
artist hotttnesss	Is the artist currently popular?
song hotttnesss	Is the song currently popular?
song currency	How recently has it become popular?
Temporal Echonest Features	
statistics on Echonest segments	Described in [116]
Metadata Features	
genre	ID3 genre extracted from tags given by LastFM api

Table 3.2 – The features used to create the graph of songs.

by using the Echonest Track API. These features have been proven to be on par with similar state-of-the-art features for genre classification. Specifically, they constitute a list of statistics: mean, median, variance, min, max, value range, skewness, kurtosis of the features contained in Echonest segments: pitch, timbre, loudness max, loudness max time, loudness start, segment duration. For more information on those low-level features we invite the reader to refer to the Echonest Analyzer documentation[3]. In addition to TEF, we add some of the high-level Echonest features listed in Table 3.2.

Social features. Capturing user's familiarity is essential when designing playlist recommender systems [97]. In [117], the weight of the social features learned by their model on a smaller version of the Art of the Mix dataset accounted for more than 35% of the total. See Table 3.2 for a list of the social features we use.

Metadata features. Using the LastFm API, we crawl all terms (tags) associated with each song. While being very informative, those terms have a low quality and lot of subjective information such as: "best song ever". To extract real music genres we use the Levenshtein distance [118] between those terms weighted by their popularity (according to LastFm) and the music genres defined in the ID3 tags. This procedure improves drastically the quality of the labels given by users for genre extraction. While it is not used directly in the final feature vector, the genre is

[3]http://developer.echonest.com/docs/v4/_static/AnalyzeDocumentation.pdf

used indirectly to learn a useful metric for audio features.

Metric learning. The default metric we use to compare features may not be optimal as it would give the same importance to each of the features. A standard technique to alleviate this issue is to learn a meaningful weighting for each audio feature that predicts at best the song genres. We use the Large Margin Nearest Neighbors model [119] that returns a linear transformation so that the original features, $x_i \in \mathbb{R}^d$, are projected to the new features $\hat{x}_i = T x_i$, where $T \in \mathbb{R}^{d \times d}$. We obtain our final feature matrix by concatenating the social features with the projections of the audio features onto the learned space as illustrated in Fig. 3.3. The size of the feature matrix is $|V| \times 237$, where $|V|$ is the number of vertices in the graph, i.e. the number of songs.

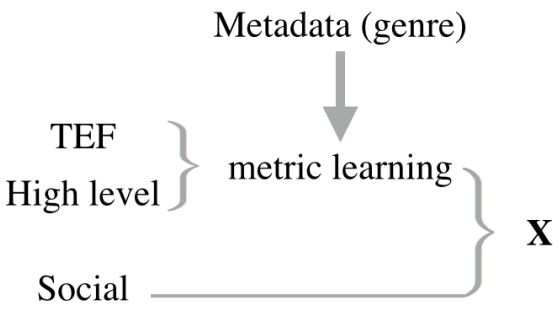


Figure 3.3 – **Features combination to build the song graph.** Temporal Echonest Features (TEF) are combined with high-level features such as acousticness. A metric is trained to recognize genres using processed tags from LastFm on this combination of audio features. Once we know how to weights each of these features to detect genres we combine them with social features such as the artist popularity or song familiarity to obtain our final feature vector x.

Eventually, we create the songs graph using the k nearest neighbors (here $k = 5$) where the edge weight between two songs j, j' is given by:

$$w^B_{jj'} = \exp(-\left\| x_j - x_{j'} \right\|_1 / \sigma) \tag{3.13}$$

where j' is the k^{th} nearest neighbors of j.

The parameter σ acts as the scale parameter of the graph and is set to be the average distance of the k^{th} neighbors. The obtained graph has a high modularity (0.64) and is quite pure with respect to song genres with around 65% of accuracy using an unsupervised k-NN classifier. A part of the adjacency matrix for the graph of songs is shown in Fig. 3.4.

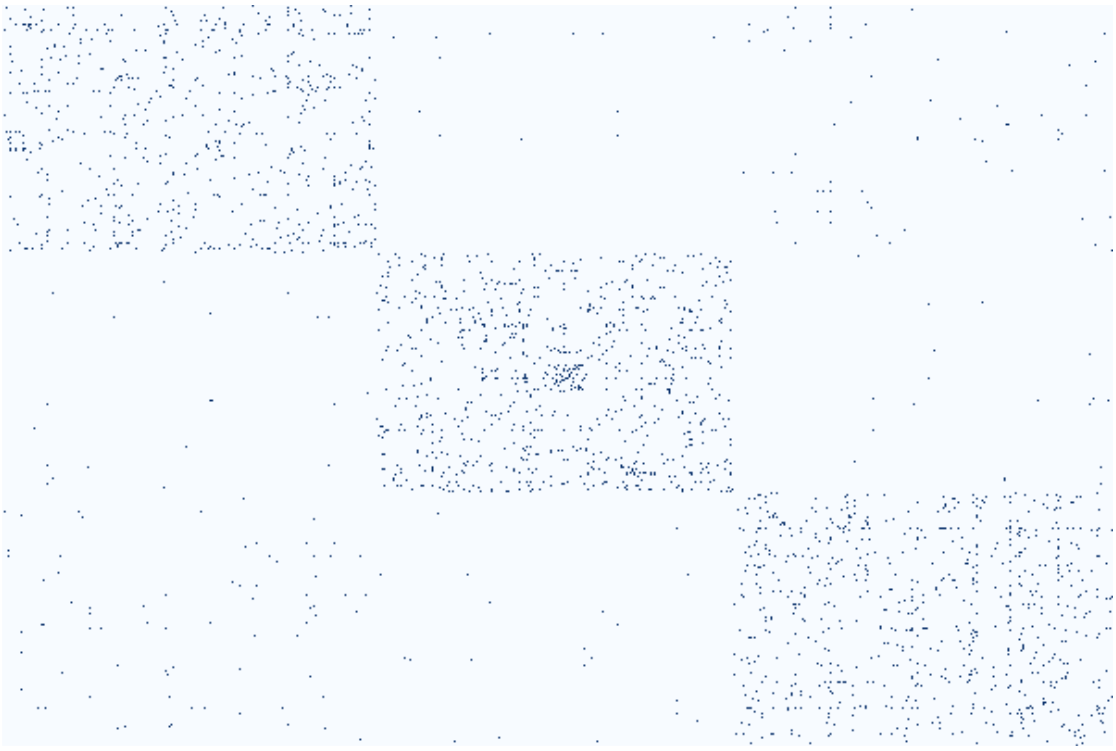


Figure 3.4 – **Part of the adjacency matrix of the song graph.** The graph is created using the 5 nearest neighbors of each song. Ordered by community ID, the graph is well structured and sparse.

3.3.6 Experimental results

In this section, we validate our approach by comparing our model against three different recommender systems on the Art-of-the-Mix corpus enriched by all the features described previously.

Assessing the quality of any music recommender systems is well-known to be a challenging problem [97]. In our system, we use a typical metric for recommender system with implicit feedback, *Mean Percentage Ranking (MPR)* described in [100] and the *playlist category accuracy*, that is the percentage of the recommended songs that have already been used in playlists from the requested category in the past.

Models. We first compare our model against a graphs-only based approach, labeled as *Cosine only*. For a given input, this model computes the t-closest playlists (here $t = 50$) using cosine similarity. We recommend songs by calculating a histogram of all the songs contained in these playlists weighted by the cosine similarity weight, as defined by eq. (3.11). The second model is NMF using KL divergence, labeled *NMF [109]*. Note that the latter can be seen as a special case of our model, where we have set $\theta_A = \theta_B = 0$, to exclude the influence of the graphs of songs and playlists on the low-rank representations A and B. The last model, *GNMF [104]* described in Sec. 3.3.1 is based on the KL divergence with Tikhonov regularization using the

graphs of our model.

Queries. We test our model with three different types of queries. In all cases, a query c_{test} contains $s = 3$ input songs, and the system returns the top $k = 30$ output songs as a playlist using eq. (3.11). The first type of queries, *Random*, contains completely randomly chosen songs from all categories and is solely used as a comparison baseline.

The second type of queries, *Test*, picks randomly 3 songs from a playlist of the test set. Lastly, *Sampled* contains randomly chosen songs from a given category. It simulates a recommender system based on chosen playlist categories input by a user.

Training. We train our model using a randomly selected subset of 70% of the playlists. The rest of playlists, 30%, is kept for testing (next subsection). As our model is not jointly convex, initialization may change the performance of the system, so we use the nowadays standard technique of NNDSVD [120] to get a good approximate solution. Another approach would be to start from different random initializations and keep the system that performs best on a validation set. In all our experiments a value of the rank $r = 15$ performs well, which is expected as each row has between 5 and 20 non-zero values. Small changes of r do not change the performance a lot, but given the very sparse structure of C, our model could overfit for much larger values. The best set of parameters $\theta_A = 18$ and $\theta_B = 1$ is found using a grid search using queries on the validation set. To prevent overfitting, we perform *early stopping* as soon as the MPR on the validation set ceases to increase. The existence of the regularization term for the graph of songs helps the training process to converge faster. Using $\theta_B = 1$ reduces by half the number of outer iterations needed to reach the maximum performance on the validation set compared to setting $\theta_B = 0$.

Validation set. We create the "playlists" of the validation set by creating artificial queries from the different playlist categories. That is, for each category, we randomly pick $s = 3$ songs that have been previously used in user-made playlists labeled by the given category. This simulates the user behavior that wants a recommendation given 3 similar songs.

Results. The performance in terms of playlist category accuracy and MPR of the different models are reported in Table 3.3 and Table 3.4 respectively. As expected, for random category queries all models fail to return playlists from the categories of the input songs. At the same time, the performance of NMF as collaborative filtering without the graphs information is poor. This can be explained by the sparsity of the dataset, which only contains 5 to 20 non-zero elements per row, i.e. only 0.11-0.46% sparsity. Collaborative filtering models are known to perform better as more observed ratings are available [103]. The cosine model performs better according to category accuracy, as it directly uses the cosine distance between the input query and playlists from pure categories. However, its high MPR value shows that our model, albeit more complex, achieves better song recommendations.

The second baseline model performs better in terms of category accuracy, as it directly uses the cosine distance between the input query and playlists from pure categories. The combined

	Cosine only	NMF [109]	GNMF [104]	$\gamma_1 = 0$ $\gamma_2 = 1$	$\gamma_1 = 0.3$ $\gamma_2 = 0.7$
Random	0.135	0.150	0.167	0.210	0.183
Test	0.530	0.236	0.332	0.544	**0.646**
Sampled	0.822	0.237	0.366	0.598	**0.846**

Table 3.3 – Category accuracy for all models for different types of 3-song queries (higher is better). Results are averaged over 10 train/validation runs with 300 queries each.

	Cosine only	NMF [109]	GNMF [104]	$\gamma_1 = 0$ $\gamma_2 = 1$	$\gamma_1 = 0.3$ $\gamma_2 = 0.7$
Test	0.208	0.248	0.181	0.153	**0.146**
Sampled	0.226	0.319	0.211	0.164	**0.074**

Table 3.4 – Mean percentage ranking (MPR) for all models for different types of 3-song queries (lower is better). Results are averaged over 10 train/validation runs with 300 queries each.

model we propose gives more meaningful output playlists both when the category information is not known during the training phase ($\gamma_1 = 0$) and when it is known ($\gamma_1 = 0.3$). It is noteworthy that in the playlists of the test set some songs have never been seen during the training phase. The same thing holds for the "sampled" queries, but the frequency of these songs is smaller in this case. This explains the fact that the algorithm performs worse on the playlists of the test set. Fig. 3.5 shows the accuracies of our model for $\gamma_1 = 0.3$, $\gamma_2 = 0.7$ for each category.

Trying to solely minimize the category accuracy if we had the category information for all playlists during training (e.g. last column of table 3.4) could be trivially done by returning songs that have previously been used in such playlists. However, we deliberately use a small part of this information ($\gamma_1 = 0.3$) so that the results are also driven by the learned user behavior proposed by the collaborative filtering, as well as the songs similarity imposed by the songs graph. If we compare columns 3 and 1 we see that even without using any category information during training ($\gamma_1 = 0$), the usage of the graphs results to more meaningful playlists.

To conclude this section, let us summarize what we have presented so far. In this work, we have introduced a flexible song recommender system that combines collaborative filtering with playlist and song proximity information encoded by graphs. We have used a primal-dual based optimization scheme based graph TV instead of Tikhonov regularization to achieve a highly parallelizable algorithm with the potential to scale up to very large datasets. We have also demonstrated the model's superiority by comparing our system against three other recommendation models on a music playlists dataset collaboratively made by humans.

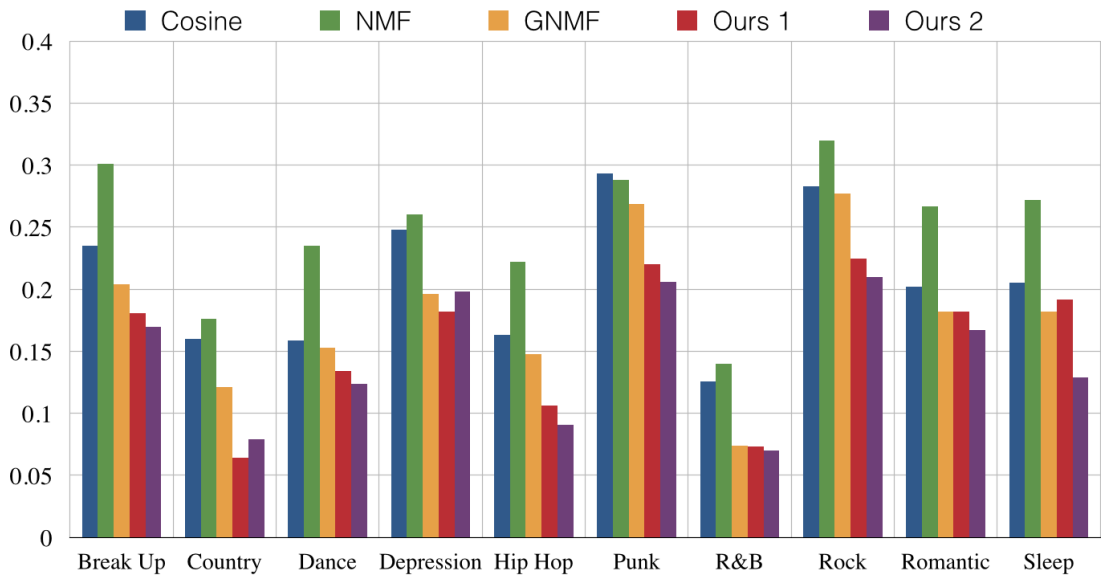


Figure 3.5 – MPR for each playlist category on the test set. Our models use the same parameters of Table 3.4. Ambiguous categories such as Rock, Punk have the highest MPR on the test set. Our model outperforms significantly the others methods on those specific categories.

3.4 Genezik

In the previous section, we have described how to recommend songs using graphs. Here, we present the inner-workings of Genezik, our real-world *playlist* recommender application that combines our previous model with an interesting distributed graph architecture for playlist recommendation. Particularly, we show several ways to generate playlists to tackle the even more challenging problem of ordering songs according user taste. Building a real-world recommender system with active users is completely different from simply building an experimental model for a research article. Closing the gap between research and industry, the development of Genezik is of special interest in the context of data science where many practical issues need to be solved for its application to real problems.

Composed of a distributed multi-platform client application (Mac, Linux, Windows, iPhone, iPad), Genezik interacts with a cloud-based service, it roughly represents 65,000 lines of code in 6 different programming languages (C, C++, Obj-C, Scala, Python, Javascript). Principally developed by myself with the help of Jean Rossier for the cloud part and Florian Carrere for the client application, our *cross-platform* application represents years of development. This design choice imposes a *vast* number of constraints that anyone familiar with software development rightfully fears. In particular, the selection of libraries to implement the different components of the system must have to be very carefully chosen as it can completely stop the deployment of the app. Genezik recommendation model relies on the combination of all the hybridization methods presented in Table. 3.1. To ease the understanding of the different parts of the system, we gradually complexity in the following sections. Our objective here

being is to present the general architecture of the system illustrated in Fig. 3.6.

3.4.1 Genezik architecture

Genezik combines our global song rec-sys, described in 3.3, returning a set of songs from user input with a personalized playlist recommender system that mixes song owned by the users to deliver a coherent and personalized playlist. The global model is based on collaborative filtering as well as the usage of two graphs: a graph of songs created using content-based features and a network of playlists as explained in Sec. 3.3.

In Genezik, both of theses graphs are extracted in a distributed manner, uploaded from the clients and merged on the server as shown in the middle left of Fig. 3.6. The advantages of using divide and conquer architecture are numerous. First, server costs to extract, collect features and recommend playlists significantly drop as we perform the computations while the user listens to music. Second, even though online streaming services such as Spotify or iTunes are being enriched daily, songs dear to the user are still missing. Being able to make playlists from our collection of tracks that could include personally recorded tracks is a great feature that would require uploading songs in another cloud-only architecture. If all of the user's library is in the cloud, Genezik can still extract audio features in a distributed manner while streaming songs and recommend playlists using the computational power of the client. Finally, another advantage of our architecture is that our recommendation systems can evolve separately as they both communicate through a simple API.

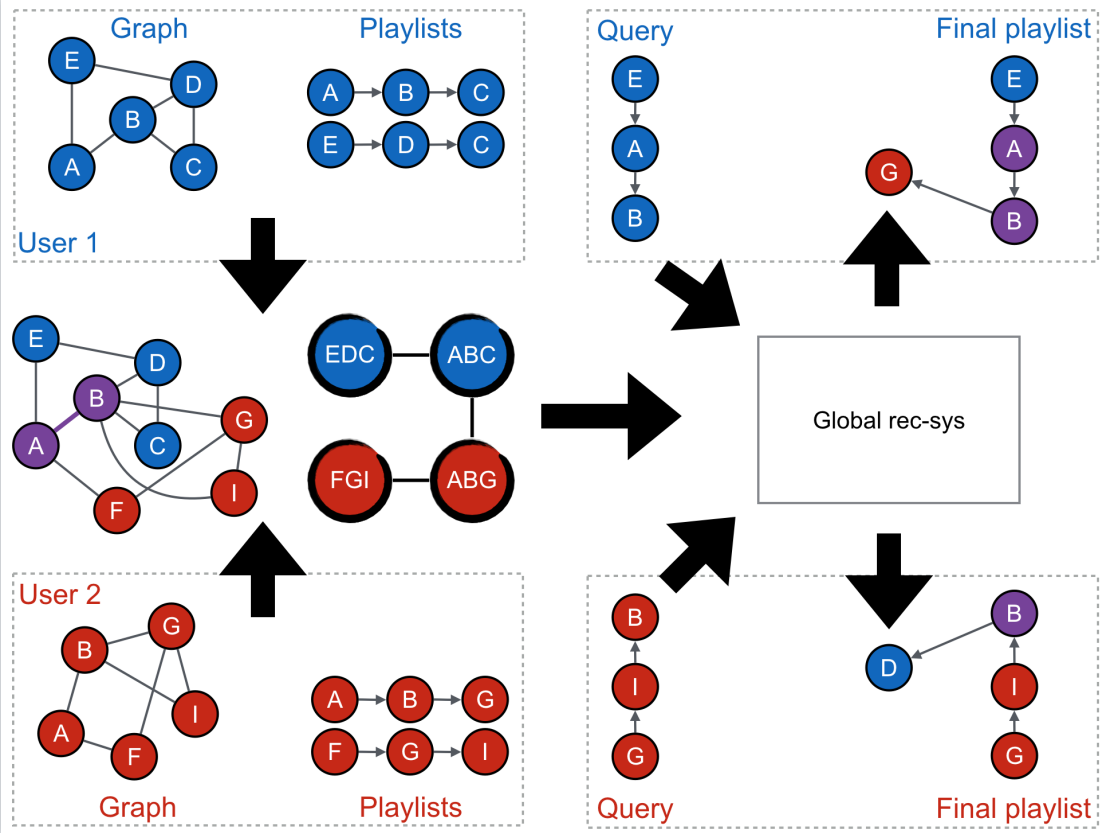


Figure 3.6 – **General architecture of the Genezik**. For each user 1 and 2, a personal graph of songs is created from their library and uploaded online as well as the previous playlists or the one generated by the local rec-sys inside the dashed line (top left and bottom left of the image).
We then merge these graphs of songs on the server and create an another graph of playlists where each node is a playlist (middle left). We use these two graphs as input to the global rec-sys (on the right). Upon user query, here a playlist made by traversing the local graph of a user, the server returns a fitting song that the user does not possess. The final stage of the pipeline, done by the local rec-sys, consists in finding the most appropriate position to incorporate the song according to the user's taste.

Moving on to the description of the local playlist recommender system, it also relies on a graph of songs, top left and bottom left of Fig. 3.6, that incorporates the extremely subjective taste of each user. This local graph will continuously be updated by the actions of users through implicit feedback, such as playing or skipping a song, and explicit feedback such as rating transitions between two songs. This last point is particularly important as to our knowledge, none of the existing systems propose to explicitly rate the transition between songs, encoded as an edge in a graph in our system. The local rec-sys thus progressively learns from the user listening habits to adapt the weights of the local graph.

Illustrated in Fig. 3.6 on the right of the graph of each user 1 and 2, our playlist recommendation algorithm generates a playlist by traversing the user graph as explained thoroughly in

Sec. 3.4.3. The final stage of our pipeline, on the right of Fig. 3.6, consists in enriching a locally recommended playlist with songs proposed by the global song rec-sys. The optimal position of the new songs in the playlist is determined by the client application making sure that it is perfectly adapted to the taste of our user.

3.4.2 Personalized graph of songs

In Sec. 3.3.5 we create a graph of songs using a collection of features mostly given by the Echonest service, a company specialized in feature extraction at large scale, on a given dataset. Starting from scratch with Genezik, the problem of data collection should be considered first as the quality of the service depends on it. More specifically, because Genezik inspects the user personal collection of music, we must find a way to uniquely identify the songs that users of the service have in common so we can merge them together on the server, as illustrated in the middle left of Fig. 3.6. Then, we must be able to extract audio features and create the graph of songs of each client.

Identifying tracks

While it may seem trivial at first, it is a challenging task as it is impossible to trust the user metadata, such as the artist name or the title of the song. Also, many versions of the same song coexist as music labels or DJs continuously propose new albums with edited, remixed, re-mastered versions of original songs. Finally, compression algorithms and softwares used to encode the song digitally also introduce complexity that we need to tackle to have a coherent corpus.

To solve this issue, Genezik mimics the data structure of the Echonest service by conceptually separating a song from a track. A Song object represents a meta-version of what users listen to without taking into account the specificity of its release here represented as a Track object with a specific album, duration, encoding. The process of uniquely identifying a track is called audio fingerprinting, reviewed in depth in [121]. Our Genezik application uses a customized version of the algorithm described in [122]. To associate similar tracks to the same song on the server, we use the algorithm described in [123] combined with heuristics to determine the correct metadata of each song.

Note that the current system could be improved with the integration of dedicated systems to identify songs such as AcousticId [4] combined with MusicBrainz [5], the free online Encyclopedia of song metadata that were not mature when the project started.

[4] https://acoustid.org/
[5] https://musicbrainz.org/

Feature extraction

To extract features from audio tracks, we first need to be able to decode compressed user files. Never actually considered in a research context where the audio is often given in raw WAVE format, this part is nevertheless crucial when implementing a real application as the audio codecs supported by the decoder directly impacts the user experience. The cross-platform solution retained in Genezik uses the C library FFmpeg [6] as it proposes an extensive collection of decoders and can be (painfully) deployed onto the clients.

Extracting features is also a synonym of blocking the user to enjoy the application right away as we need time to process his library initially. Once again, this aspect of the problem is not considered when developing a new recommender system but is the main limiting factor here. Indeed, to keep the processing under 1 second per file on a recent mobile phone, Genezik needs multiple passes on the user library.

In the first pass, we simply extract low-level features such as Chroma and MFCC in a multi-threaded manner. In short, Chroma or pitch class, maps the audio spectrum onto 12 bins, 12 distinct semitones of the musical octave. The chroma C thus represents all possible Cs, in whatever octave position [124]. Chroma features are interesting for similarity as humans perceive the same notes in different octaves as particularly similar. MFCC stands for Mel-frequency cepstral coefficients and represent short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. We use them as a indication of the timbre of instruments [125].

In a second time when the first graph of songs is created, we extract higher level audio features similar to the features presented in Tab. 3.2 using dedicated feature libraries such as Yaafe [126] or LibRosa [127]. Another trick employed by Genezik is to download features already computed by other clients from the server as soon as the track is formally identified on the server. Because a large proportion of users share the same popular tracks, downloading the complete features greatly increases the enjoyability of the user experience as a complete version of the graph can be created much faster.

Graph creation

The graph of songs $G = (V_G, E_G, W_G)$ has a similar definition to the one introduced in Sec. 3.3.5 so that both systems are coherent with their definition of song similarity. However, we want to emphasize here that the k nearest neighbors (kNN) cannot be computed in an exact manner (as in our global rec-sys) to be used practically as the number of exact computations grows quadratically with the number of songs. Instead, we must resort to using approximate nearest neighbors algorithms [128] to speed up calculations, implemented using the Flann library [129] in Genezik.

[6] https://ffmpeg.org/

The similarity score between each k closest songs given by Eq. 3.13 is stored as a property on the corresponding edge of the graph with other properties related to user feedback, described in the next section. To increase the quality of the connections between songs and clean the data, we also filter the edges with a similarity score too low to be considered as a smooth transition by humans. Principally developed as a set of heuristics, this threshold ϵ is adaptive to the number of songs the user possesses as well as his listening habits. The introduction of this minimum song-similarity threshold, formally defined as the ϵ-neighborhood graph, considerably increased the quality of the network as it can filter out connections from valid songs to ill-formed or short sounds excerpts, such as interludes, intros, outros or claps. Without this threshold, the topology of the graph on messy user collections leads to a small-world network where most of the songs are connected to these sounds increasing the number of heuristics needed to recommend clean playlists. More formally, the function to obtain the similarity weight between songs $j, j' \in V_G$ is given by:

$$\text{sim}(j, j') = \begin{cases} d & \text{if } d = \exp(-\left\|\mathbf{x_j} - \mathbf{x_{j'}}\right\|_1 / \sigma) < \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{3.14}$$

where $\mathbf{x}$ is the set of features described in Tab. 3.2, and $j' \in V_G$ is the k^{th} nearest neighbors of $j \in V_G$. The parameter σ acts as the scale parameter of the graph and is set to be the average distance of the k^{th} neighbors.

The similarity threshold ϵ is initially set as the mean distance between disliked transitions on the global graph of songs. However, note that the quality of the recommendations and thus the threshold are dependent on the popularity of the service for the global recommendation part or user feedback for local part.

In the case where the user does not have a sufficiently coherent library meaning there are simply not enough songs in total or per music style, the quality of the service is not optimal, and the threshold needs to be loosened to propose an exciting experience. Fortunately, if the user can stream music, gaps are filled with recommended songs instead of degraded transitions.

When new songs are added to the user's library, we compute their closest neighbors and attach them to the existing graph. However, when the graph changes too much, the graph of songs is entirely recomputed from the features in the background not to miss transitions between songs. This last use case brings many challenges notably for the persistence of the data. More specifically, how to store nodes and edges of the graph knowing that we must support new insertions, deletions and continuous modifications of the weights of the network. The application must also recover from crashes and always be able to generate playlists while the network is being modified in the background.

The solution we choose in Genezik relies on the usage of Sparksee[7], an embedded NoSQL Graph Database, designed from the ground up to support graphs. At the time of the writing, it was the solely embeddable database that could work without running a server on the client. In addition to easing the management of the graph, Sparksee proposes dedicated APIs to traverse the graph efficiently. We use them extensively for playlist recommendation as explained in Sec. 3.4.3.

User feedback

A real strength of Genezik is its extreme personalization to user taste. To do so, most of the user actions in the application such as play, skip are monitored and stored as node properties in the graph. In addition to the counters on the nodes, we also watch explicit feedbacks on the edges of the network as we show in Fig. 3.7. To do so, we designed the GUI (graphical user interface) of our app to be able to rate the transitions made in a recommended playlists as it represents the core principle of our playlist recommendation algorithm described in Sec. 3.4.3.

	Node	Edge
Base features	Audio Social Metadata	Song similarity
Implicit	Play count Repeat song Skip Cluster id	Play count Skip
Explicit	Play count Swap song Add/ remove in playlist Mood label	Rating

Figure 3.7 – **Principal node and edge features used in Genezik**. Nodes base features are described in Tab. 3.2. As implicit features for nodes, we have the number of play counts of songs in generated playlists. The number of times a song has been repeated, skipped as well as the cluster ID when we perform a community detection on the graph. On the edges, we have the number of times a transition has been played or skipped. Our explicit features for nodes are the explicit play count (user double-clicked on the song), the number of times this song was swapped for another, added or removed in generated playlists.

Let us now detail how user feedback changes the graph of songs. We define as $\mathbf{im_j}$, the implicit feature vector, $\mathbf{ex_j}$ the explicit feature vector associated with node $j \in V_G$. Similarly we define

[7] http://www.sparsity-technologies.com/

$\mathbf{im_{jj'}}$ and $\mathbf{ex_{jj'}}$ the edge feature vectors between nodes j and j' described in Fig. 3.7. We further define four feature importance vectors, $\mathbf{n_{im}}$, $\mathbf{n_{ex}}$ that weight the importance of each node feature category, and $\mathbf{e_{im}}$, $\mathbf{e_{ex}}$ the importance of edge feature category.

The final edge weight $w_{jj'} \in W_G$ between two nodes j and j' is given by:

$$w_{jj'} = \gamma_s \text{sim}(j, j') + \gamma_i \left(\mathbf{n_{im}}^\mathsf{T} \cdot \left(\frac{\mathbf{im_j} + \mathbf{im_{j'}}}{2} \right) + \mathbf{e_{im}}^\mathsf{T} \cdot \mathbf{im_{jj'}} \right) + \gamma_e \left(\mathbf{n_{ex}}^\mathsf{T} \cdot \left(\frac{\mathbf{ex_j} + \mathbf{ex_{j'}}}{2} \right) + \mathbf{e_{ex}}^\mathsf{T} \cdot \mathbf{ex_{jj'}} \right) \quad (3.15)$$

where γ_s weights the importance of musical similarity, γ_i the importance of the implicit feedback and γ_e the explicit feedback. We require that $\gamma_s + \gamma_i + \gamma_e = 1$, $\gamma_s > 0$, $\gamma_i > 0$, $\gamma_e > 0$.

We see that Eq. 3.15 takes into account musical similarity as well as user feedback. To be as fast as possible, Genezik directly modifies the properties of the graph and recompute edges weights when the user interacts with the application. For instance, when a user plays a song, we recompute the weights of all our edges. This technique allows distributing the computations while the user is listening to the music instead of when the app is generating playlists. It radically changes the experience as the user instantaneously see songs appearing instead of having to wait for the end of the playlist generation, particularly on a mobile phone.

In addition to being original, our GUI, shown in Fig. 3.8, also encourages the user to listen actively to music by proposing some gamification schemes [130] to rate transitions, add new songs or correct metadata in his library. The introduction of reputation points showing the user's expertise in music recommendation is critical as Genezik needs implicit and explicit feedback to adapt to the taste of users. By encouraging an active immersion in the recommended playlist with ratings and reputation, the user feels more engaged and quickly tune his model while contributing to the global taste of all users when the individual graphs are patched together.

Lastly, we also propose an "expert view", where the user can move a song across the lane that represent user-defined categories or moods. We restrict the placement of a song to only one mood at a time as it captures the current feeling of the user. We use it to adapt our playlist recommendation as we show in Sec. 3.4.3. Because it is very easy to switch lanes, we also notice a greater user interaction with the app as the user feels the need to move the songs out of the gray (default) mood. The special purple lane *Discovery* on top is dedicated to the recommendations from the global rec-sys. When the user is connected and when recommendations are available, the playlist forks between local and global recommendations. If the user can stream music, with the association of his Spotify account (for instance), the transition at the end of the current song jumps to the recommended songs (an edge is drawn with a song in the discovery lane). In other cases, we only display the recommendation, but the playlists continues normally with local songs.

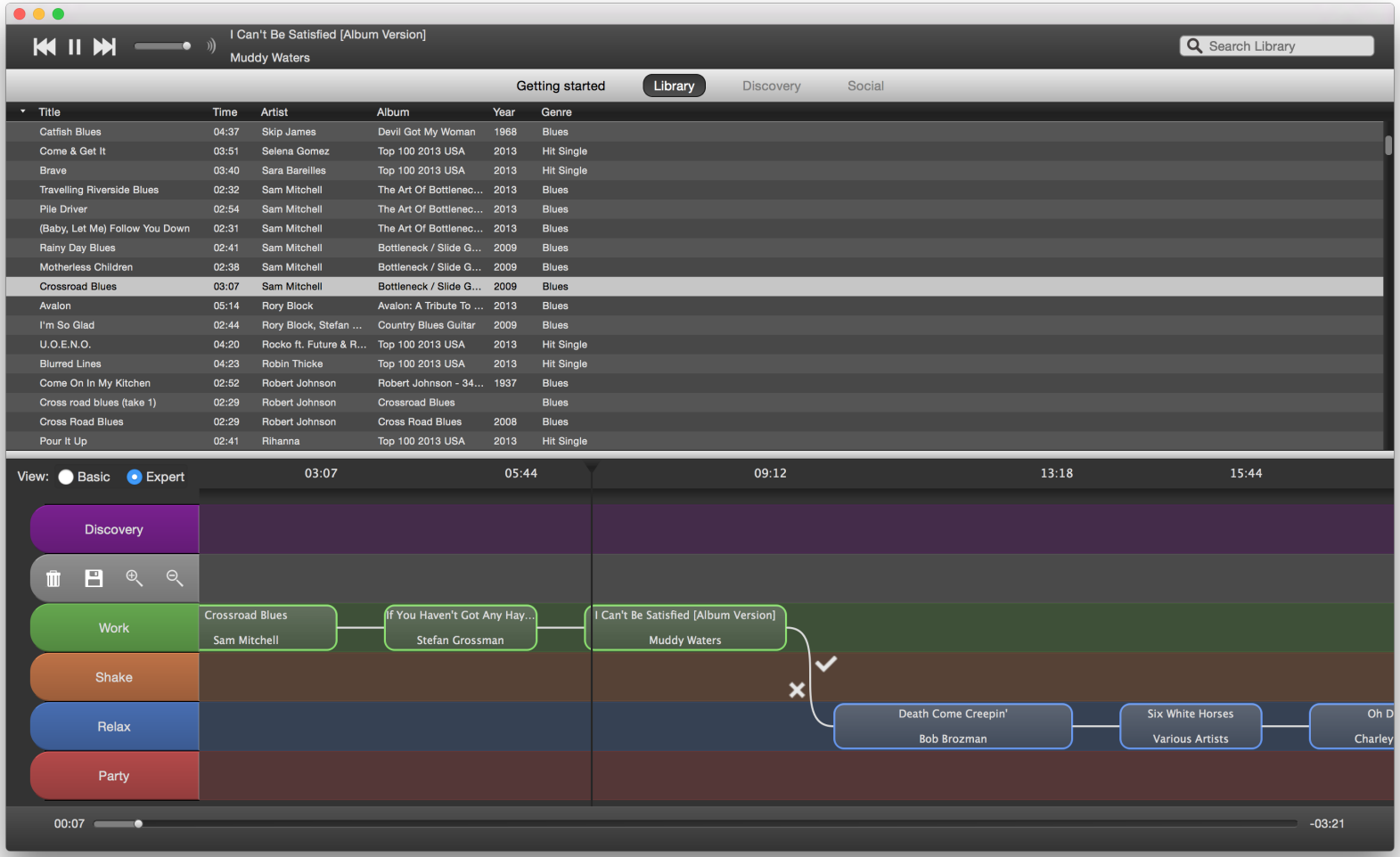


Figure 3.8 – **Genezik GUI.** The recommended playlist is displayed on the bottom of the screen in the timeline of songs. The transitions between songs of the playlists can be rated positively or negatively by clicking on the dedicated icons. Once a transition is rated, it disappears for the current playlist to give a sleeker experience. The different colored lanes correspond to user-defined moods that help classify and organize user library while encouraging user interactions within the application.

3.4.3 Playlist recommendation

The playlist recommendation algorithm in Genezik is based on a customized traversal of the graph of songs. In theory, traversing the graph according to following the shortest path between songs would always lead to the smoothest playlist as we encode the musical similarity as well as how the user likes transitions between songs. In practice, the exploration of the graph of songs by following the strongest weights may be too simple to yield coherent playlists.

Before describing the several recommendation schemes implemented in Genezik, we wish to underline the novelty of our approach with the introduction of four different manners to generate playlists that do not exist in competing systems such as iTunes, Spotify, Pandora or Google music.

In all of our recommendation schemes, the end of the playlist is systematically regenerated as soon as the user dislikes a transition. We also delete the edge from the graph of songs to prevent making the same mistake again. While it may seem harsh, we notice that users explicitly dislike when the transition is unnatural. Removing the edge prevents to bother the user once again.

Lead Me On

Genezik activates its default mode *Lead Me On* when the user chooses a song from his library and drops it into the timeline as the starting point of the playlist. The strategy here is to recommend songs by exploring the neighborhood of the given song while ordering them as smoothly as possible until the maximum playlist length of 15 songs or 60 minutes (default) is reached. To ease the comprehension, the algorithm, called *snail walker*, is illustrated in Fig. 3.9. Dashed lines represent unvisited transitions while arrows represent the path taken by our walker to generate the playlist. The starting point or seed song s_0 is colored in black, the endpoint e in red.

Because our algorithm has to run on mobile phones and offer an immersive experience, we choose to output a song one at a time instead of having to wait for the whole playlist creation. This choice also limits the number of computations and forces us to choose the next song depending on the past without looking one or two steps forward.

Let us define the playlist $P_t = (\{s_0, s_1, \ldots, s_t\})$ as an ordered set where s_t is a song in the playlist at position t. The next song s_{t+1} is chosen from the set of songs evaluated at step t, denoted S_{t+1}. For two nodes u, v we denote by $u \sim v$, the connection between them on the graph.

$$S_{t+1} = \min_{k \in \mathbb{N}} k, \text{ subject to } X_k \neq \emptyset, \tag{3.16}$$

$$X_k = \{u \in V_G \mid u \sim s_t \text{ and } u \sim s_k \text{ and } u \notin P_t\} \tag{3.17}$$

Let $p_t(i)$ be the probability of choosing the next song from the set S_{t+1}. $I_{t+1} = \{i \mid u_i \in S_{t+1}\}$, is the set of indices of songs belonging to S_{t+1}. We have:

$$p_t(i) = \frac{w_{t,i}}{\sum_{j \in I_{t+1}} w_{t,j}} \quad \forall\, i \in I_{t+1} \tag{3.18}$$

In simpler terms, the algorithm tries to minimize the divergence from the seed song by choosing the next song to put in the playlist s_{t+1}, so that it stays connected to the seed song s_0. In the case no edge exists, it tries to connect to s_1 and the neighborhood of s_0. If once again no edge exists, the snail walker will try to connect song s_{t+1} to s_2 and s_1's neighbors. This recursive procedure stops when the maximum number of songs in reached.

In the default case, when many songs are competing or when no edge exists between the current set that is evaluated and the previous set X_k, the node is chosen according to the probability of edge weights $p(i)$ rooted at the current node. We remove already visited nodes and only compute the probability of jumping to available nodes at the next step. Illustrated at node 5 and 7 in Fig. 3.9, the algorithm chooses the last node exactly like a random-walker [131], the probability of selecting this node is 40% from node 7.

Note that at step $t = 1$, we always choose a transition with a null play count if possible, as our goal is to gather implicit and explicit feedback from the user as quickly as possible to improve our recommendation algorithm.

Finally, once the playlist is generated, we use the global graph of songs collectively created by all users to propose recommendations in the discovery lane. Here, our recommendation strategy consists in finding on the server a clique of 3 nodes, a triplet, where 2 nodes are consecutive on the local playlist. The new recommended song is guaranteed to be adjacent to the two others and thus should match the style of the playlist.

Ride and Roll

The second recommendation strategy implemented Genezik is called *Ride and Roll.* To activate this mode, the user inserts a song in an already generated Lead Me On playlist. The algorithm here finds the shortest path between the song placed before the insertion point and new song dropped in the timeline. Based on Dijkstra's shortest path algorithm [132], we also filter songs that already played in the current playlist among various heuristics.

Once common issue that we address in this playlist mode is the length of the playlist. Indeed, the shortest path algorithm aims at reaching the end point as fast as possible, minimizing the number of hops. When the graph is well connected or when the start and end point are similar, the playlist length tends to be too short with around 4 or 5 songs. In this case, we fill the playlist by finding the best triplets between consecutive songs. In the rare occasions where

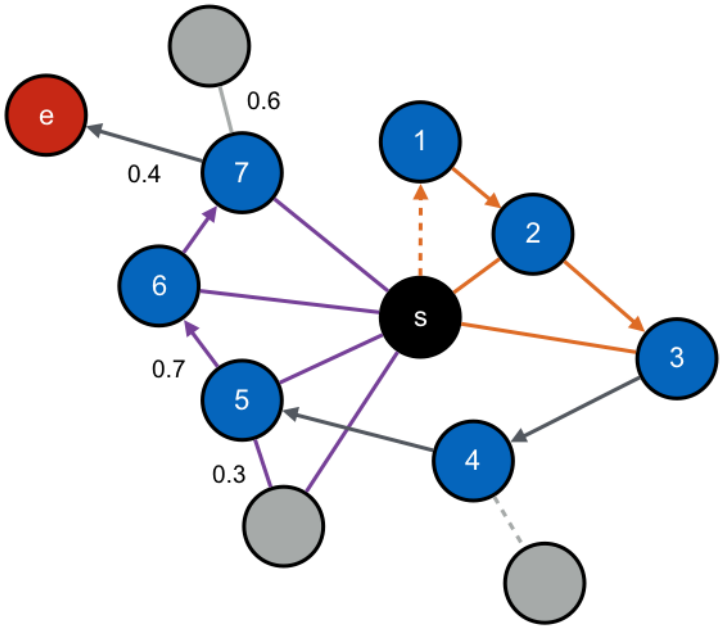


Figure 3.9 – **Lead Me On, the default playlist recommendation mode in Genezik**. Dashed lines represent unvisited transitions while arrows represent the path taken by our snail walker to generate the playlist. The starting point or seed song e is colored in black, the endpoint e in red. The algorithm is described in Sec. 3.4.3. The path created from node s to e resembles a snail shell as it tries to order neighbors of s as smoothly as possible.

the playlist is too long, the algorithm tries to cut the playlist by folding triplets of songs.

Finally, it happens that user library forms disconnected clusters of songs and that no path exists between two songs. In that case, we chain songs from the same mood even though they do not connect on the graph. If songs are not labeled with moods, we apply the same technique with genres, artists, and albums. In the case the user listens to the transition without giving a negative feedback such as “skip” or “dislike”, we permanently create the edge. As a result, we progressively connect the user library. However, if none of these techniques work and that the user can stream music, we must resort to performing the same Ride and Roll algorithm on the global graph of songs in the cloud between the start and end songs.

Traversing the graph on the cloud is computationally expensive if it is large. To drastically speed up the throughput of our service, we employ approximate shortest paths methods based on a hierarchy of coarsened graphs. Forming of a multilevel graph shaped like a tree, it allows to quickly prunes unneeded parts of the original network and thus increases the throughput. More information can be found in [133].

In The Mood

Genezik also proposes to generate playlists by just clicking on a mood button when we open the expert view, see Fig. 3.8. Here, we randomly choose a song labeled in the mood as starting point and use the Lead Me On method, described in Sec., to generate the playlist. The algorithm is slightly modified to first choose the songs with strongest weights in the given mood before the rest.

Manually moving songs from the default lane to the colored mood lanes can be tedious if the user library is huge. To help the user, Genezik applies the label propagation algorithm described previously for the Star Wars corpus (see Sec. 2.5.2) and propagates known labels assign unlabeled songs to moods. The label propagation algorithm needs a sufficient number of known moods to be able to predict the missing values accurately. To speed up the convergence, we first find the communities in the graph [115] and restrict the label propagation algorithm to the set of nodes of a community when the number of tagged songs is above 50%. This trick allows filling the labels in a controlled way instead of propagating few values over unlabeled areas of the graph. To indicate that the song has been automatically put in a lane, we draw the border of the rounded rectangle representing a song in a dashed line.

Take Me On A Tour

The last mode for playlist generation in Genezik is experimental. Triggered when the user drops 3 or 4 songs simultaneously in the timeline, *Take Me On A Tour* tries to find the smoothest playlist that stops by the songs chosen by the user. Implemented as successive Ride And Rolls, the difficulty here is to find an optimal way to order the key points so that the playlist stays coherent. For instance, let us take two Rock songs and one Classical, the most logical order would be chain the two Rock songs together and not alternate Rock, Classical, and Rock.

To be as coherent as possible, we thus order key points by mood and genre. The best transitions between moods and genre are learned by computing the likelihood of reaching a given genre from the edges of the graph.

If none of these rules are applicable, we greedily compute local Ride And Rolls between a starting song with all others and select the shortest path. We iterate with the second song until all key points are exhausted. This solution is used as last resort as it is computationally expensive and not applicable in a mobile context. Future work could include a relaxation of Traveling Salesman algorithm [134] to include all key points while incorporating playlist duration constraints.

For the discovery part, we fully leverage the global recommender system introduced in Sec. 3.3 as we give as input key points selected by the user. The returned unordered list of songs is then further processed to find the best triplets that we could insert in our local playlist similarly to what is done for the *Lead Me On* mode.

Validation

The experimental validation of Genezik playlist algorithm was made by a panel of users as academic metrics to measure the playlist purity or coherence does not truly matter in a real-life situation: only user's engagement metrics do. To gather feedbacks, we anonymously logged all the actions performed by the user to generate playlists such as the number of songs dropped in the timeline as well as the average library size to cluster user behavior and adapt the algorithm for each category. In addition, we designed a beta-tester panel in the application to be able hand-tune the different factors of the algorithm. Theses factors such as the artist alternation or likelihood to jump to another genre were then gathered to give the best compromise when the app was released.

Saving playlists

Genezik approach for saving playlists is radically different from other systems: instead of freezing a generated playlist as a static list of songs, we only save the songs selected by the user for its creation. The playlist is thus always regenerated around the key points, called *play-points*, defining the soul of the playlist. This technique alleviates the common pain of shuffling songs or editing a playlist that has been listened too many times by our users. In case the user wants Genezik to behave like a regular music player application, we nevertheless propose to "lock" the playlist. To give more flexibility to the system, we also allow any song to be promoted to a key point.

Subscribing to playlists made by others is the principal recommendation strategy of many services such as the Art of the Mix or Spotify. The playlists are in read-only mode, and only their creators can add, remove or reorder songs. Another advantage of play-points concerns the social part of Genezik. Instead of sharing playlists, we now propose to share play-points and let the playlist be generated differently onto each client. This method allows a better adaptation to the taste of each user while keeping the original intent of the playlist creator and his skill in handpicking the play points.

3.5 Improving models and reproducibility in MIR

All models and application in Music Information Retrieval community require a corpus of audio records that is analyzed in an automated manner to classify, separate, transcript or generate sounds. A common issue that many researchers face in this field is the lack of data. Indeed, the most common dataset in MIR is GTZAN gathered by G. Tzanetakis and P. Cook in [135]. This dataset of 1000 30-seconds excerpts has been collected by hand from the main author's personal library. After being used for more than a decade, the illegally distributed GTZAN tracks have achieved their limits [136].

Another popular dataset in MIR came from the now offline TagATune game, the dataset called

MagnaTagATune [8] has also been used extensively in the community [137, 138]. The limit of this dataset comes from its static nature and its size of 5405 mp3 and 230 artists[9] which is not relevant anymore in this era of big data.

Previously mentioned in Sec. 3.3.5, the Million Song Dataset (MSD) [114] is also famous in the community. Extracted using the online Echonest service[10], it is composed of audio features such as pitch, timbre, key as well as metadata for one million contemporary and popular tracks. Moreover, the community has contributed to a cluster of datasets around the MSD with cover songs, lyrics, user data or labels. While the scale fits with today's algorithms, the critical issue with this dataset is the lack of audio files (copyright issues).

A possible solution to alleviate this problem is to download 30-seconds previews from a third-party service and compute our own features. In addition to being complex to maintain, the limitation of the service and the lack of control on which part the preview comes from makes this solution impractical. Worst, the Echonest identifiers for each song and the features of the MSD dataset changed with the successive updates of the Echonest service. It is now impossible to retrieve the exact songs and features of the original dataset, destroying all chances of an evolution of this dataset. Researchers are thus forced only to use the old Echonest music features, also subject to intellectual property or create a new version of a dataset as we did with the Art of the Mix revamped dataset in Sec. 3.3.5.

Finally, the dataset gathered by the usage of Genezik, unfortunately, has the same limitations of the Echonest features as we can not legally distribute audio files. The access to Montreux Jazz Festival archive data where Genezik was used is even more restrictive as the streaming of audio files for research is subject to a Non-Disclosure Agreement. The average duration of authors' rights is 70 years after the author's death, the corpus of the current Montreux Jazz should thus be entirely free in around 150 years!

3.5.1 Introducing the Free Music Archive dataset

We now see clearly the utility of a large, freely accessible and legal dataset of raw audio files actively maintained by a rich community of users. In this regard, the Free Music Archive [11] appears as the perfect candidate to solves all the issues presented so far [139]. With more than 77,000 high-quality audio files (with full metadata) including more than 14,500 tracks mapped to the latest Echonest audio features cross-referenced with all majors streaming vendors (Spotify, Deezer), the Free Music Archive is the biggest legal dataset to date. Besides, the FMA also tracks the number of play counts for each track, making the dataset suitable for all collaborative filtering methods.

The gathering of the FMA dataset involves using both the API given by the website to collect

[8] Downloadable at http://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset
[9] See stats at https://musicmachinery.com/2009/04/01/magnatagatune-a-new-research-data-set-for-mir/
[10] http://the.echonest.com/
[11] http://freemusicarchive.org/

Genre	Count
Electronic	2962
Rock	2385
Pop	1636
Hip-Hop	1059
Folk	1025
Punk	882
Indie-Rock	601
Jazz	588
Old-Time / Historic	460
Psych-Rock	456
International	431
Classical	408
Chiptune	254
Blues	245
Psych-Folk	245
Post-Punk	222
Metal	191
Soundtrack	172
Trip-Hop	171
Post-Rock	118

Table 3.5 – **Repartition of the most popular curated genres in the Free Music Archive.** We remove low-quality categories that would not be representative of a classic genre prediction task such as: Noise, Lo-Fi, Experimental or Avant-Garde, Audio Collage, etc.

all the metadata associated with a song as well as using a web scraper to download the actual audio file. The link to download audio files is unfortunately not included in the metadata associated with the song, our only way to extract it is thus to browse the web page of each track and read the markup language.

To give the reader an overview of the possibilities offered by this dataset, we tackle the difficult genre prediction problem often used in MIR contests[12]. To do so, we extract the features presented in Sec. 3.3.5 and split the dataset into a standard 70% train, 30% test set. Note that because we use the Echonest features to compare with other datasets, only a subset of the FMA archive (14,511 songs) could be employed. We report the results of the classification task by a k nearest neighbor classifier and a support vector classifier using the L1 or L2 norm [140] in Tab. 3.6. Additionally, we display the repartition of the genres utilized for the classification in Tab. 3.5.

These preliminary results should be further refined in future work where the dataset will hopefully be used by other researchers to develop and compare algorithms similarly to what computer vision and pattern recognition scientists do with the MNIST dataset [141].

[12] http://www.music-ir.org/mirex/wiki/MIREX_HOME

	L1	L2
Knn-5	**0.473**	0.429
Knn-10	0.477	0.438
SVC-linear	0.468	0.304
Random	0.05	0.05

Table 3.6 – **Genre prediction results for the FMA archive.** Results are averaged over 10 runs. The best result is given by k nearest neighbor classifier with $k = 5$ using the $L1$ norm to compute the distances between features. We show the random classification score as a baseline.

3.6 Discussion

In this chapter, we have shown how networks could be used to recommend music adapted to user taste by tuning similarity relationships between songs using user implicit and explicit feedback [9]. We have seen how to integrate local playlist recommendations with a cloud-based service based on matrix factorization with the usage of graphs of playlists and songs collectively collected from users [10]. We also presented new playlist recommendations schemes based on the traversal of a graph of songs continuously updated to fit user's taste in music. Finally, we have demonstrated the novelty and worth of saving *play-points* instead of static playlists to have variety and adaptability to one's taste while keeping the original intent and skill of the play-points creator.

Let us underline that building a real-world music recommendation application is very challenging. Like any app, it involves designing the architecture of the app as well as its graphical user interface. The implementation needs to be both maintainable and yet efficient in the choice of the algorithms for the feature extraction, graph creation or traversal for instance. The principal difficulty came from the deployment of low-level audio decoding and features extraction components on multiple heterogeneous platforms (Windows vs. iPhone), as we have tried to have a single codebase for maintainability.

The development of Genezik has led to the creation of two large-scale datasets for MIR community. First, we have revamped the Art of the Mix dataset to include all possible features we could gather from online services (features, lyrics, social tags). Second, we have built the Free Music Archive dataset: the largest legal dataset of raw audio files with associated metadata dedicated to MIR practitioners. The goal of this dataset being to gather the community around a common ground truth to improve the reproducibility of algorithms and the comparison of results similarly to what the image processing researchers do with the MNINST dataset.

More generally, as we have seen in this chapter, the development of Genezik closely match most of the aspects of network science with data-mining, feature extraction, similarity measures, graph creation, weight learning, graph traversals or network visualizations as we show in Fig. 3.10. An interesting aspect of music recommendation that we have not tackled here is the contextual adaptation of our algorithms to the time of day, weather or geo-localization [142]. With the deployment of Genezik as a mobile application, it would be interesting to gener-

ate playlists depending on the heart-rate of a jogger or a combination of sensors such as geo-localization and accelerometer for car or bus playlists recommendation. Future work could also include the possibility of capturing the evolution of taste in music through time to improve our recommender system. It would mean that we could detect volatile mood swings over a few days or radical changes of the appreciation of a music genre over a few months for instance and thus adapt the recommendation in consequence.

This reflection around the possible evolution of taste using networks is clearly not limited to particular applications linked to music as it relates to the dynamical activity of users over graphs.

This ascertainment led to a turning point in my research as I started to investigate how to encode in a scalable manner both the graph and user activity to go beyond existing graph methods. As we will see in the following chapters, this combination proves to be invaluable as it opens new perspectives in various research domain such as neuroscience, knowledge mining or social networks that were previously impossible to capture with a separate analysis of the signal on the one hand and the graph on the other.

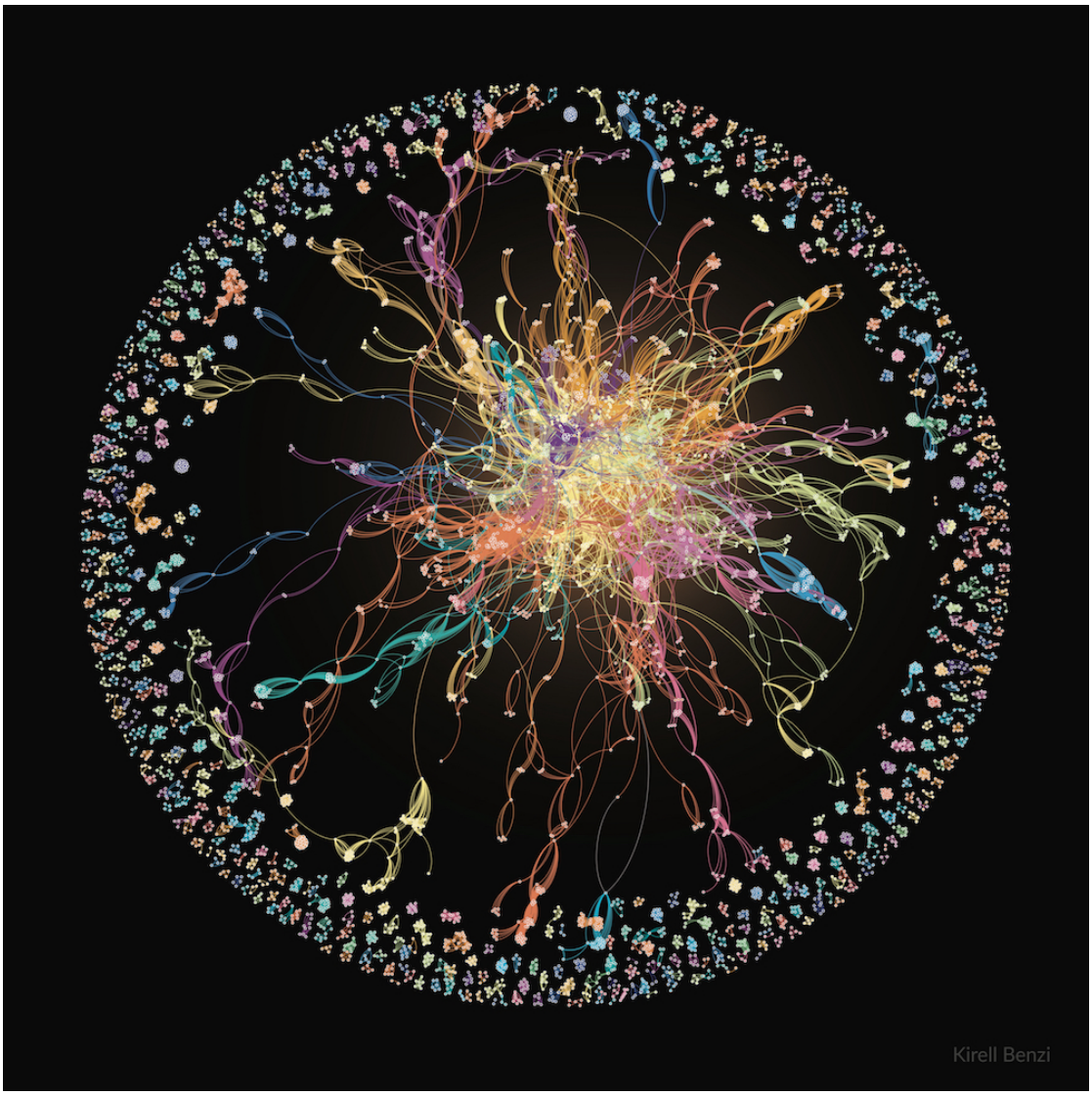


Figure 3.10 – **When the Music is Good - 2013.** The Montreux Jazz Festival is one of the most famous music festivals in the world. Starting with Jazz, it has progressively expanded to other genres of music and as always attracted world-class performers such as Queen, B.B. King, Prince, or Santana. This network shows with whom musicians of the festival play with, revealing two different categories of artists. At the border of the ring, we have the artists who only perform with their band, forming many disconnected communities. On the opposite, those who jam with everyone, the stars of the festival, are well-connected and are naturally located in the center of the ring. One of the brightest stars was George Duke, the champion of appearances at the festival with 53 concerts. Copyright Kirell Benzi.

4 Exploring dynamical processes over networks

Building on the previous challenges brought by the development of Genezik, we propose in this chapter to broaden the field of application of network science by studying how the activity over the nodes (or edges) of a network can be leveraged together with the graph [8].

The analysis of dynamical processes taking place on a network is a typical use-case of this combination with applications in various domains such as neuroscience [143, 144], the study of epidemics in physics [145, 146] and rumor spreading in social networks [147]. In these examples, some quantity or state (such as activity, information or congestion) diffuses over a network, as illustrated in Fig. 4.1. Two properties define activity patterns formed by these dynamical phenomena or processes: i) their localization both in space and time and ii) the way they spread, always propagating through the neighborhood over time. In the following, we refer to those particular processes as causal processes. Note that the concept of space here englobes physical space among others and should be understood as space onto which the graph lives as we will see in our applications in Sec. 4.3. A *causal process* on a graph is thus a particular type of dynamical process that models a recursive physical phenomenon propagating and spreading from a node to its neighbors in successive time steps.

Besides, in many applications such patterns appear regularly on the network in the same locations, possibly with some variations. The repetition of the dynamics offers a chance to understand the underlying process better, helping us to seek the cause of the spreading as well as to anticipate or forecasting possible future spreads of a pattern using historical data.

In the present work, we introduce a novel and natural method designed to track recurring patterns of activity induced by causal processes on graphs. It relies on the *causal multilayer graph* (CMG), a particular kind of multilayer graph designed to follow the propagation of events in successive time steps (see the following section). From the causal multilayer graph and the data attached to its nodes, we extract dynamic activation components, subgraphs of the CMG representing patterns of activity. We then classify and analyze these patterns to reveal global trends and insights on several applications showing that our framework applies to a large class of problems.

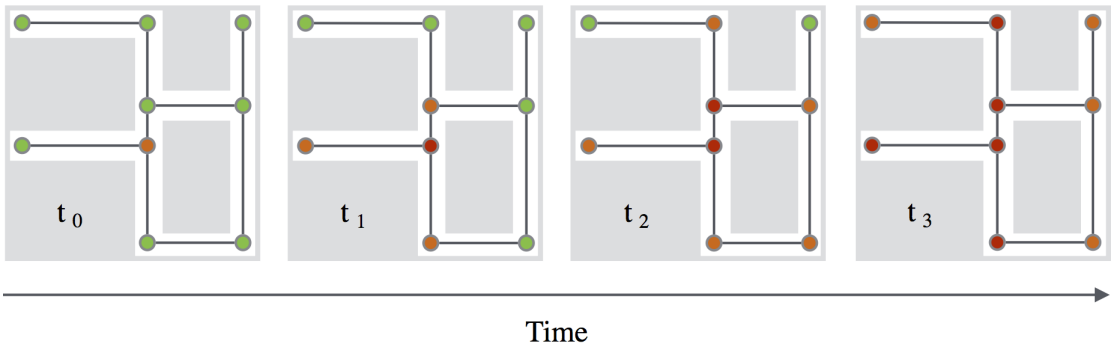


Figure 4.1 – **Evolution of a congestion pattern in a localized area of a city represented as a graph.** Each intersection is associated with a node, and each road is mapped to an edge. The color of each node represents the concentration of vehicles at this particular intersection (from green, fluid, to red, congested). Different snapshots of the same area show the evolution of the congestion through time. The congestion starts from a single node at time t_0 and can only spread over the adjacent intersections through time. Over the days, thousands of patterns can be analyzed to extract valuable insights such as how patterns spread, how long they last or what is the variability of the diffusion for instance. Learning the particular characteristics of congestion patterns and classifying them would be very beneficial to drivers as appropriate measures could be taken by the authorities as soon as congestion appears.

Note that the idea of pattern detection on graphs has some similarity with the temporal motif detection of [148] and the temporal subgraph isomorphism of [149]. However, motifs are restricted to be subgraphs of a few nodes, typically 2 or 3 (due to the NP-hard graph matching process), while our method can deal with patterns of any size. Our patterns can be considered as mesoscale structures within temporal graphs as described in [150]. These structures seem important for understanding dynamical activity in temporal networks but are not well explored yet (except for graph communities). We present new results in this direction, as we show in our applications.

Although there has been an enormous amount of research on multilayer networks in the past few years, scalable data-driven methods dedicated to the analysis of dynamic data evolving on multilayer graphs are still lacking [151, 150]. On the contrary, our method based on large-scale graph analytics frameworks [152, 52] scales linearly with the number of nodes and time-steps making it possible to handle billions of nodes by leveraging today's multicore architecture. Besides, it is simple to tune as it mainly depends on a single parameter and is very flexible as it supports directed, weighted and dynamic graphs.

We divide this chapter into three main parts. First, we describe the unique structure of the causal multilayer graph (CMG) and the causal multilayer graph of activity containing the dynamical patterns. We then propose a method to compare and cluster/classify activity patterns that we call dynamic activated components. The second major part of this chapter is dedicated to the application of our framework to four different real-world problems. Similarly

to the early example of Fig. 4.1, we first analyze flux of pedestrians and congestion patterns in a train station. We then come back to music recommender systems in our second example by showing special "mood" patterns that could be used to extend the system presented in the previous chapter.

The third application tackles the challenges of neuroscience by proposing to track brain region interactions in resting state by combing both functional MRI and diffusion MRI. In our last application, we extract dynamic patterns of activity in a social network and compare our approach to the work of De Domenico *et al.* on rumor spreading in [147].

The final part of this chapter relates to the implementation of the CMG of activity to handle large datasets alongside its generalization to dynamic graphs. We conclude with a discussion around the usage of network science to study causal events and on the difficulty of to gather dynamic graph datasets.

Our main contributions in this chapter can be summarized as follows:

- the introduction of a new concept merging data and graph altogether;
- a mathematical definition of the causal multilayer graph linking to previous theoretical works on multilayer networks;
- a large-scale practical implementation capable of handling billions on nodes on commodity hardware;
- a generalization of our method to dynamic directed weighted graphs;
- the application of our framework to four different real-world applications expanding our knowledge of dynamical processes on networks.

As part of the open science movement, we release the code and data related to this work to the community under the GPL v2 license on the laboratory's Github account [153].

4.1 Tracking dynamic activity patterns

To track dynamical activity in real data in a computationally efficient way, we need to introduce several mathematical objects and divide the method into several steps. We start by presenting the CMG as a conceptual object to guide the reader and help to make connections to previous works on multilayer networks. However, in the actual implementation, the construction of the complete CMG is avoided. Instead, we reduce the CMG to small parts by combining the signal describing the activity on the network and the network itself. In this section, we focus on the description of the concept and leave the technical details of the implementation in for the last part of the chapter 4.4. For clarity, we also restrict the presentation to a static graph and delay to Sec. 4.4.2 the introduction of a generalized version of our method that takes into account the possible evolution of the network through time.

Several notations are introduced throughout this section. To help the reader, they are summarized in Table 4.1.

G	Static graph
W, w_{ij}	Weight matrix of G and entry i, j of W
V_G, E_G	Set of vertices and edges of G
K	Causal multilayer graph (CMG)
N, T	Number of nodes per layer and nb of layers of the CMG
H	Causal multilayer graph of activity
S, M	Signal and mask matrices
L_t	Layer t of the CMG
Ω	Set of intra-layer edges
Ω_x	Set of inter-layer edges between neighbors on the layer
Ω_{xs}	Set of inter-layer edges being self-edges
μ	Threshold value to obtain the binary mask M from S

Table 4.1 – Notations for the different mathematical objects.

4.1.1 The causal multilayer graph

Multilayer, multislice or multiplex networks and the applications they model are a topic of great interest with a relatively large literature [154, 155, 156]. A multilayer graph is made of layers (distinct subgraphs) bound together by *inter-layer* edges. Different rules exist for building multilayer graphs. Our causal multilayer graph belongs to the family of temporal graphs [157], however with a particular way of connecting layers to account for the causality. To our knowledge, the configuration we propose has only been used in a recent theoretical study on the spreading of an epidemic [158]. However, in that reference, the purpose is purely theoretical. It simplifies the mathematical expressions and allows for new insights on the dynamics of epidemics on temporal networks. In our case, the goal is application oriented: working with this structure leads to a fast method able to track dynamic activities within a (possibly changing) network.

To construct the CMG two elements are needed. First, a graph that encodes the structural connectivity between the vertices onto which the time-series is defined. Here, we refer to this network as the spatial graph $G = (V_G, E_G)$. For the sake of simplicity, we assume G to be unweighted and undirected although it is possible to apply our method to weighted and directed graphs. V_G is the set of nodes (vertices) with $|V_G| = N$, $E_G = \{(i, j) \,|\, i, j \in V_G\}$ is the set of edges. Second, a matrix S of temporal signals made of N time-series of length T, one per vertex of G, must be given.

4.1.2 Definition

The causal multilayer graph, illustrated in Fig. 4.2, is made of layers $\{L_0, L_1, L_2, \cdots, L_{T-1}\}$, each one being a distinct copy of the spatial graph G associated to one time-step $t \in [0, \cdots, T-1]$. We denote by i_t the vertex of layer L_t associated to vertex i on G. Since each layer is a copy of G, the set of *intra-layer* connections in the causal multilayer graph connecting vertices within each layer is the set $\Omega = \{(i_t, j_t, w_{ij}) \,|\, i, j \in V_G,\, t \in [0, \cdots, T-1]\}$, with w_{ij} being the weight of edge (i, j).

To capture the spreading of an event on the graph on successive time steps, each vertex i_t is also connected to its neighbors on G at time step $t+1$. This set of inter-layer edges is denoted $\Omega_x = \{(i_t, j_{t+1}, 1) \,|\, i, j \in V_G,\, t \in [0, \cdots, T-2]\}$. The weight value for the inter-layer edges may be set to an arbitrary value. To simplify and to have an equal treatment of the spatial and temporal dimensions it is here set to 1.

Optionally, to follow the activity of the same vertex through time each vertex i_t at L_t is also connected to itself, i_{t+1} at L_{t+1} as it is done in temporal graphs [157]. This set of temporal self-edges is denoted $\Omega_{xs} = \{(i_t, i_{t+1}, 1) \,|\, i \in V_G,\, t \in [0, \cdots, T-2]\}$.

In total, the causal multilayer graph $K = (V_K, E_K)$ is composed of $V_K = N \times T$ vertices and of the union of intra-layer, inter-layer and self-edges:

$$E_K = \Omega \cup \Omega_x \cup \Omega_{xs}. \tag{4.1}$$

Remark that in the presented applications, intra-layer edges (the set Ω) are dropped to better capture the propagation of events on the graph and to make clearer visualizations. In the following, E_K is the set:

$$E_K = \Omega_x \cup \Omega_{xs}. \tag{4.2}$$

Relation to previous work and alternative definitions. The CMG K can be viewed as a particular case of a multilayer network defined in [159] via tensor products. Adopting the notations of this reference we can write the adjacency tensor $K^{\alpha\tilde{\gamma}}_{\beta\tilde{\delta}}$ of K as:

$$K^{\alpha\tilde{\gamma}}_{\beta\tilde{\delta}} = \sum_{\tilde{h},\tilde{k}=1}^{T} \sum_{i,j=1}^{N} w_{ij}(\tilde{h}\tilde{k}) \mathcal{E}^{\alpha\tilde{\gamma}}_{\beta\tilde{\delta}}(ij\tilde{h}\tilde{k}), \tag{4.3}$$

where $\mathcal{E}^{\alpha\tilde{\gamma}}_{\beta\tilde{\delta}}(ij\tilde{h}\tilde{k})$ is the fourth-order tensor of the canonical basis of $\mathbb{R}^{N\times N\times T\times T}$ and $w_{ij}(\tilde{h}\tilde{k})$ denotes the weight of the edge between node i on layer $\tilde{h}$ and node j on layer $\tilde{k}$. In the case of the definition given in Eq. (4.2), $w_{ij}(\tilde{h}\tilde{k}) = 0$ when $\tilde{k} \neq \tilde{h}+1$ for all $i, j, \tilde{h}$. Moreover, $w_{ij}(\tilde{h}(\tilde{h}+1)) = w_{ij}$ (edges associated to Ω_x) for $i \neq j$ and $w_{ii}(\tilde{h}(\tilde{h}+1)) = 1$ (edges associated to Ω_{xs}). If the connections within layers are taken into account, similarly to (4.1), we have in addition $w_{ij}(\tilde{h}\tilde{h}) = w_{ij}$.

We can also matricize the tensor to have a matrix form representation. In that case the adjacency matrix W_K of K for the definition given in Eq. (4.1) is the following tensor product of matrices:

$$W_K = I_T \otimes W + O^{(1)} \otimes W + O^{(1)} \otimes I_N, \tag{4.4}$$

where I_T is the identity matrix of size $T \times T$ and O_1 is the off-diagonal matrix where only the upper first off-diagonal part is non zero, i.e. $O^{(1)}_{i,j} = 1$ for $j = i+1$ and zero otherwise. W is the weight matrix of G. Again, the first term is associated to Ω, the second to Ω_x and the third to Ω_{xs}. For the second definition (Eq.(4.2)):

$$W_K = O^{(1)} \otimes W + O^{(1)} \otimes I_N. \tag{4.5}$$

Note that K can also be seen as the strong Cartesian product of the graph G with the directed path graph of T vertices, as defined in [160]. An illustration of the CMG is given in Fig. 4.2.

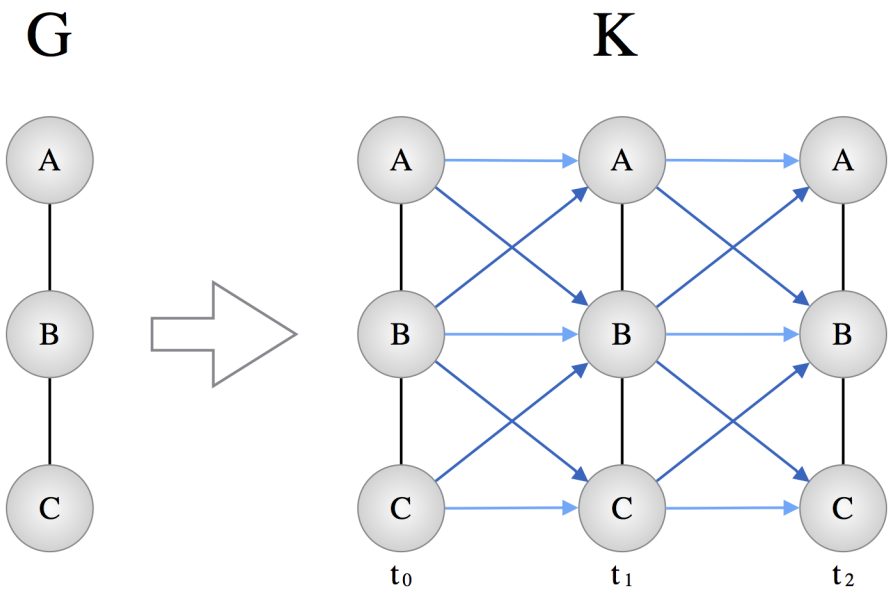


Figure 4.2 – **An illustration of the causal multilayer graph.** The CMG is made of layers: copies of the spatial graph G associated to each time-step (t_0, t_1, t_2). The number of time-steps here is 3. Nodes of the CMG, K, are connected by intra-layer edges from the set Ω (black), by inter-layer edges in the set Ω_x between two successive time steps if they are neighbors in G (dark blue). Finally, self inter-layer edges from the set Ω_{xs} connect the same node in G in successive time step (light blue).

4.1.3 Causal multilayer graph of activity

From signal to binary states

In the general case, a signal is defined as a set of real values without any priors. However in many applications involving causal processes, the activity over the network is binary: active/not active, infected/healthy, congested/not congested, etc. In the present study, we assume that an arbitrary signal associated with each node of G can be cast in a binary activation vector describing if the node is active or not at a particular time step (i.e., by thresholding the signal). We then label each vertex of the CMG with the binary value associated with spatial node i at layer t. For example, in the previous traffic illustration (see Fig 4.1) each node of the CMG could be labeled as "congested/not congested" by setting a limit value for the number of vehicles at each intersection over which the node is considered congested.

Let us denote by $S(i,t)$ the value of S at vertex i_t of L_t. The way that the matrix S of signals is cast into a binary "activation" mask, $\mathbf{M}$, depends on the dataset. It is determined by the definition of the events one wants to track and may involve several application-dependent parameters. In our applications, we use a simple threshold μ, applied after a z-score normalization of the signal as illustrated in Fig. 4.3. The entries of $\mathbf{M}$ are given by thresholding the input signal S with a fixed threshold μ:

$$M(i,t) = \begin{cases} 1 & \text{if } S(i,t) > \mu, \\ 0 & \text{otherwise.} \end{cases}$$

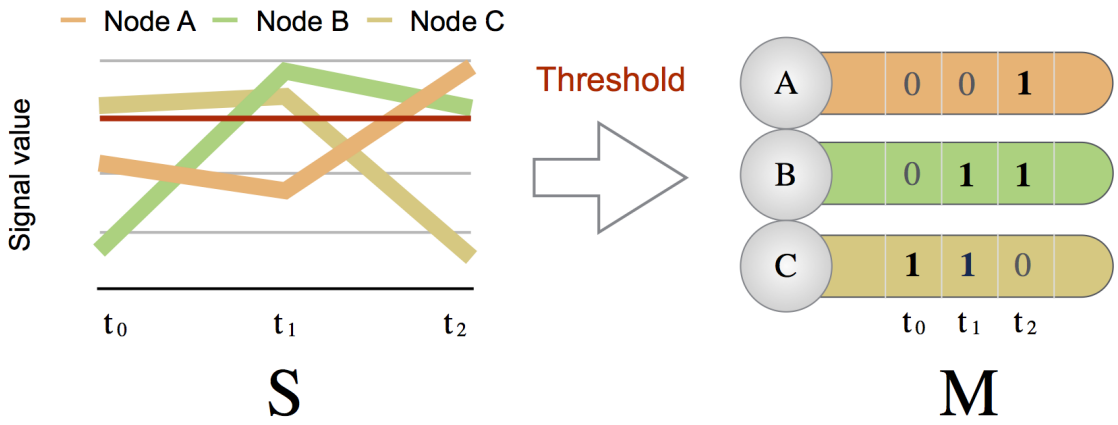


Figure 4.3 – **From signal to binary states.** The input signal is normalized and thresholded to yield a binary activation vector on each of the nodes of the graph.

Combining the causal multilayer graph and the binary mask

The CMG, K, is the skeleton onto which the activation mask is incorporated. From K we create a subgraph, $H = (V_H, E_H)$, called the causal multilayer graph of activity, by taking into account only the set of activated vertices of K. We say that $i_t \in V_K$ is activated and hence belongs to V_H if $M(i,t) = 1$. Edges of H, *causal edges*, exist between nodes if they exist in K. In other words, a causal edge exists if both nodes are neighbors on K and activated. To ease the comprehension, the Fig. 4.4 summarizes the creation of the causal multilayer graph of activity H from the CMG, K, and the binary matrix M.

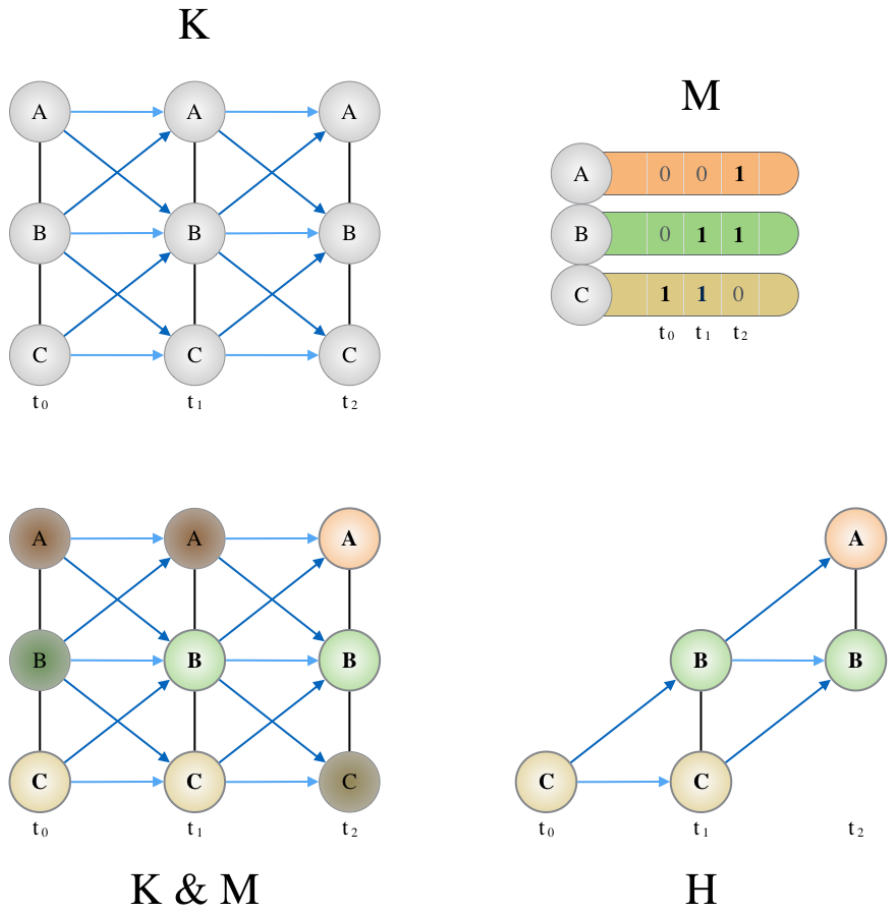


Figure 4.4 – **Construction of the causal multilayer graph of activity H.** From the CMG, K (top left), and the mask, M (top right), the CMG of activity, H, is constructed by labeling each node of K with the binary state from M (bottom left). This operation is represented by the ampersand '&' symbol. Then, only the activated nodes are kept in H, the rest are discarded (bottom right).

In its current form, the construction of H scales poorly with the size of the data as we need to create K with a number of nodes, $V_K = N \times T$, which can be gigantic. We thus optimize this data structure and avoid its full construction by creating H directly from the data. In

Sec 4.4, we describe an efficient, parallelized algorithm, to create the causal multilayer graph of activity. The complexity is linear in the number of edges and vertices of G and in the number of time-steps.

4.2 Analyzing dynamic activity patterns

4.2.1 Dynamic activated components

We define as dynamic activated components (DACs) the weakly connected components of the causal multilayer graph of activity H. Each component, extracted using the standard HCC algorithm [161], encodes a distinct pattern of activity induced by the causal process on the spatial graph G. For each component, we name *width* the number of layers over which the component spans and *spatial spread* the number of distinct nodes of G contained in the component. Note that, subgraphs containing only one node (width equals 1, and spatial spread equals 1) are discarded as they add a little information to characterize the dynamic nature of an event happening in the data.

Except for grid-like structures, networks do not generally have a regular topology. As a consequence, detecting groups of vertices appearing in a repeated manner for a large number of components proves to be challenging. A first solution could be to compare two DACs using approximate subgraph isomorphism and to cluster them according to their similarity score. Here, we propose a more scalable method that encodes each subgraph as feature vectors. We then rely on a standard clustering algorithm to extract global patterns of activity from DACs (see next section).

The first vector, named static feature vector, is a layer-invariant vector constructed by compressing each layer of the DAC and by counting how many times each node is activated. For example, if a spatial node at index i is activated on 3 layers in one DAC, its value on the vector at index i is 3. If necessary, the static feature vector may be normalized using the ℓ^2-norm to help cluster together components with similarly activated nodes but of different temporal width. The idea is similar to the bag-of-word feature vector in text mining. While losing the dynamical aspect, an activation signature remains in the static feature vector. This is enough to perform a significant clustering in the applications presented here.

In addition to the static features, we also encode each DAC as a dynamic feature vector. In essence, it is just a vectorization of the activation mask of each active component. It contains all the information concerning the dynamic arrangement of the component while being memory friendly and computationally efficient. These vectors will be used later on to extract a typical representative pattern from each cluster (see the analysis of cluster properties). To get a clearer picture of these two feature vectors we refer the reader to Fig. 4.5.

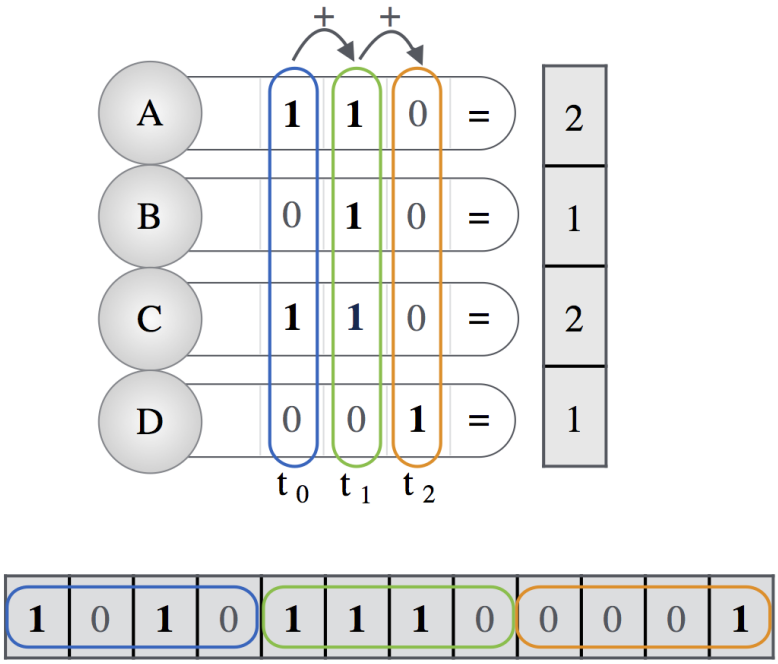


Figure 4.5 – **Creation of static and dynamic feature vectors.** On the right side, the static feature vector is obtained by compressing the layers and counting the occurrence of each node. On the bottom, the dynamic feature vector is created by vectorization of the component's activation mask.

4.2.2 Clustering the components

Clustering DACs is a crucial step in our method as it reveals meaningful groups of components which share common characteristics. In most cases, the groups are unknown, and the clustering is thus unsupervised. While our method is not tied to a particular algorithm, the large number of DACs imposes a fast clustering method. We choose the k-means algorithm as it scales nicely with the load in quasi-linear time using Llyod's method [162]. This algorithm partitions the dataset into k sets, $\{A_1, A_2, \ldots, A_k\}$ so as to minimize the within-cluster sum of squares:

$$\sum_{\ell=1}^{k} \sum_{\alpha_i \in A_\ell} \|\alpha_i - \overline{\alpha}_\ell\|_2^2, \tag{4.6}$$

where $\overline{\alpha}_\ell$ is the centroid of A_ℓ. We expect here k distinct types of DACs that are repeating themselves, with copies possibly differing by a few nodes (due to noise or other activation behaviors). Using Lloyd's method, the complexity is proportional to $N_c N k i$ where N_c is the number of DACs and i is the number of iterations before convergence. In practice, this number is small so that k-means can be considered to scale linearly.

Choosing the right k

Automatically choosing k has fueled decades of research in cluster analysis [163]. While a universal solution to alleviate this issue remains to be found, several methods propose to estimate the number of clusters [164, 165, 166, 167]. Particularly, the silhouette width [168] is a data-driven method that can give a good estimation of the number of clusters in various datasets [163, 169]. However, the automatic detection of a unique k has some limitations. Indeed, a community structure can exist at different scales within the same dataset. Since the silhouette coefficient provides only one k, it sets the scale of the observation to a single level of details. In this work, the choice of k is driven by physical considerations on the data, see the applications on traffic and music data in Sec.4.3. It allows us to work at a desired scale that is physically meaningful and provides interpretable results. More precisely, we first estimate the order of magnitude of k (how many clusters should we expect? 10, 100, 1000?). We then compute several clusterings for different k at this chosen magnitude and study the resulting clusters for validation *a posteriori* (see Sec.4.3). Also, we check the robustness of the clustering: if a small variation of k involves a considerable variation in the clusters obtained, the clustering is not reliable, and another method must be used.

4.2.3 Analysis of the cluster properties and average activation component

By grouping similar DACs, the clustering exhibits structure and proves to be invaluable in the analysis of the underlying causal process. However, since we use static feature vectors as the input of the clustering algorithm, the dynamic activity of each DAC is lost in the "compression" and cannot be retrieved using the k-means centroids. To go beyond the mere analysis of the centroids, we propose to create average activation components (AAC) to represent the average dynamic activity of each characteristic pattern. To do so, we use the dynamic feature vectors associated to the DACs of a given cluster. For each layer, we compute the number of times each node appears for all these DACs. We proceed similarly for causal edges. Note that the choice of the clustering technique used to assign each DAC to a cluster number is irrelevant here as we only need the cluster number to create an average activation component.

From nodes and causal edge counts, we compute node and edge likelihoods in each cluster and store them as node and edge weights respectively. For a node, it represents the likelihood of activation at a particular layer. For an edge, it represents the chance of a node on the current layer to get activated as a consequence of a node activated on the previous layer. The final step consists in sparsifying the AAC by discarding nodes and edges with small weights (typically $< 5\%$). It removes a significant part of the noise induced by the clustering of the underlying components.

4.3 Applications

In the following sections, we test our framework's ability at retrieving and analyzing recurrent patterns of activity induced by causal processes on a graph. We also illustrate the versatility of our framework by creating causal multilayer graphs of activity and extracting dynamic activation components on four completely independent datasets, showing that it can be applied to a large class of problems.

We gradually add complexity and depth to the analysis of each application starting with the visualization of crowd movements in a train station. Second, we come back to Genezik with an extension of the global recommender system based on the dynamic analysis of thousands of playlists. Switching to another research field, our third application deals with neuroscience as we uncover the dynamics between the main anatomical regions of the brain. Finally, we bring evidence of the formation of spatio-temporal communities of users as we study Twitter activity in our last application.

Note that it would be unrealistic to cover in full all the applications that we present here as it would outgrow the subject of this thesis focusing on network science. We thus restrict ourselves to preliminary results that reveal the worth and potential of our model and hope to tackle each example more thoroughly in future work. For clarity purposes, the causal multilayer graph of activity is now simply named causal multilayer graph.

4.3.1 Visualizing crowd movements in a train station

Our first application builds on the motivational example introduced in this chapter by proposing to visualize and quantify pedestrian movements in the train station of Lausanne, Switzerland. Far from a purely academic interest, the Swiss Federal Railways (CFF) are seeking to expand the corridors in prevision of an ever increasing traffic. By proposing to visualize repeated crowd movements in the station, we thus hope to help the CFF to improve the life of thousands of daily commuters.

The dataset, gathered by Alahi in [170], is composed of 42 million points (x, y, t, pedestrian id) tracked by a connected network of cameras placed in the two main corridors of the station. The data collection spans over two weeks at the most crowded hours (7-8 am and 5-6 pm).

The graph

Before the creation of the causal multilayer graph, a spatial graph of connected regions onto which the crowd moves needs to be created. To facilitate the interpretation of the results, we divide the station corridors into small areas of one square meter. The number of persons over time passing on each of these regions directly gives the congestion rate in person per square meter. However, some locations are more important than others as the crowd does not evenly spread over the corridors. To account for the crowd's density, we use an adaptive algorithm

that robustly cuts the space into fine-grained areas on the most congested zones and into coarser areas where the traffic is less dense. Also, areas are constrained to be within the same range of surface, around 1-meter square. This adaptive grid is thus more precise where the congestion flow needs to be monitored and controlled. Less dense regions are represented by coarser areas leading to a more computationally efficient algorithm on those parts of the grid. The result can be seen in Fig. 4.6. This technique is commonly used in computer vision applications to cut images into "Superpixels". The method we use is detailed in [171] where the pixel colors are here replaced by the total number of persons who crossed each spatial point. The nodes of the spatial graph G are the superpixels and the edges are created by linking adjacent superpixels.

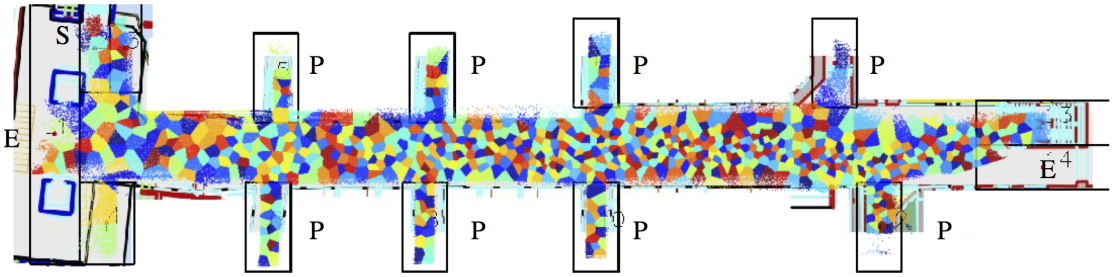


Figure 4.6 – **Segmentation of the west corridor into regions of interests according to the density of pedestrians.** On the corridor map, the eight black boxes labeled as 'P' represent the platforms to access the trains. The letter 'E' stands for the two exits and 'S' is a shop. Each colored polygon is a small area of the corridor, represented by a single node on the graph. The primary access to the station, on the right, is busier than the one on the left, giving smaller polygons.

The signal and mask

We naturally choose the pedestrian's density in the station as the signal. The signal sampling rate is set to 5 seconds. This order of magnitude for the time sampling emphasizes the slow movement of a large crowd (congestion) over the faster standard flow (5 sec is the average time needed for a person stuck in a congestion to move from one superpixel to an adjacent one). The value of the signal at each time step is the number of pedestrians that have crossed the area within this 5 sec duration. We normalize each signal to have a zero mean and a standard deviation of one (z-score) as we are more interested in the variations rather than the absolute number of pedestrians. We create the congestion mask M by empirically setting the threshold μ and applying it to the normalized signal. This casts the traffic activity in a congested/noncongested state where the width and spatial spread of DACs are directly related to the threshold value (higher values give smaller components). In this application, the best threshold ($\mu = 1$, one standard deviation of the original signal) is chosen so that the spatial spread of the DACs are on average within the range of the distance of one exit to another. This choice of the threshold is relevant to model pedestrian trajectories as it emphasizes on the

crowd direction and global behavior within the station.

Analysis of the activated components

The causal multilayer graph is made of thousands of components naturally split by the activation threshold. The basic statistics of each component, such as width and spatial spread, are useful to quantify the impact of an event (e.g., a train departure) in the whole station. The width gives the duration of an event and the spatial spread the number of regions impacted by it. Note that the congestion event is tracked over time, the width and spatial spread take into account all of the congestion patterns even if it moves along the corridors.

The k-means clustering of thousands of components of similar shapes creates average activated components correlated with the departures and arrivals of trains in the station. To choose an appropriate k we proceed as follows. Knowing that the time-series span over two hours per day, the expected number of departures and arrivals is between 15 to 25. This gives an order of magnitude for k since we expect clusters to be correlated with arrivals and departures of trains. We then compute the clustering for different k around these values. With $k = 20$, clusters have flow patterns with a spatial length of several meters, representing accurately pedestrians trajectories within the station. Two examples are given on Fig. 4.7 (left and right). Each one represents an average dynamic trajectory of pedestrians inside the corridors. On these average components, a node represents an activated (crowded) area of the corridor, the node color represents the time dimension, from the start of the component (blue nodes) to its end (red nodes). The two examples display mean congestion patterns evolving in time as the crowd moves along the corridor. It demonstrates the ability of our causal multilayer graph model to track the crowd movement and extract relevant information from it.

To conclude this section, average activated components (associated with each cluster) can be used as a basis for the analysis of the most recurrent events in the station. It provides information such as the most frequently crowded areas or the largest congestion in time, on space, highlighting any traffic "bottleneck". Also, unusual events (delay, accident, etc.) in the station causing a congestion can be detected by comparing its activated component to the average one of each cluster. A significant dissimilarity with all the groups is considered abnormal and may require human intervention in the station.

4.3.2 Analyzing thousands of collaborative audio playlists

Our previous application extracted activated components from causal processes modeled as time-series on a graph. Here, we use the causal multilayer graph approach to analyzing another kind of dataset replacing time series by causal relations between nodes of the spatial graph. Our objective is to show how to use activated components to complement Genezik recommendations, see 3.4, using frequent pattern mining in [172] on the Art of the Mix dataset previously introduced in 3.3.5. Also, we complement our analysis by visualizing groups

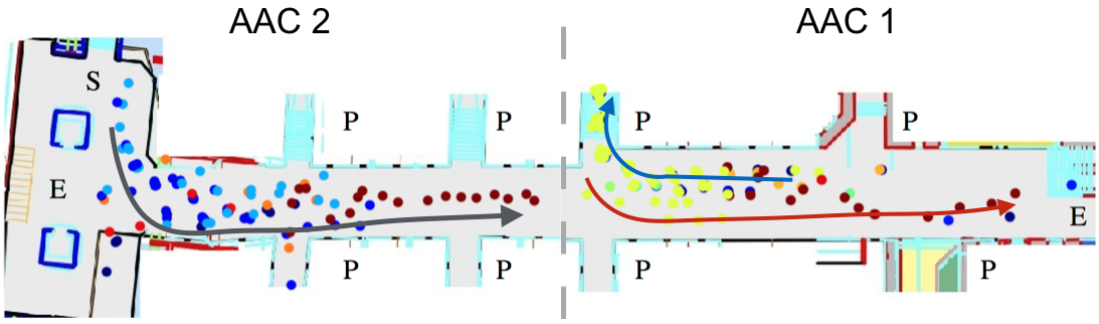


Figure 4.7 – **Two examples of average activated components plotted on the same figure.** Each circle represents a crowded area at some point in time. The color scale gives the time of appearance of congestion at a given location. It ranges from dark blue (start of the congestion) to yellow, then to red for the end of the component. On the right is an average activated component giving a repeated congestion pattern between one platform and the main exit. At the beginning of the component, pedestrians go on the platform (blue arrow), just before a train departure. Then the crowd coming out of the train can be tracked, from yellow (earlier time) near the platform access to red (later time) as it approaches the main exit following the red arrow. The second activated component on the left shows the crowd coming both from the left entrance and from a shopping store and entering the main corridor following the gray arrow. This component ends inside the corridor as the crowd then spreads to different platforms, reducing the congestion rate below the threshold.

of common listening patterns of users. We delayed the introduction of this application in the second position to let the user be familiar with the method before moving on a more sophisticated type of relation between nodes. This example shows that a wide range of applications can be modeled by our framework.

As we recall, the Art of the Mix dataset regroups 101,343 collaborative mixes from 1998 to 2011. A mix is a particular kind of playlist where songs are chosen to have meaningful transitions between them. In other words, songs are put in a specific order in a mix because there exists a causal relationship between them. The position inside the mix is also important as a mix possesses a global evolution. The types of songs (their mood, energy or degree of danceability) in the first part are often different from the ones in the middle or at the end. In addition to the ordering of songs, a mix is also associated with a playlist category by a user. These playlist categories such as: “Rock”, “Romantic”, “Single Artist”, help to navigate between the thousands of playlists on the site.

The graph

A graph, G, is constructed by doing the union of all the playlists in the dataset. As a consequence, an edge between two songs only exists if at least one playlist contains this particular sequence of songs. This graph thus encodes song “affinity” together with their *causal relation-*

ships. The number of nodes of G is over 159,000. Every edge in the graph has been created from an actual human-made audio playlist. It is thus perfectly suited for building new playlists following personal tastes.

The signal and mask

As a signal on the graph, we use the likelihood that a song is in a particular place on a playlist. To compute it, we count how many times a particular song has been set at a particular position for all playlists. For example, take a song A which has been placed three times in the first position of a playlist and five times in the third position. Thus, in this example, the vector on the node A has 2 non-zero values $(3,0,5,0,\cdots)$. The number of entries of each vector is equal to the number of songs in the longest playlist. Similarly to the previous application, we normalize the signal by z-score, giving a likelihood to be at a given position in a playlist. As a consequence, a song with positions evenly distributed in playlists do not reach the threshold. Only songs appearing at a limited number of positions in playlists, e.g., only in the first and third position in the previous example have non-zero binary values for these positions. Note that having distinct locations for songs is important as we use their ordering in playlists as causal relationships between them. Setting $\mu = 0.1\sigma$ proves to be a reasonable choice of threshold as the extracted components have an average width of around 9 steps: it is close to the mean length of a handcrafted playlist in the dataset.

Application to music recommender systems

A direct application of average activated components obtained by k-means clustering could generate "moods" in an entirely unsupervised manner taking into account the long-term coherence of a mix using the playlists collected from all users, see Fig. 3.6. The mix itself is either generated automatically using user taste and play points or mined from existing playlists that are uploaded to the cloud.

On the Art of the Mix dataset, exclusively composed of humanly generated playlists, we show that the application of our method reveals that different music moods are associated with the clusters. Moods such as "Electroish", "Metallic", "Rocky" (see Fig. 4.8 for example) can be viewed as a meta-genre of music regrouping related music genres such as Rock, Indie, Alternative, etc. Theses moods collectively learned from thousands of playlists could also be integrated to dynamically change the weights of edges while Genezik evaluates the best song to recommend at each step of the algorithm.

We extract activated components of songs from the causal multilayer graph by thresholding the normalized popularity vector (keeping the largest peaks of appearances of songs in playlists). Each activated component is a group of songs fitting nicely together and respecting a precise order within the playlists. These activated components are then clustered together using k-means clustering. The number of clusters $k = 30$ is here naturally given by the number of

dominant genres present in the dataset. This is a natural choice as we expect playlists to be classified by genre. Also, music genre often proves to be one of the most important criteria when creating a mix: in the dataset, half of the distribution of the playlist categories are labeled with a music genre. This choice of k is validated by the results which are indeed meaningful in term of genres.

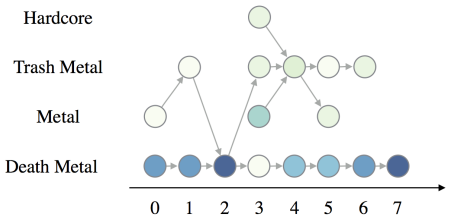
Each cluster can be labeled with a mood *a posteriori* by analyzing music genres present within the clusters. Once the mood has been chosen by a user, the algorithm selects the average activated component associated with the mood. To generate a playlist, a seed song, the first of the playlist, is chosen in the first layer of the component. In its simplest form, the selection is made at random. However, many criteria such as user history, ratings or time since last played, can be used to select a starting point. Then, the rest of the playlist is constructed by doing a random walk on the *causal* edges of the component. The familiarity versus discovery ratio can be tuned by modifying the edge weights according to the popularity of each node. The random walker would have more or fewer chances to reach a popular song depending on the user's will.

A good playlist should alternate between familiarity, discovery, and smoothness in transitions between songs [173]. An average activated component is a coherent weighted subgraph of songs, where each node and edge are weighted by their likelihood of appearance at that particular position. The most popular songs, appearing in many activated components, have a significant weight, filling the contract for the familiarity part. Less popular songs will also be clustered together giving a choice for the discovery part. Finally, the smoothness of transitions is guaranteed by the graph G: each song to song transition has been created by a human and is appealing to at least one of them.

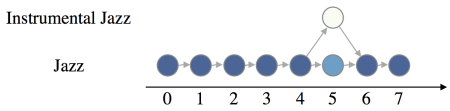
Keeping the original ordering of songs in a playlist (successive positions of several songs, not just 2) is a major factor in the design of recommender systems according to [173] and [172]. Following the causal edges of an average activated component takes into account the ordering of several successive songs with their positions within playlists, unlike a simple random walk on the graph G which considers only one to one coupling between songs. The implementation of this system in the global recommendation service of Genezik could prove very useful to discover new music horizons completely unfamiliar to the user.

Data visualization of mood patterns

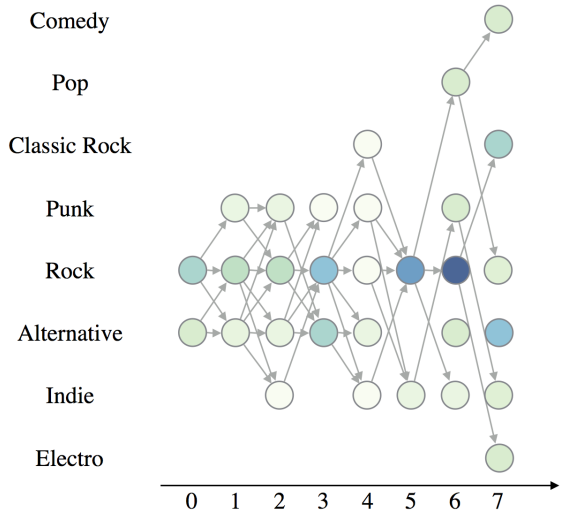
To end this part on the usage of activated components on music, we propose to visualize some on the activation patterns detected by our method. To do so, we group songs on each layer by genre as it drastically reduces the dimensionality and exhibits interesting insights on how users create playlists. Note that genres are here to validate the methodology and have not been used to cluster the activated components together. Like in the previous application the method is completely unsupervised. The results are shown in Fig.4.8a, Fig.4.8b, Fig.4.8d, Fig.4.8c.



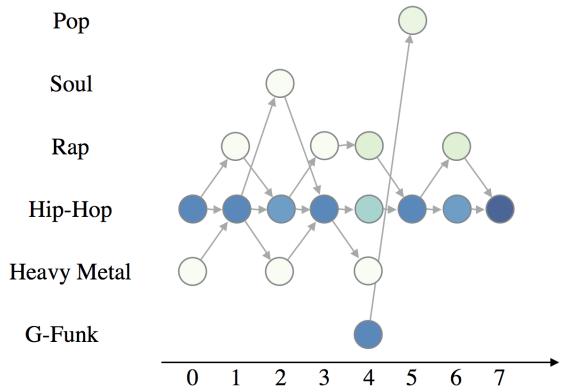
(a) Metal component. While the genre diversity is a bit higher than in the Jazz example (b), all genres belong to the same meta-genre: Metal.



(b) Jazz component. We clearly see that Jazz playlists are very pure and do not mix with other genres. The same phenomenon also exists for other music niches such as Classical music.



(c) Rock - Alternative component. The diversity of the genres is much higher than in previous examples but stays coherent



(d) Rap / Hip-hop component. Heavy Metal nodes are not outliers but are indeed connected to Hip-Hop with songs from famous artists such as Korn, Limp Bizkit, Public Enemy, etc.

Figure 4.8 – **Average activated components representing music moods.** For a clearer visualization, we have discarded nodes with a low probability of appearance. The horizontal axis represents the layers: the position in the playlist (limited to the eight first layers). The nodes are colored from light to dark according to their weights, i.e. their likelihood of appearance (larger is darker).

While the dataset is biased towards Rock, Alternative, and Indie (more than 40% of all the songs), the clustering still achieves to extract relatively pure patterns of related genres, or music "moods", as it is shown in Fig. 4.8a and Fig. 4.8b. As music experts could have expected, songs of favorite genres are more volatile: they can easily be mixed with other genres and have a higher spatial spread, see Fig. 4.8c and Fig. 4.9. On the contrary, songs of "connoisseur genres" such as Metal, Jazz, Hip-Hop or Classical stay clustered in their universe.

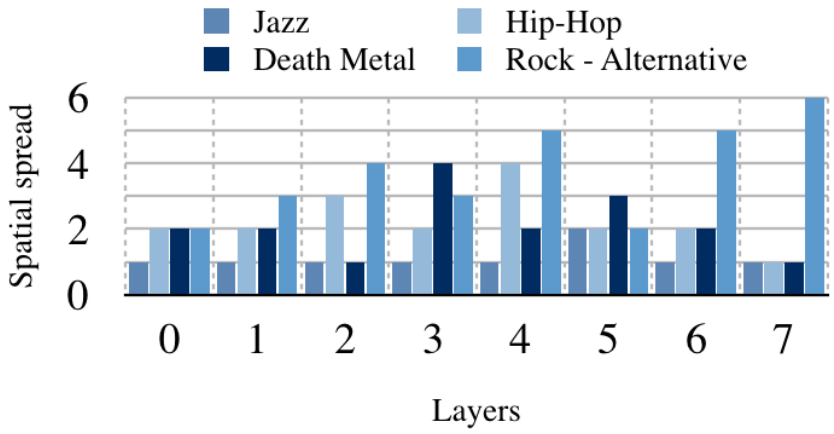


Figure 4.9 – **The spatial spread of the different average activated components presented in Fig. 4.8a, Fig. 4.8b, Fig. 4.8d, Fig. 4.8c.** The spatial spread is the number of genres per layer of the average activated component. As one could expect, popular genres such as Rock and Alternative have a bigger spatial spread and are easily mixed with other genres.

4.3.3 Dynamic activation patterns in brain MRI data

In this example, we radically change the domain of application of our method as we try to uncover the dynamical interactions of brain regions using on magnetic resonance imaging data. Magnetic resonance imaging (MRI) is one of the most modern techniques to analyze the brain anatomy and function. By collecting information on the brain's activity over time via functional MRI recordings and its connectivity map through diffusion MRI, these imaging data are expected to be extremely useful for the characterization of pathologies such as Alzheimer's disease, multiple sclerosis, epilepsy or psychiatric disorders [174, 175].

Recently, processing the brain's low-frequency activity during task-free experiments allowed to identify spatial patterns of coherent brain activity: called resting state networks (RSNs) [176, 177]. These RSNs have mostly been extracted using spatial independent component analysis (ICA) on the very noisy data obtained by resting state fMRIs. While resting state activity schemes are utterly complex and difficult to visualize, these first steps mark a substantial progress in the understanding of the brain's behavior.

It is important to stress that these methods probing resting state activity are limited to non-stationary aspects of the functional time series. Spatial ICA and correlation approach output average spatio-temporal patterns over the whole recording time, discarding a large amount of information. Besides, they do not take into account the brain structural connectivity network available through diffusion MRI which could bring additional and complementary information. Indeed, patterns of strong anatomical connectivity between brain areas have been associated to high functional coherence [178, 179]. On the other hand, some studies are devoted to the structural connectivity of the brain [180, 181], yet, they do not take into account the activity in each region.

The application of the causal multilayer graph could thus go beyond the limited static analysis of activation patterns and reveal how brain regions interact over time in a dynamic manner with a much finer precision than existing methods. In addition to being more robust to noise due to the particular combination of graph and signal, the visualization of brain patterns would additionally be of great use to practicians allowing them to interact with the patient in real-time while performing the MRI scan.

Our dataset is composed of high-resolution morphological MRI (T1w contrast), resting state functional MRI (fMRI), and data from diffusion MRI (more specifically, diffusion spectrum imaging) that was acquired for 75 healthy volunteers on a Siemens Trio 3T scanner equipped with a 32-channel head coil. This work has been done in collaboration with Alessandra Griffa for the data collection, preprocessing and the interpretation of the results, see [182].

The graph

The construction of the structural brain network from diffusion MRI data follows the technique previously described in [183]. Notably, each subject T1w volume is segmented into

68 cortical regions [184], which constitutes the vertices of the brain network. After that, we determine the presence of an anatomical connection between pairs of brain regions from whole-brain tractography on reconstructed Diffusion Spectrum Imaging (DSI) data [185], building anatomical connectivity networks. To reject noise and extract common connectivity patterns among subjects, we choose to pool the subjects-wise connectivity information and to create a group representative graph where an edge between two regions is present only if it exists in at least 75% of the investigated subjects, similarly to [186]. Eventually, the graph is binarized by setting all the connection weights to one. We assume that regions which work together during resting state are not influenced by the strength of their edge.

The signal and mask

To associate a time-series of activations per region of our atlas, we compute the average fMRI time series over voxels belonging to each region as suggested by neuroscientists, see [182, 187]. fMRI data are corrupted by multiple sources of noise and artifacts that we rectify following the methods documented in the literature [188, 189, 190]. More precisely, we follow a standard denoising approach, including motion correction, nuisance signals regression, and linear detrending using the UNLOCBOX[1]. After that, we apply a spatial Total Variation (TV) denoising in 3 dimensions as suggested in [191, 192] to enforce sharp boundaries between active and not active regions. In addition to the spatial treatment, we apply a temporal band-pass filter of band-pass (0.01-0.1 Hz).

Despite our efforts, we observe that the intensity of the resulting node time series may vary considerably from node to node because the 68 brain regions may not perfectly match fMRI activated regions. We thus normalize the data, i.e. subtract the mean of the time series and divide it by its standard deviation (a common procedure in fMRI studies). This process reduces the discrepancies between regions and allows for a global thresholding approach of the time series, used in all our applications so far.

The choice of the activation threshold to create the binary mask is guided by biological considerations. Too high, it leads to a reduced number of small subgraphs, difficult to interpret. Too low, it gives an enormous amount of components where a part of them are due to the noise of the activation signal. Besides, there are a remarkable number of large components, made of several activation patterns, complex to classify and understand. Hence, we set μ as follows.

First, we require the width of each component to be between 3 and 8 time steps, which roughly corresponds to an activation between 6 and 16 seconds. This range corresponds to the expected duration of the hemodynamic response function, signaling a brain activation. Second, we filter the spatial spreads of the components between 2 and 11. It allows limiting the error in the input data where component could activate a major part of the brain, and it also limits the nodes that only activate themselves through time. If we expect 7 RSNs, each of them

[1] Matlab toolbox available at http://unlocbox.sourceforge.net/

activating around three times for each of the 75 patients, we get a number of RSN components equals to 1,575. We must add to this value an amount of components not related to resting states. In consequence, μ is first set to a large value, and it progressively decreased so as to get around 1,800 components after filtering too small and too large components. We found that less than 8% of the components are larger than the width and height limits, validating our hypothesis on the scale of brain activations. The positive and negative activations (or deactivation) are equally treated by looking at the absolute value of the fMRI signal. The way of handling negative values is still discussed in the literature and may be changed in future work. Note that the effect of the preprocessing pipeline has been shown to be difficult to quantify [193] and would require a detailed study, out of the scope of the present work.

Recovering the resting state networks

Before revealing the dynamics of brain regions, our first step is to retrieve the resting state networks to validate the pipeline. To do so, we apply our method and extract a large number of repeating activation components that we further classify with k-means to obtain average activation components. If our assumptions hold, average activation components should represent RSNs. Also, these clusters should be well separated as they correspond to different parts of the brain and share only a small number of nodes if any.

To compare our clustering with RSNs, we associate each node of the brain network to a functional resting state network id according to the map given by [176]. There are 7 RSNs: 1) Visual (V), 2) somatomotor (SM), 3) dorsal attention (DA), 4) ventral attention (VA), 5) limbic (L), 6) frontoparietal (FP) and 7) default mode (DM) networks. Note that our anatomical atlas of 68 nodes does not exactly correspond to the shape of these RSNs given via fMRI data. For example, some large regions such as the fusiform region are in the limbic RSN and also in the visual one but we labeled it as a visual RSN part. Moreover, some RSNs are made of small pieces scattered all over the brain (dorsal attention and frontoparietal) which render difficult their description with our atlas. These two RSNs are made of only two and three atlas regions respectively, which make them harder to detect. As a consequence, some discrepancies will appear when comparing our results to the RSNs of the literature.

Since the number of clusters k has to be set manually, we tested different initializations and number of clusters. Our experiments showed that the clustering is robust: for each test, a large number of clusters could be associated to a single RSN. Moreover, a significant number of clusters stay unaltered from experiment to experiment.

We present the results of a clustering with $k = 12$ on Fig. 4.10 showing that our method can retrieve known RSNs. Indeed, the clusters are in good correspondence with the RSNs with cluster 11 associated to the visual RSN, cluster 4 to the somatomotor RSN, cluster 7 to ventral attention and cluster 2 to the default mode. Only a few of the clusters contain a mixture of RSNs.

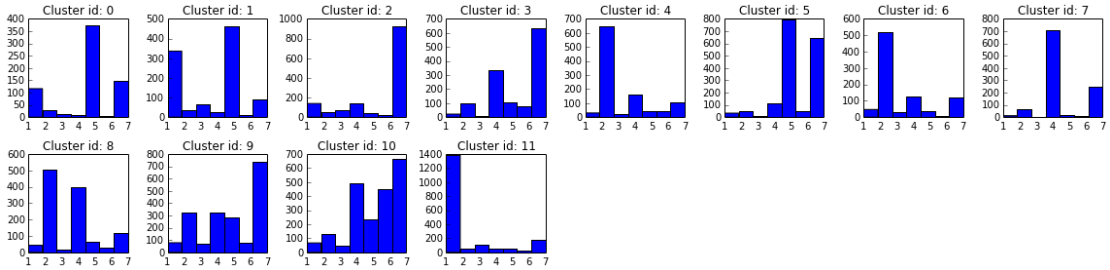


Figure 4.10 – **Node distribution among the 7 RSNs for each cluster.** The clustering is done for $k = 12$.

This is even more convincing when plotting the average activation component for some of the clusters with $k = 12$ on Fig. 4.11, 4.12. One can see two categories of components: symmetric ones on the left and right hemispheres such as cluster 2, 4, 5, 6, 7 and 11 and non-symmetric (unilateral) ones which can be grouped by pairs such as 0-1, 8-9 and 3-10. The coexistence of symmetric/dissymmetric components is similar to what [194] obtains by directly clustering at the voxel level (with 11 clusters). The pair 3-10 corresponds to a frontoparietal activation, which can be interpreted as a part of the dorsal attention network (known to possess a lateralized behavior) even if some regions are labeled as default mode network. The clusters 0-1 (limbic RSN) and 8-9 contain regions of the temporal lobe while being dissymmetric.

Some components appear to be part of more than one resting state similarly to the results of [195] obtained by a temporally-independent component analysis. As an example, some of their dynamic components contain part of the default mode as well as other resting states (see their mode TFM 8, a symmetric version of the mean component of cluster 9). It is also the case in [196] where a sliding window ICA is used to retrieve dynamic patterns.

We summarize results of the clustering in Tab. 4.2. For each cluster, the dominant(s) RSN(s) is given from the histogram of Fig. 4.10. The corrected dominant(s) RSN(s) is obtained by looking at the average activation component of each cluster and deducing the most likely associated RSN. We also state if a cluster has a lateralized component and which cluster contains its symmetric counterpart in parenthesis.

Table 4.2 – Clusters and RSN distribution.

Cluster	0	1	2	3	4	5	6	7	8	9	10	11
RSN	5	1-5	7	7	2	5-7	2	4	2-4	7	-	1
Corr. RSN	5	5	7	3-6-7	2	5-7	2	4	2-7	2-7	3-6-7	1
Lateralized	y (1)	y (0)	-	y (10)	-	-	-	-	y (9)	y (8)	y (3)	-
Nb of comp	104	125	133	124	95	164	84	125	136	231	231	157

Visualization of the dynamical activity of the brain

Now that we retrieved the results from the literature, we can move on to the novelty brought by the usage of the CMG by analyzing the dynamics of components. We start by focusing on the number of occurrences of each of the 12 "activation modes", associated with each cluster. One can see from the previous Tab. 4.2 that clusters 9, 10, 5, 11 are the largest so that components associated with the DM, DA/FP, SM and V networks are occurring more often in the recordings. To reveal more clearly the most occurring ones, we group clusters by their associated RSNs in Tab. 4.3. As we can see, the sum of modes involving the default mode (DM) network or part of it gives the highest activity by far, underpinning the central role of the default mode in resting state.

Table 4.3 – **Occurrence of each resting state network in the dataset.**

RSN	1	2	4	5	7	3-6-7	5-7	2-7
RSN name	V	SM	VA	L	DM	DA-FP-DM	L-DM	SM-DM
Nb of components	157	179	125	229	133	355	164	367

Concerning the AAC themselves, we can see in Fig. 4.11 that cluster 11 can be separated into two components by filtering the edges with a low likelihood of activation in dashed lines. These two components match the primary visual cortex in purple and the extrastriate visual cortex in orange: both are known to correspond to two resting patterns [197] working most of the time separately.

This space-time visualization also gives some hints on how the regions interact between hemispheres. For instance, in cluster 2 the precuneus regions equally interact between themselves and their symmetric counterpart but less with the isthmus-cingulate. In the lateralized clusters, the nodes are more linked to themselves in successive time steps than to their neighbors' active regions. In cluster 4, see Fig. 4.12, the somatomotor RSN, connections are balanced with parts equally interacting with each other.

To conclude with the application of our method to brain signals, we underline the fact that doing statistics over many components emphasizes common behaviors but also averages their dynamics. A much richer dynamic response can be found by looking at raw dynamic activation components in Fig. 4.13. For example, in the left figure, we can see activations passing to different brain areas along time. Here, it concerns the default mode network, showing that this RSN is made of smaller constituents with the frontal area in orange and temporal area in purple interacting together in a dynamic manner, as suggested in [196, 195].

We can also witness another interesting kind of dynamics for the visual area in the right figure. Here, activations progressively enlarge before shrinking and vanishing. This is an expected behavior especially in the visual cortex where the information is first processed in the primary part and then transmitted to the surrounding regions. This example illustrates that a much

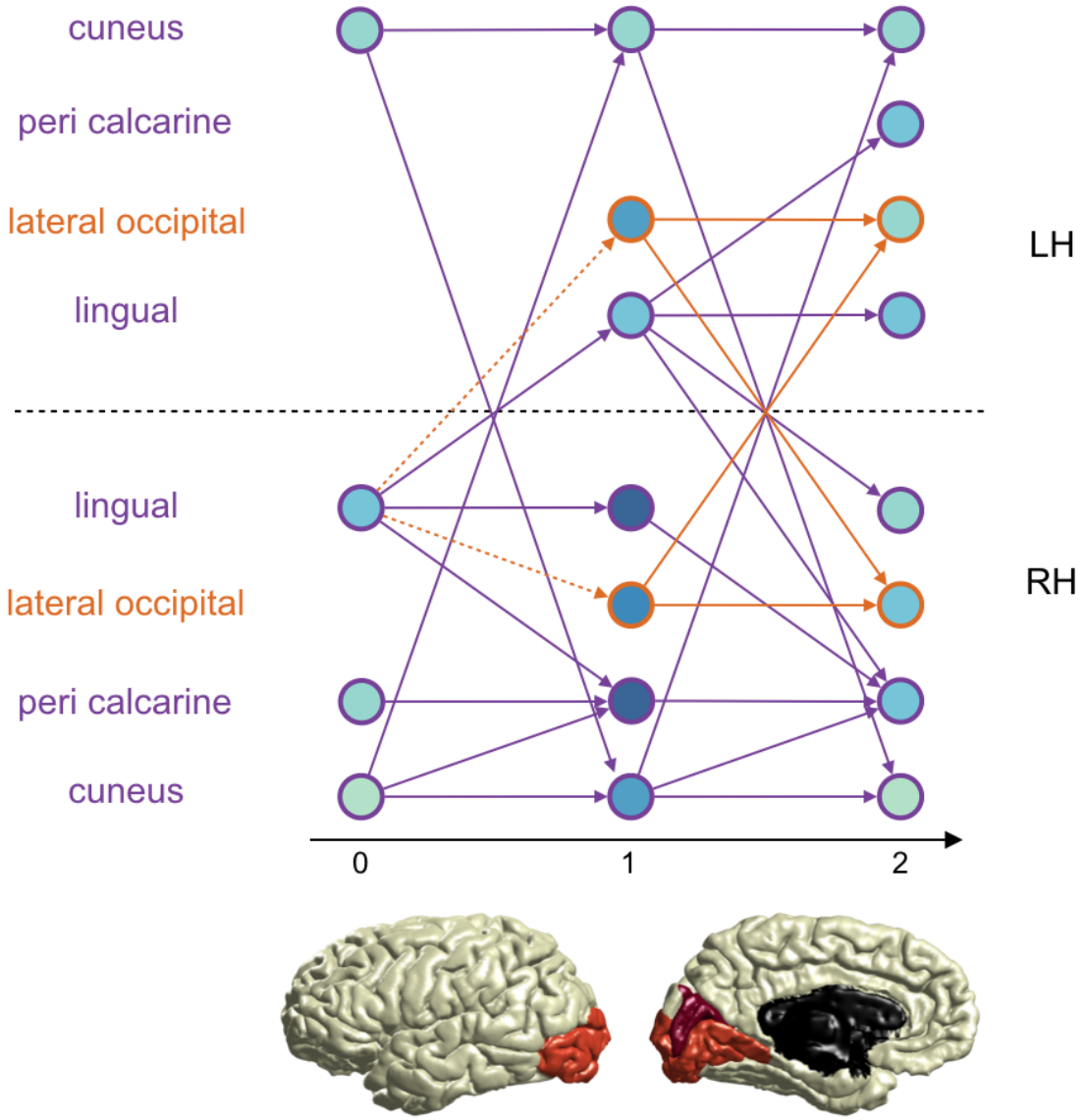


Figure 4.11 – **Average activation component of cluster 11, corresponding to the visual RSN.** It is symmetric between the left and right hemispheres and contains the peri-calcarine, the cuneus, the lingual and the lateral occipital regions. The node colors correspond to the amounts of activations for each node, with dark blue for the nodes most present in the components. The dashed edges have a low likelihood of activation. If we remove theses edges, the AAC is cut into two parts, the primary visual cortex in purple and extrastriate visual cortex in orange.

finer clustering is possible and would be of interest in future work.

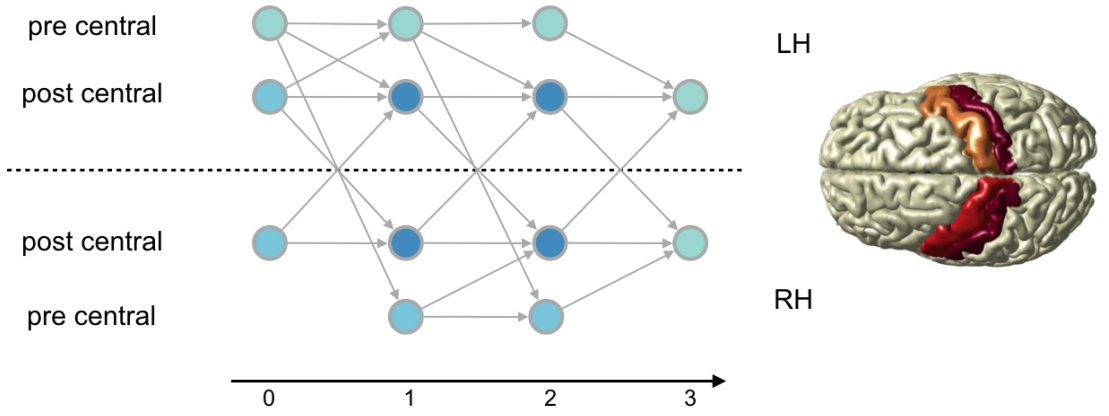


Figure 4.12 – **Average activation component of cluster 4, the somato-motor RSN.** Both left and right pre-central and post-central regions are active, the post-central region being activated more often among the components belonging to the cluster. The relatively similar edge color indicates that all the regions equally interact with each other.

4.3.4 Rumor spreading on Twitter and dynamic activity of communities

Our last application aims at revealing the dynamic activity of communities when a rumor spreads in a social network. To do so, we focus on the Twitter activity around specific hashtags related to the Higgs boson discovery by CERN in 2012 extending the work of [147]. The dataset, containing more than 0.4 million nodes and 14 million edges[2], also asserts the scalability of our method "in-vivo" by testing its implementation on a commodity server of 24 cores.

The Twitter activity has been recorded before, during and after the announcement of the discovery of the Higgs boson by CERN on the 4th of July 2012 at 8:00 AM GMT. The recording starts from the 1st of July and lasts until the 7th. The authors have recorded the Retweet, Reply and Mention activity containing selected keywords related to the Higgs boson. The graph of Twitter followers is provided together with the activity of the users during the event. To study the dynamical activity, [147] takes advantage of the fact that the activity can be tracked over time using retweets. In addition of being timestamped, a retweet provides the causal link between two users. A user retweets information as a consequence of a first user having tweeted the information. The cascade of retweets can be followed without relying on a causal multilayer approach. Note that the retweet information is asynchronous (not having regular time steps) and can not be directly cast into a causal multilayer graph. It would involve connections between layers not necessarily adjacent. The analysis of the retweet dynamics allowed the authors of [147] to reveal the bursty behavior of retweet chains, in particular around the official announcement.

As a proof of concept, the first goal in this example is to confirm the retweet dynamic made of bursty events discovered in [147]. If this is so, the DACs should be large especially around

[2] available at SNAP [198]

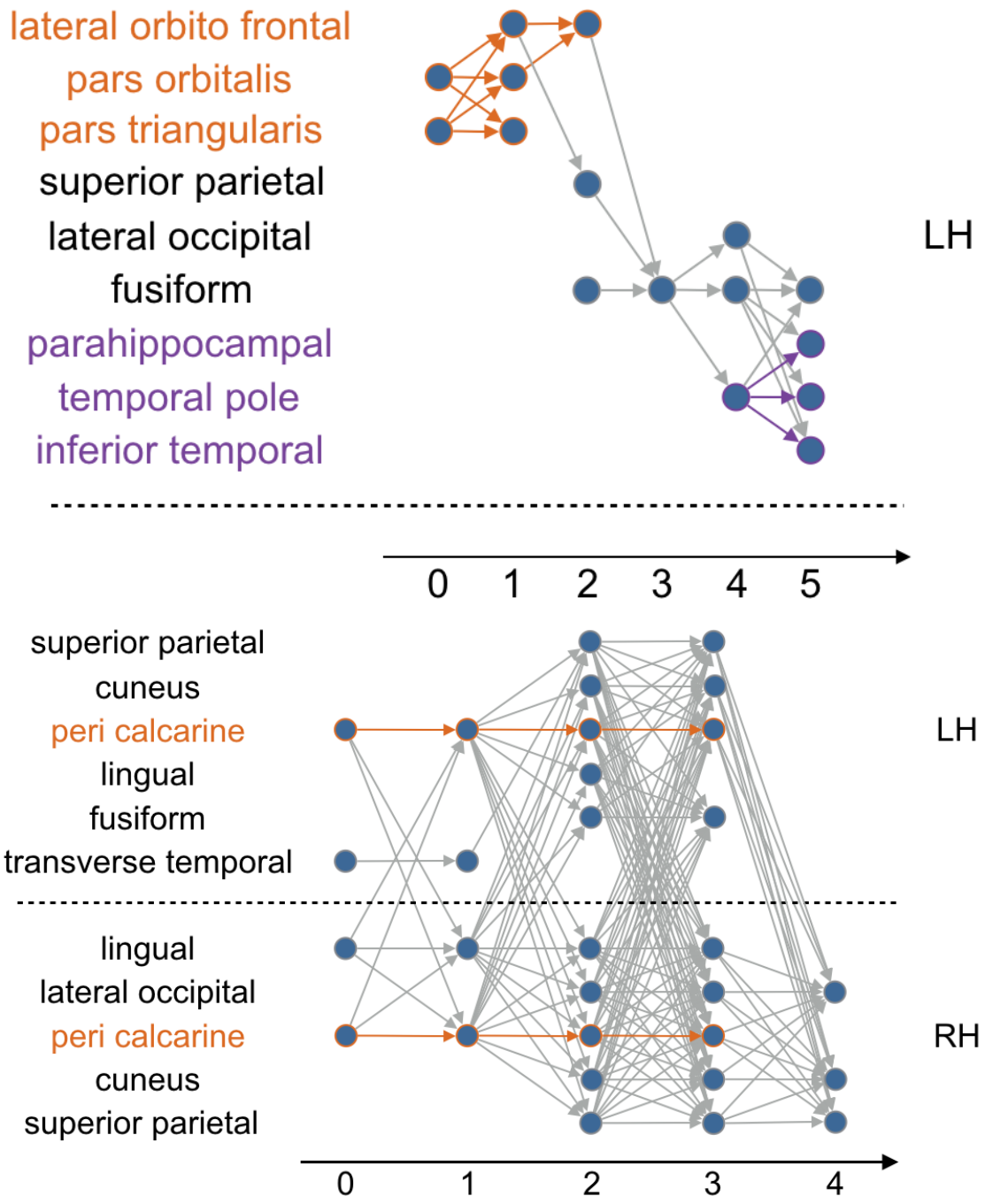


Figure 4.13 – **Examples of raw dynamic activation components in the brain.** On top, the activation of brain left hemisphere regions moves from the frontal area in orange to the temporal one in purple over 6 time steps. On the bottom, the activation spans across several brain regions then reduce before vanishing, mostly involving the visual area. It starts from the peri-calcarine region, which stays active during the whole process.

the announcement. The second task is to show that the activated components can bring new insights on the dataset and rumor spreading mechanisms. Since the dataset is related to a single *extraordinary* event, we do not expect to find repeated patterns of activity. Nevertheless, the analysis of repeated patterns could be done in future work with a larger recording of Twitter activity, containing different events appearing over several weeks.

To build the causal multilayer graph we use the activity over time (time series) and combine it with the graph of followers (the social network) as follows.

The graph

The Twitter follower graph is a directed network where a user (source) is connected to another (destination) if he is followed by him/her. The graph is made of 456,626 nodes of users who have been active (retweet, reply, mention) at least once during the recording and more than 14 million edges.

The signal and the mask

To track the activity over time and users, we cut the Twitter recording into regularly spaced time steps. Within a time step, a user is active if he/she has retweeted[3] about the Higgs boson. Different activity patterns may appear at various scales of time (from seconds to hours to days) and the time step is chosen in order to select a particular scale.

Remember that those connections within the causal multilayer graph are only allowed within layers and between two successive layers. The retweet action of a user may occur after several time steps, and this is not taken into account in our construction. However, there is evidence of bursty behaviors in social activity [199, 200] and in particular in this dataset [147]. A retweet is likely to be done shortly after a tweet appears in the user feed and the likelihood of retweeting decreases with time (following a power law). Hence the time series configuration captures most of the dynamic activity.

Moreover, it is also of high interest to focus on tracking the activity solely due to the bursty behavior. Of course, the time step length must be chosen in order to match the time scale of the bursty processes: a duration of 1 minute is a reasonable choice according to the results of [147]. On the fourth panel of Fig.5 in their paper, the curve shows a maximum of retweets having a time delay of 1 minute and the number of retweets drops exponentially when the delay increases. To highlight the impact of such a choice, we run a second analysis with a 10-minute step. On the curve, the number of retweets within a 10-minutes delay is already two order of magnitude smaller than the 1-minute delay.

[3] We do not include replies and mentions to be able to compare with [147] which focuses more on retweets than the other activities.

Analysis of the activated components, recovering the results of the literature

In the activated components, nodes represent users active at particular time-steps. Extracted in only a few seconds per scale, it shows that our implementation can easily handle the scale of this dataset. In Tab. 4.4, the largest activated components extracted from the causal multilayer graph are shown (10 min sampling). The largest one appears on the 4th of July (the announcement day) and covers most of the day. Moreover, the number of users involved is vast showing how information spreads over the network. The component starts before 8:00 a.m. as rumors and discussions on the topic spread before and increase as the announcement time approaches. The other components, distributed between the 2nd and 5th days of July, are at least one order of magnitude smaller with a duration of one to three hours.

Bursts of activity may appear indistinctly during the day or night as the event is of worldwide importance. However due to the lack of geo-localization tags in the dataset and we could not verify whether components involve particular countries or regions.

Table 4.4 – Largest activated components with their size and time of appearance for the 10 minutes time-steps.

# Nodes	# Layers	Social spread	Start	End
55037	108	36800	04, 03:10	04, 21:00
357	15	324	03, 17:40	03, 20:00
299	12	277	05, 11:30	05, 13:20
254	12	231	05, 05:00	05, 06:50
244	9	235	05, 00:00	05, 01:20
232	12	212	02, 16:20	02, 18:10
200	14	163	03, 21:00	03, 23:10
169	5	107	04, 14:40	04, 15:20
166	9	160	05, 10:30	05, 11:50
142	9	128	04, 20:50	04, 22:10

The largest activated components obtained by the one-minute sampling are displayed on Table 4.5. As expected, the largest components appear on the 4th with 4,704 different users active over 114 minutes. This frenetic activity appears *just before* the official announcement. The second largest component comes as a consequence of the announcement with a quick information propagation starting at 8:01 a.m. until 8:18 a.m. We notice then that most of the largest components last 10-15 minutes and involve around a hundred users. They also take place on the 4th, where the activity is so frenetic that information can be retweeted in less than one minute and propagated to tens of users in only 10 minutes.

Analysis of the activated components, new results, evidence of dynamic communities of active users

The retweet activity does not fully account for the spreading of rumors. For example, a user may see several tweets concerning the Higgs boson in their feed and decide to retweet only

Table 4.5 – Largest activated components with their size and time of appearance for the 1 minute time steps.

# Nodes	# Layers	Social spread	Start	End
8593	114	4704	04, 05:17	04, 07:10
255	18	214	04, 08:01	04, 08:18
216	15	164	04, 04:51	04, 05:05
151	9	130	04, 05:09	04, 05:17
142	5	133	04, 13:41	04, 13:45
100	9	83	04, 07:13	04, 07:21
98	15	98	04, 13:22	04, 13:36
95	15	75	04, 15:08	04, 15:22
95	14	95	02, 19:57	02, 20:10
93	16	91	04, 14:12	04, 14:27

one of them, or possibly decide to come back to the initial source of information to retweet it. In another case, he can tweet the information without mentioning any source. In these cases, the action of tweeting does not take into consideration the full user network even if it has a clear influence on him/her.

To confirm this behavior, we compare the graph of followers to the graph obtained by connecting users according to the retweet data. The result is clear: only 59% of the edges of the retweet graph match the follower's graph. Thus, a significant portion of the retweets is not from direct neighbors. Despite this fact, our causal multilayer graph approach is still able to reveal the existence of communities of users appearing dynamically as the information spreads over the network. We provide evidence for this claim in the following.

First, we remark that the number of layers (time spread or width) of the two different sampling rates is similar: the largest component in each is around 100 layers long while the others are around 15. Because the number of layers is the number of time-steps, components with a 10 min sampling rate last 10 times longer than the ones with a 1 min sampling rate. For these two cases there seem to be a scale invariance in time. However, the social spread (number of different users in the component) is less than 10 times larger between Tab. 4.4 and Tab. 4.5. With the exception of the largest component, it is only multiplied by 2 (roughly). These two facts tend to advocate for a community-like activity where information is retweeted within communities, limiting the number of users involved.

To better understand the rumor spreading dynamics, it is also interesting to focus on the largest activated component on the 10 min sampling rate, described on the first line of Table 4.4. This component covers the official announcement of the Higgs boson discovery. As each DAC is a graph, it is possible to run a community detection algorithm on it. Using Louvain's method [201], we obtain 56 different communities[4] with 4 main ones containing a large

[4] Due to anonymization of the data and the non-availability of the geo-localization data we were not able to check if these communities correspond to a particular location or group of people.

number of nodes (11%, 9%, 7%, 7% of the total number of nodes respectively). A visualization of the graph with the different well-connected communities is shown in Fig. 4.14.

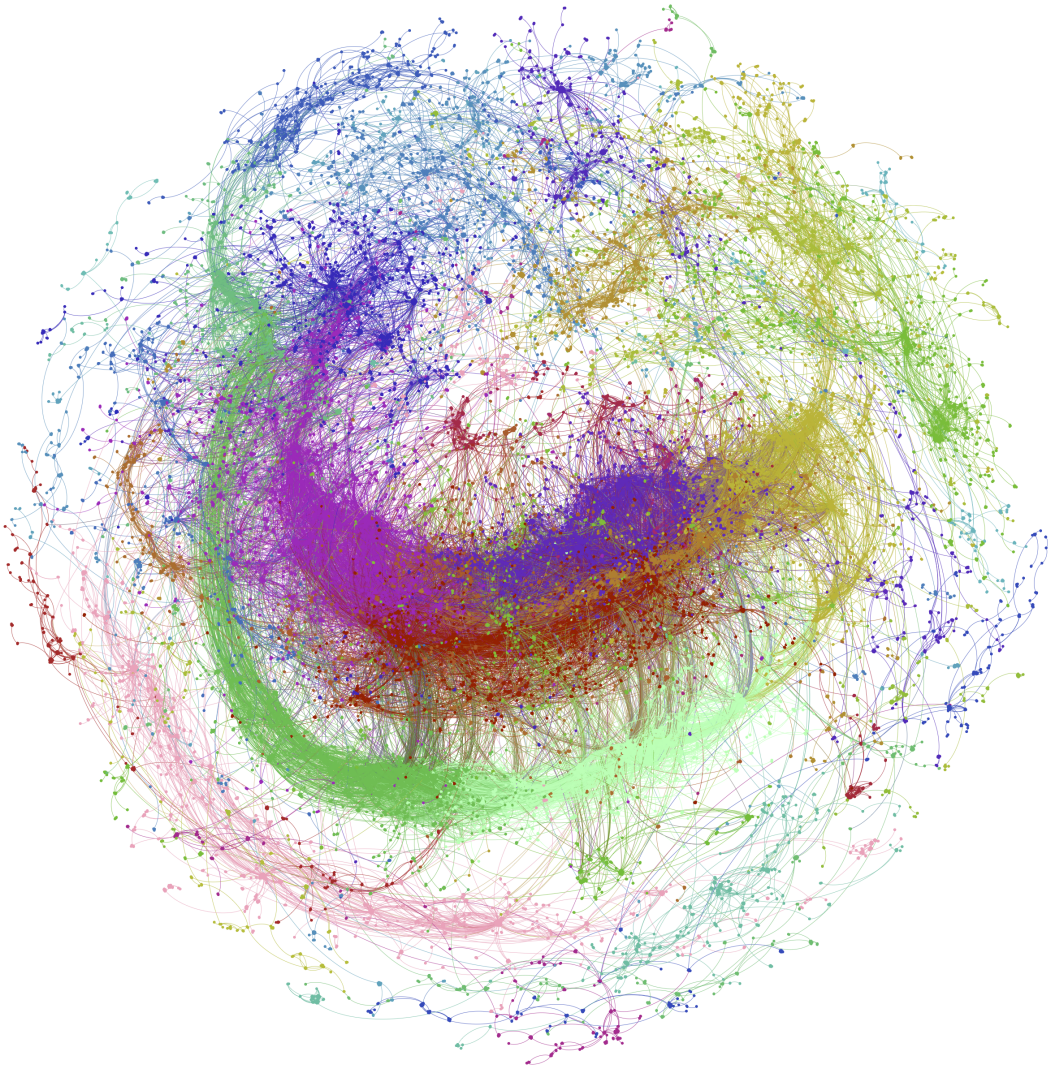


Figure 4.14 – **Graph of the biggest activated component from the 10-minute sampling, colored by communities.** The graph layout has been generated using open ORD [62]. The different colors correspond to the various communities. However, the set of colors is limited and different communities may have the same color. Edge colors correspond to the color of the source nodes they connect to.

Communities are elongated as connections in the graph exist between successive layers only: 2 nodes in the same community may not be directly connected but are connected through their neighbors at successive time steps. We recall that each node of the graph is a user active at a particular time-step. Hence several nodes can correspond to the same user and communities in the activated component do not necessarily represent communities in the Twitter graph of followers. Communities in DACs are thus sets of connected users *active within a particular*

time interval.

A part of the dynamical activity of the component is plotted on Fig. 4.15. On the left, the graph is colored by communities similarly to Fig. 4.14 whereas on the right, each color corresponds to a time-step (a layer) instead of a community. By comparing the two graphs, we notice that communities span across layers. Different communities can be active at the same time (purples and greens on the left) while some of them activate in a serial manner (dark green to light green on the left) leading to *spatio-temporal communities* of users. Some communities are tightly connected on several successive layers (red and purple on the left figure) while some others seem to be interacting by bursts of connections over time (red and greens). This phenomenon looks fascinating and deserves further investigations. Future work also includes designing new data visualizations to depict more precisely these phenomena.

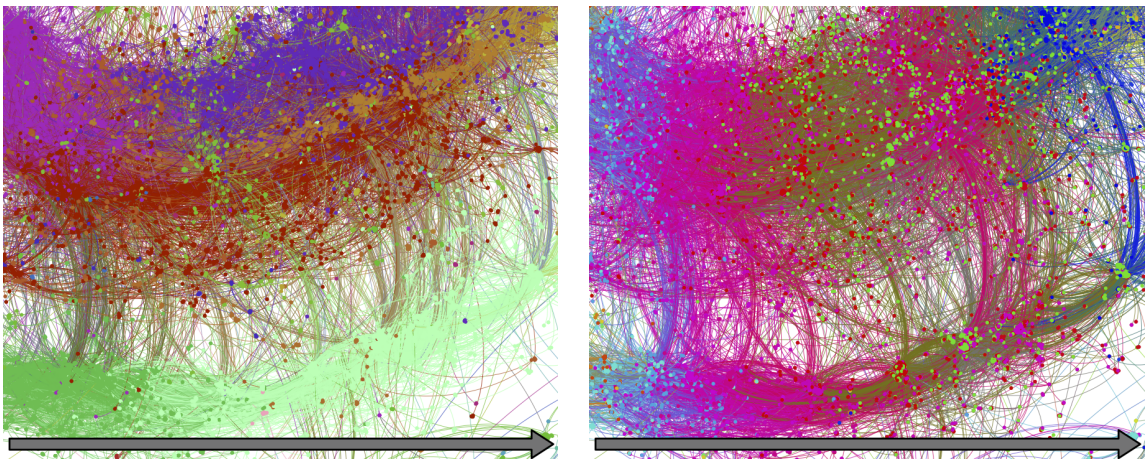


Figure 4.15 – **Zoom in a region of interest from the Higgs CMG of activity**. The arrow represents the evolution of time. Edge colors correspond to the color of the source nodes they connect to. The two images represent the same part of the network of Fig. 4.14 but colored differently. On the left, the graph is colored by communities (as for Fig. 4.14). On the right colors represent different and successive time steps of 10 min. The time increases following the color order: light blue, pink, brown, dark pink, dark blue (from left to right). These images reveal a part of the rich inter-community dynamics. On the left image, elongated communities evolve over time. We can see interactions between communities: some are tightly connected (dark pink, purple and red), some are weakly connected but active in the same time (green and dark pink or light green and purple). There exist bursts of connectivity between communities appearing over time (between the green community and the others for instance).

4.4 Efficient construction of the causal multilayer graph of activity

Now that we saw the usage of the method on 4 different examples, we describe the various steps to construct a scalable implementation of CMG of activity directly from the spatial graph, G, and the mask, $\mathbf{M}$. For clarity, several technical details aimed at optimizing further the

implementation, but which are not crucial to the understanding, are given in the next section.

We first combine the data into one object. Each binary activation vector, $M(i,\cdot)$, is stored as a property (label) on node i of G, creating a property graph which is still denoted G. The causal multilayer graph of activity H is created in a series of steps illustrated in Fig. 4.16 and detailed as follows:

1. Iterating over each edge, $e \in E_G$, of G linking a source node, i, and a destination node j, the algorithm first reads the vectors $M(i,\cdot)$ and $M(j,\cdot)$. An inter-layer connection is created between layer t and $t+1$ whenever i_t and j_{t+1} are activated (since we already know that i and j are spatial neighbors) i.e. $M(i,t) = M(j,t+1) = 1$. That is to say, an edge exists in Ω_x whenever $M(i,t)$ & $M(j,t+1) = 1$ where & is the bitwise And. We introduce a new vector u_e of size $T-1$ associated to the edge e, for all $t \in [0,\cdots,T-2]$:

   $$u_e(t) = M(i,t) \,\&\, M(j,t+1). \tag{4.7}$$

   The value $u_e(t)$ encodes the existence (1) or absence (0) of an inter-layer edge between vertices i_t and j_{t+1}. The vector u_e is stored as a property of edge e.

2. Since we are also interested in self activation of vertices across time (the set Ω_{xs}) we compute an additional vector $u_{\text{self},i}$ for each *vertex* i of G,

   $$u_{\text{self},i}(t) = M(i,t) \,\&\, M(i,t+1), \tag{4.8}$$

   for all $t \in [0,\cdots,T-2]$. The vector $u_{\text{self},i}$ is stored as a property of node i.

3. The construction of the network H is then done by reading the collection of edge vectors, $\{u_e\}_e$, node vectors, $\{u_{\text{self},i}\}_i$, and adding edges between successive time layers when ones are encountered.

The complexity of this process is linear in the number of edges and vertices of G. It is also linear in the number of time-steps. The scalability of our method comes from the properties of H. Layers are time-ordered, allowing connections between layers to be encoded as binary vectors. The creation of edges only depends on the state of pairs of nodes. Therefore, it relies on local values, which allows for, an efficient parallel implementation. The main task consists in handling the properties (vectors) of triplets, $\{i,e,j\}$, where $e \in E_G$ is the spatial edge connecting a source node, i, and a destination node, j. This is sufficient to create all causal edges between i and j ($i \neq j$). This attractive property is particularly suited to large scale graph analytics frameworks such as GraphLab Create[5] [152] or Apache GraphX [52], which have dedicated processes for applying functions to all triplets in parallel. Thus, our implementation gracefully scales with the number of cores and is much faster than a naive sequential implementation. To our knowledge, no other implementation of multilayer graphs matches the speed and the scalability of the method proposed here.

[5] https://turi.com/products/create/open_source.html

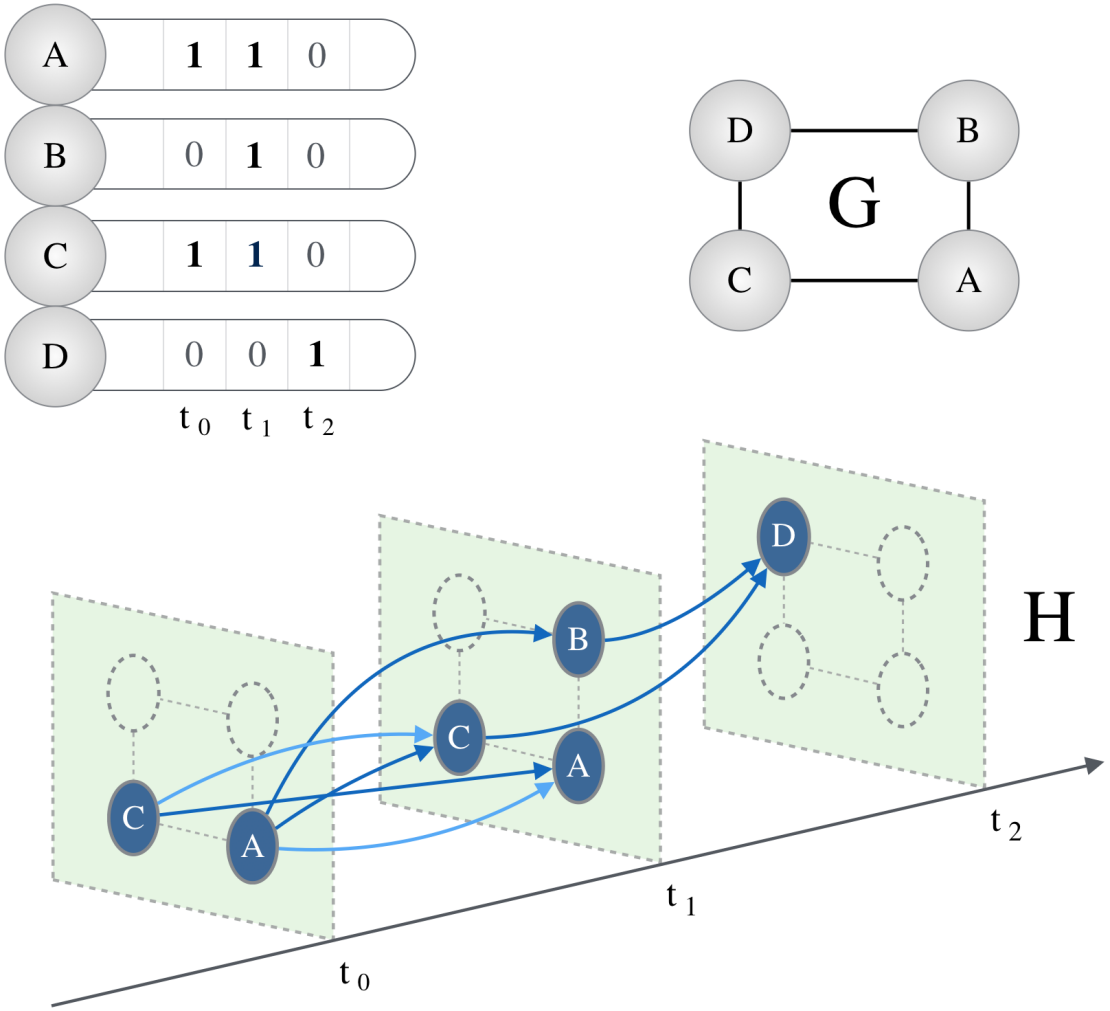


Figure 4.16 – **Actual implementation of the CMG of activity.** The CMG, H, is directly constructed from the activated entries of the binary mask (top left) and the original graph, G (top right). The construction of K is not needed.

4.4.1 Implementation details

This section describes additional implementation tricks for a better computer efficiency of the method. The reader interested in the general method more than its implementation may skip it.

The first step of the algorithm reads both activation mask vectors $M(i,t)$, $M(j,t+1)$ from i and j, respectively, as arbitrary precision integers [202]. An arbitrary precision integer can store any integer number (limited by available memory) and can be seen as a list of "standard" 32 bit integers with a common interface. GraphLab Create only allows the storage of basic data types as properties of nodes and edges such as integer, double, bool, and string (at the time of the writing). We had to transform the rows of our activation mask, M, to bitstring to be able

to store them on vertices and edges. Any bit compression algorithm can be used to reduce storage space. The arbitrary precision integer stored as ascii string offers a compression ratio of more than three over the raw bitstring where each 1 or 0 is stored as a character.

In the second step of the algorithm, the vector $u_e(t)$ is created by performing a logical And (&) between $M(i,t)$ and $M(j,t+1)$. It amounts to performing a logical And between two vectors: $M(i,\cdot)$ and the left-shift version of $M(j,\cdot)$ (hence involving a logical And and a bit shift). The last (least significant) bit in this operation is dropped (u_e is of size $T-1$) as it would correspond to a link between layer $T-1$ and T, the latter of which does not exist. Once the vector $u_e(t)$ is created, all the ones have to be found to create the causal edges. For each 1, its position in the vector gives the time t and allows the creation of two pairs, (i,t) and $(j,t+1)$, a causal edge, which is then added in the causal multilayer graph of activity H. Instead of looping through all the bits of $u_e(t)$ and checking for a 1 at each position, we have implemented another strategy that jumps from 1 to 1 in $u_e(t)$ and gives the position of the layer number t. This optimization is better than the classical For loop when $u_e(t)$ is sparse. The details are given in Algo. 1. We invite the interested reader to refer to [203] for more information on this low-level bit manipulation trick. The Fig. 4.17 illustrates the algorithm formally defined in Algorithm 1.

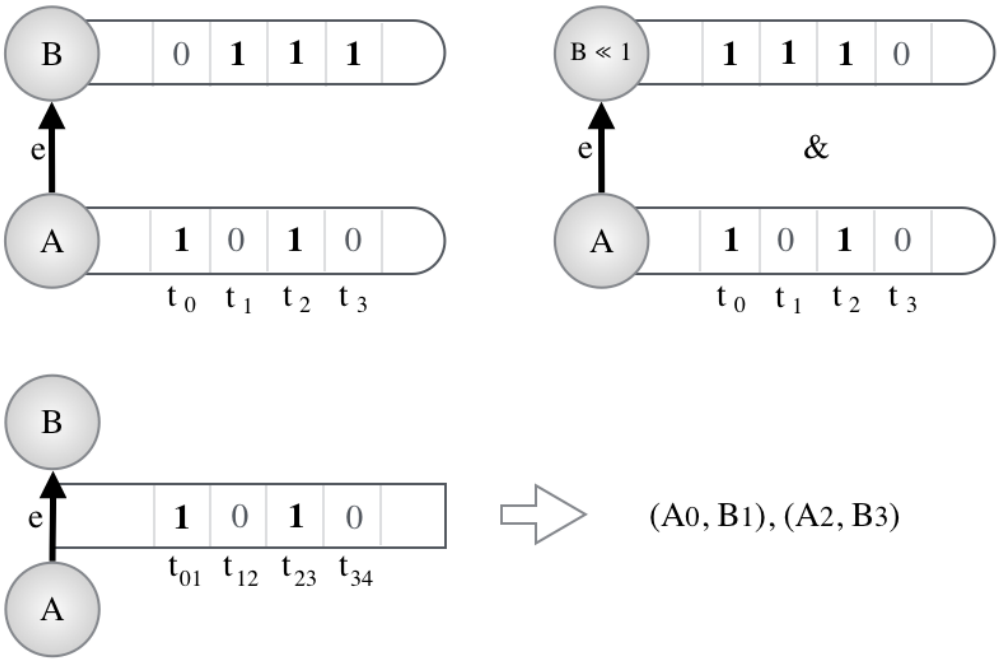


Figure 4.17 – **Creation of causal edges from a triplet** (A,e,B)**.** The binary mask vector of the destination node, B, is shifted (in this illustration left-shifted) to align source layer, t, and destination layer $t+1$. Then, a logical And is performed between the two vectors (top right picture). On the bottom figure, the result u, stored in e, is read to create the edges (and nodes) of H.

Input: Graph G having the binary activation vectors stored on its nodes
Output: Graph H

$H \longleftarrow$ create empty directed graph;
Parallel foreach *triplet* $(src, e, dst) \in G$ **do**
    $m_s \longleftarrow$ read src binary vector m as arbitrary-sized integer;
    $m_d \longleftarrow$ read dst binary vector m as arbitrary-sized integer;
    $u \longleftarrow m_s \,\&\, (m_d \gg 1)$;
    `// Find all activated causal edges`
    while u *is not* 0 **do**
        `// extract least significant bit on a 2s complement machine`
        $index \longleftarrow u \,\&\, -u$;
        $u \longleftarrow u \oplus index$ `// toggle the bit off;`
        `// Get activated layer number`
        $layer \longleftarrow -1$;
        while *index is not* 0 **do**
            $index \longleftarrow index \gg 1$;
            $layer \longleftarrow layer + 1$;
        end
        `AddEdge(`*H, (src, layer), (dst, layer+1)*`)`;
    end
end

Algorithm 1: Creation of the causal multilayer graph H on little-endian systems. The least significant bit is on the right.

4.4.2 The generalized causal multilayer graph of activity

In Sec. 4.1.3, we introduced the mathematical foundations of the causal multilayer graph of activity. Here, we generalize this model to account for dynamic spatial graphs, where edges or nodes are allowed to appear or disappear across layers. By encoding the position of nodes and edges as additional vectors on the nodes and edges of the generalized spatial graph G, the generalized CMG of activity can be constructed very efficiently using a variation of the algorithm introduced in Sec 4.4.

Let us assume we have a set $\{G_t\}_t$ of T graphs, one for each time-step $t \in [0, \cdots, T-1]$. The number of vertices and edges is allowed to change from graph to graph. Also, a mask M that associates a value (0 or 1) to each vertex of the set $\{G_t\}_t$ is given. In this case, it may not be a matrix. The mask can also be computed from a signal S on the vertices of the collection of graphs $\{G_t\}_t$.

We first create the generalized spatial graph G, which concatenates all $\{G_t\}_t$. A vertex i belongs to V_G if there exist some layer t such that $i_t \in V_{G_t}$. Similarly, an edge $e \in E_G$ connecting vertex i and j of G exists if for some t there is an edge $e_t \in G_t$ connecting i_t and j_t. Taking the definition of Eq. (4.2), the expression of the weights in Eq. (4.3) is $w_{ij}(t(t+1)) = W_{i,j}(t)$ for $i \neq j$, where $W_{i,j}(t)$ is the weight of the edge between i_t and j_t; For $i = j$, $w_{ii}(t(t+1)) = 1$ and

zero for the rest.

The second step associates vectors to each vertex and edge of G. Similarly to Sec. 4.1.3, a vector $M(i,\cdot)$ of length T is associated to each vertex i. Additional vectors, i.e., values of the signal S_i can also be stored on each vertex. In the case where vertex i is not present in some graph G_t the entry $M(i,t)$ exists and is set to zero. It is equivalent to adding an extraneous inactive and unconnected vertex i at layer t. We now introduce an additional mask: the edge mask M_E. It associates a binary vector of length T to each edge of G i.e.: for each edge e of G between node i and j, $M_E(e,t) = 1$ if there is an edge between i_t and j_t, zero otherwise. Notice that edge weights can also be stored as vectors on each edge, similarly to what is done to the signal S on vertices. For the sake of simplicity, we assume the graphs to be unweighted here.

For an efficient construction, we modify (4.7) to account for the existence of edges between layers. It becomes:

$$u_e(t) = M(i,t) \;\&\; M(j,t+1) \;\&\; M_E(t). \tag{4.9}$$

In the case of self-edges, Eq.(4.8) is unchanged: two vertices will be connected if they exist in two successive layers and are active.

Fig. 4.18 explains the construction of the generalized CMG of activity using the different vectors. Similarly to the standard model, for each edge e the destination vertex mask is shifted before performing a logical And with the source vector. The only difference is the introduction of the edge mask M_e which is logically "Anded" between the source and shifted destination masks. We use the same process at the vertex level to account for self-edges (connections between i_t and i_{t+1}).

The rest of the construction leading to H is similar to the simplified case. Once H has been constructed, the creation of feature vectors and the clustering are identical.

4.5 Discussion

In this chapter, we have presented a new framework to extract and analyze sparse repeated patterns created by dynamical processes on graphs as illustrated in Fig. 4.19. Based on the concept of merging graph and data altogether, our approach relies on the causal multilayer graph, a new multilayer graph structure that encodes the propagation or spreading of events across time. In addition of providing a mathematical framework linking with previous theoretical works on multilayer graphs, our implementation is computationally efficient and handles billions of nodes on a single commodity server.

Moving on to the applications, we have demonstrated in four entirely different real-world examples that the study of dynamic activation components and the clustering of similar activity patterns reveal new insights on the underlying causal processes previously hidden in

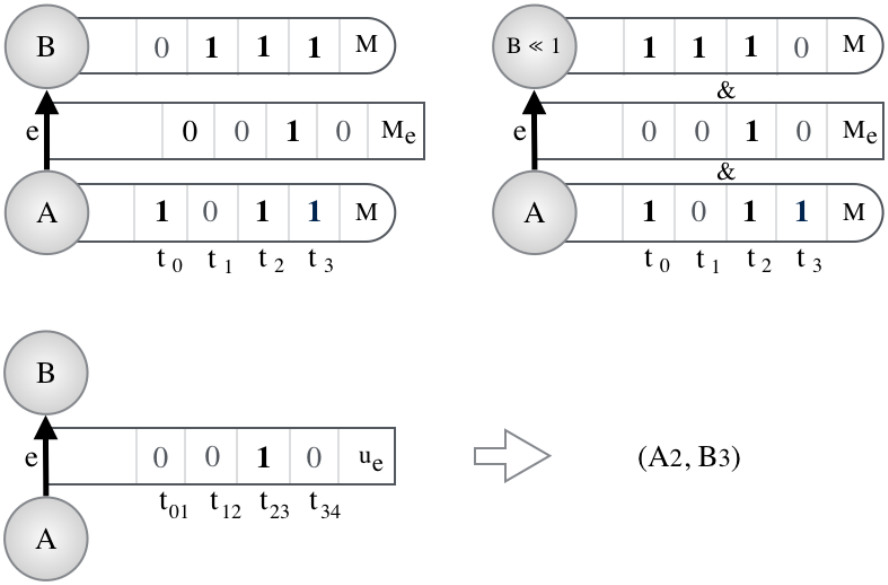


Figure 4.18 – **Creation of causal edges in the generalized CMG of activity.** Top left: a source node A and a destination node B connected by edge e, together with their respective mask. Top right: the binary mask vector of B, is shifted (in this illustration left-shifted). Then, a logical And is performed between the three vectors. On the bottom figure, the result u, stored in e, is read to create the edges (and nodes) of H. In this example, the pair (A_{t_0}, B_{t_1}) is disconnected because the first entry of M_E is 0. Only one edge between $A_2 = A_{t_2}$ and $B_3 = B_{t_3}$ is created.

the data. A possible improvement of our method could nevertheless come from the activation threshold that determines whether a node is considered active at a particular layer. Sole parameter of the method, its tuning requires a good knowledge of the data. We could envision putting constraints on the expected shape of patterns for example, and automatically adjust the threshold to fulfill them. Another future work could include the application of our method to temporal networks, allowing a different approach as well as an additional degree of model complexity.

Coming back to our original idea of the evolution of taste in Genezik, we have also highlighted how the study of one particular problem has led to the development of a new general framework for network science with possible applications in spreading of epidemic outbreaks, social network activity or any sensor networks. Indeed, applications where time series have been recorded on the vertices of a network are numerous, present in many fields of science such as engineering, social, biological, physical or computer science and keep increasing with the current data deluge. However, it is surprisingly difficult to access sufficiently detailed time series as they are most often kept private by companies that do business out of it. If we leave aside technical difficulties to store and access live feeds for research, data unavailability limits the expansion of network science to uncharted domains that could in turn benefit to the original data providers such as Twitter or Google. Besides, the usage of private APIs for

research may introduce a bias with unforeseen consequences especially when dealing with social interactions or human behavior [204].

Based on this fact, we see that if we wish to apply our method to push forward our understanding of human interactions as we have done with the Higgs example, the dataset should be free and open to anyone to be as unbiased as possible. In addition, it should be sufficiently large and accurate so that our findings are representative of the large class of the population.

How can we then find and process this dataset? Most importantly, what are the findings that we can extract using the causal multilayer graph of activity? We answer these questions in the next chapter.

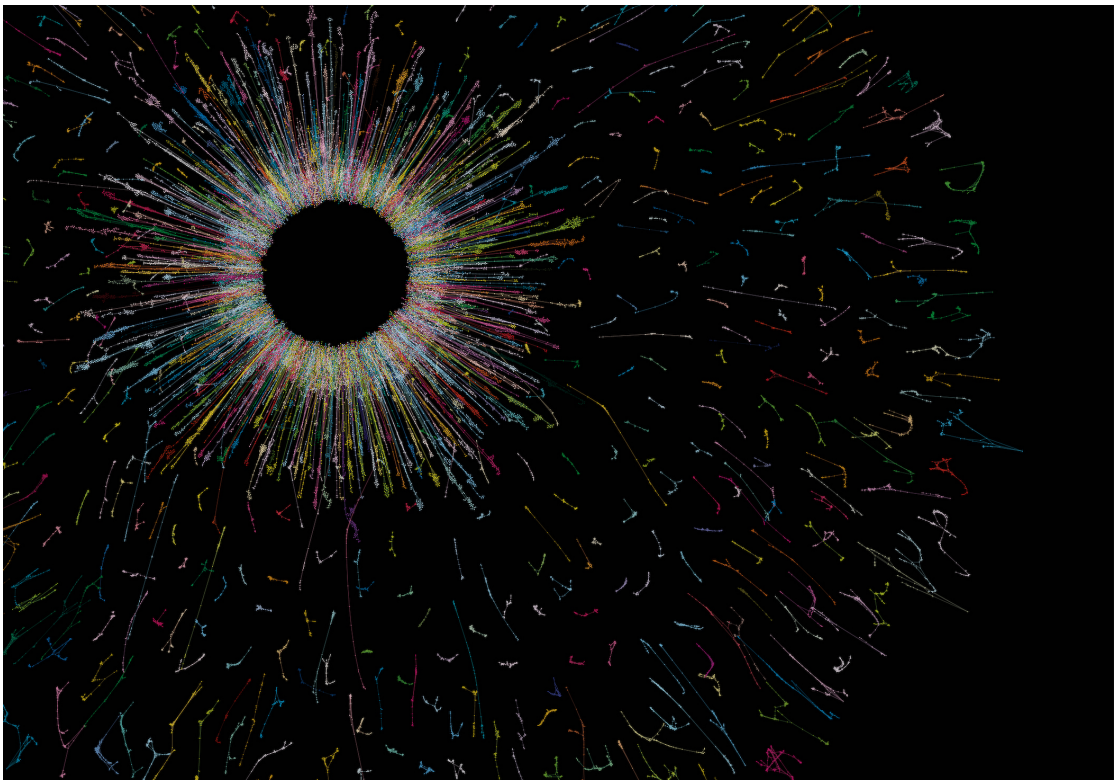


Figure 4.19 – **On time - 2014. Copyright Kirell Benzi.** The train station in Lausanne is filled with cameras that anonymously track people walking through its corridors. Used to estimate the congestion in the station, the analysis of average pedestrian trajectories is crucial to the improvement of the life of thousands of daily commuters. To do this analysis, we divide the train station into small regions of around one square meter linked together if they are adjacent on the ground. Using more than 42 million pedestrian positions, we determine when a region is sufficiently crowded at a given time and record the propagation of the congestion on the network. It yields to the creation of thousands of disconnected sub-networks, here colorized on the image. If you have ever been to the Lausanne train station, your trajectory is probably contained in what you see!

5 Social behavior analysis with network science

An increasing number of data-driven studies based on the analysis of dynamic activity broadens our understanding of social behavior. In the era of "big data", the formidable amount of data available to researchers allows for large scale studies uncovering social processes that were not be possible or even conceivable a few years ago. Fueled by the rich literature on epidemic processes over networks coming from statistical physics [205], one of the main research topics in this area has focused on the mechanisms involved in the diffusion or sharing of information [206, 207]. This applies to the analysis of disease or rumor spreading [147, 208] as well as other aspects of information spreading, for example, the significant influence of our social circles [209, 210, 211] or the identification of influencers in networks [212].

Building on the method we previously introduced, we here propose a novel approach to social behavior analysis that partly reveals how knowledge is structured by observing the collective activity of millions of Wikipedia visitors. To do so, we propose to use the network of knowledge provided by Wikipedia together with the visit of users on each page over time. Combining the network and its activity appears as a crucial choice here. Firstly, the network brings a strong structure to the data. It has been shaped by humans, and its organization is believed to optimal for human learning because the network architecture is *de facto* related to our way of processing information [213, 214]. Secondly, and of equal importance, user visits and their dynamic activity shed light on the particular interest of people over time, which is complementary to the network information. The combination of both aspects should thus lead to new insights on human information retrieval and structuring.

Choosing to study Wikipedia here is a logical choice alleviating all our previous interrogations on data availability and trust. Indeed, Wikipedia is broad, reliable, neutral and most importantly a free source of information [215, 216, 217, 218] in addition to being one of the most visited websites in the world. Thanks to its openness and its 40 million pages in total, it has become the center of interest of many studies in a wide range of different fields of research [215, 218]. Interestingly, a website has even been dedicated to the tracking of academic studies on Wikipedia [219]. However, the collective exploration of Wikipedia pages in its dynamical aspect is largely unknown. Questions about the time evolution of the knowledge

network and patterns of user visits have recently started to raise interest, e.g. the suggestions of new reference links in pages [220, 221].

To go beyond existing works on Wikipedia, we here propose a novel network science method that creates small networks of interconnected pages of Wikipedia that have received a large enough number of visits during the same timespan. Based on the causal multilayer graph of activity, these connected networks named Wiki-souvenirs, emerge as representative entities associated with events or news that have attracted the interest of people at some point in time. From these Wiki-souvenirs, we then design an algorithm, the deep recollection, that can validate the veracity of what we extract and pinpoint precisely for uninformed users what to read to catch up with the news.

We hypothesize that the study of what is collectively browsed from thousands of past events reveals some of the most fundamental aspects of human nature such as curiosity. We also hypothesize that curiosity is guided by rules that optimize the collection, storage and retrieval of facts in memory.

To confirm our hypothesis, we structure this chapter as follows. First, we present how to build Wiki-souvenirs (WS) from the causal multilayer graph of activity. We exhibit some of their properties and demonstrate their relevance to track any major event happening in real-time worldwide with our deep recollection algorithm.

In a second part, we propose to characterize some of the rules that could drive human curiosity by performing a data analysis of WS. We then monitor how Wiki-souvenirs connect to each other through time, witnessing the emergence of a structured knowledge graph that is different from all known classes of networks. The unique architecture of this network called Wiki-memory reveals the existence of common rules guiding the collection, storage and retrieval of information based on human curiosity.

Finally, we conclude this chapter by presenting our ongoing work on the construction of an original contextual search engine using Wiki-souvenirs, the Wiki-memory, and the deep recollection. Originally envisioned to replace the Wikipedia internal search system, we present the challenges of its implementation for a much broader usage that could be of great interest to the general public, historians, journalists, sociologists and marketers alike.

Our most significant contributions in this chapter are as follows:

- the application of network science to perform a large-scale analysis social behavior analysis;
- a novel network science method that extracts contextualized networks following news and event happening in real-time;
- the characterization of some of the rules guiding human curiosity drawn from the observation of billions of visits over the Wikipedia hyperlink network;

- the presentation of a new class of network, the Wiki-memory, shaped by human activity different from random or scale-free graphs associated with social networks;

- the introduction of a new contextual search engine, first of its kind, based on Wiki-souvenirs, the Wiki-memory and the deep recollection process that could replace Wikipedia internals entirely.

5.1 Structuring social activity on Wikipedia

5.1.1 Extracting dynamic activation components

To explore social activity, we associate a network of all Wikipedia English articles linked by their hypertext references (the Wiki-graph) with the hourly, daily and weekly number of visits, given for each page. To alleviate the scale issue and to be able to process billions of data points collected over a period of 7 months from October 2014 to April 2015, we rely on the causal multilayer graph of activity introduced in the previous chapter [8].

The first step of the method consists in creating the graph of Wikipedia articles, the Wiki-graph. To do so, we gather Wikipedia SQL dumps [222] for the English language of April 2015 and filter them so as to keep only regular articles. They form the nodes of our graph. Note that category pages are not used here contrary to what we did in the Star Wars analysis in chapter 2. Intra-wiki links between articles, the edges of our graph, can also be given by another Wikipedia SQL database. However, we must be careful to take into account all the redirects from old articles and remove them from the Wiki-graph. Additionally, we also remove nodes with a very high in-degree (above 8,000) such as "Global Positioning System (GPS)" or the Main Page of Wikipedia in order to get a more meaningful graph and facilitate the analysis. The deleted nodes only represent a set of 1,071 nodes over a total of 4,856,639 (0.02%).

The second step required to build the causal multilayer graph of activity (CMG) consists in collecting Wikipedia visit logs to measure the activity over the vertices of the Wiki-graph. Using the hourly visit dumps given by Wikimedia[223], we sum the mobile and desktop visit counts for Wikipedia English pages. Because of the way the visits are collected, redirected pages have their entries in the logs. To be accurate, we add these visits to the counter of real non-redirected pages that users see when accessed. To obtain visit counts for the day, we simply take the hourly logs from midnight to midnight and sum them. We proceed similarly to get weekly visit counts. Capturing user activity in different time-scales allows observing precisely how users interact with pages on average as well as finding global trends in the data if the scale is coarser.

The causal multilayer graph of activity is designed to account for causality in dynamical processes over networks. In our case, it is used to detect events formed by a significant number of visits to the pages of the Wikipedia network. The method creates a new graph that connects active nodes to their neighbors' counterparts on the Wiki-graph in the next time step if they are

both considered as active. We choose arbitrarily to consider a node as active if its hourly visit count is superior to 1,000. Note that the average hourly visit count for pages of Wikipedia is of 32 visits. To obtain the day and week scale threshold, we simply multiply the hourly threshold by 24 and 24×7 respectively. By extracting the weakly connected subgraphs of the CMG, we obtain spatio-temporal patterns of activity, the dynamic activation components. They encode the dynamics of how users collectively interact with pages of the Wikipedia network in time.

5.1.2 Crafting Wiki-souvenirs

Until now, we have simply applied our method to yet another dataset. The crafting of Wiki-souvenirs requires extra steps from the dynamic activation components (DACs).

From a DAC, the next step consists in folding the time dimension to obtain its corresponding subgraph of pages on the Wiki-graph. These patterns are called static activity components (SACs). For instance, if a page A is activated 3 times in a DAC of 5 layers with 3 nodes A_1, A_2, A_5, the folded node is simply A with the total number of visits from these 3 nodes. In the folding process, we also count how many times the pages and links between pages are activated through time. After normalization, each node or edge of the static activity component has a score corresponding to its likelihood of activation in the dynamic activation component.

A common limitation of some network science methods comes from the topology of the graph. If the studied graph resembles a small-world network [224] similar to the Wiki-graph [225], it often becomes intractable to deploy algorithms or to isolate useful parts of the network. In our case, the most visited static activation components can include a vast number of nodes that correspond to different world events.

To isolate individual patterns, we filter in-edges and out-edges of the SAC if their likelihood of activation less than 5% in its dynamic counterpart. It is similar to what we did with average activation components in the previous chapter. The SAC thus naturally decomposes into smaller static activity patterns, that may still comprise different events. Our last step consists in extracting communities on the static activity patterns using a stochastic block-model algorithm[226]. The resulting communities, small connected and directed subgraphs of activity are what we call Wiki-souvenirs (Fig. 5.1).

In Fig. 5.1.A, we observe, at day scale, user activity around the propagation of Ebola in the USA from October 13th, 2014 to October 19th, 2014. The main page in orange links to the medical description of the virus, the list of outbreaks as well as the first Ebola patient diagnosed with Ebola, Thomas Eric Duncan. We also see that user visited the page of Ron Klain, the Ebola response coordinator appointed by Barack Obama on October 17th in the middle of the duration of the WS, as well as the page of Thomas R. Frieden, the Director of the U.S. Center for Disease Control and Prevention. The last page of this WS concerns a Hollywood movie about humans infected by a virus, Outbreak.

The second example in Fig. 5.1.B concerns Halloween's night in 2014 at the hour scale. We

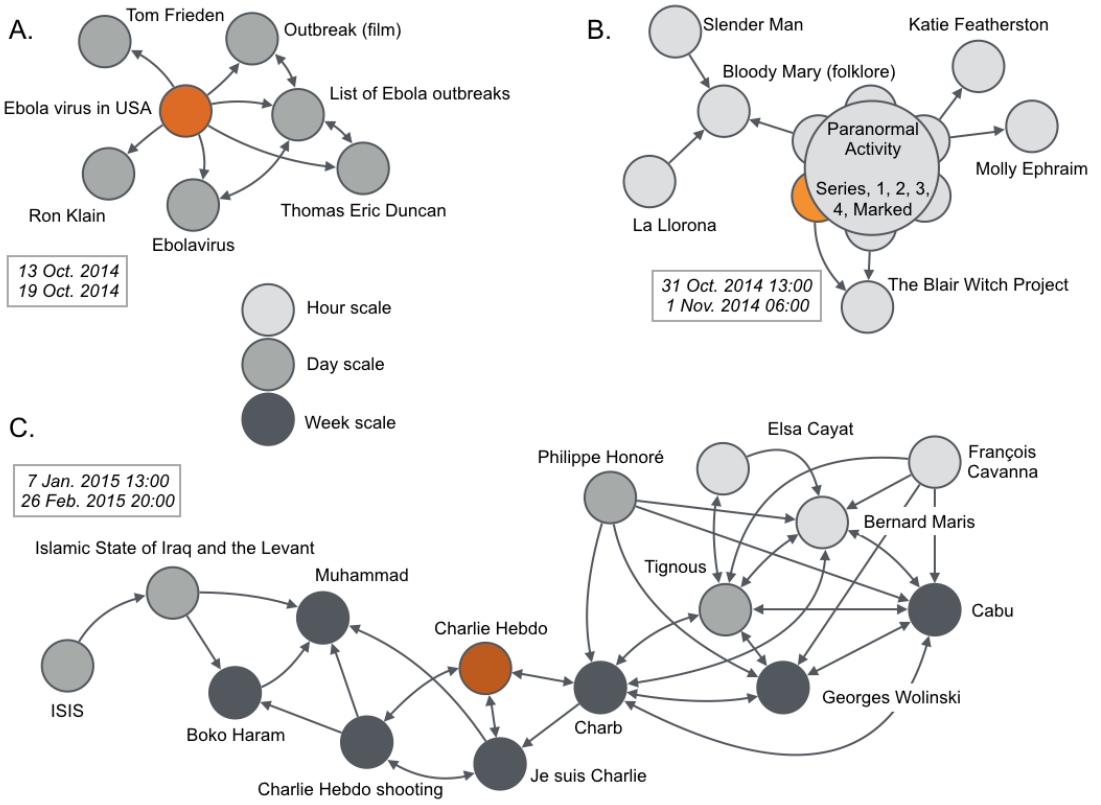


Figure 5.1 – **Examples of Wiki-souvenirs extracted from collective activity on Wikipedia.** For each event, the orange node is the main page, with the highest number of visits. Node colors show which nodes stay activated when changing the time scale. They are colored from light gray (only active in the hour scale) to mid gray (active at the hour and day scale), to dark gray (active on the three scales). **A,** Ebola virus in the USA. **B,** a Wiki-souvenir from Halloween's night in 2014. **C,** The Charlie Hebdo terrorist attack.

observe that the core of WS is composed of all the Paranormal Activity movies and deduce that Wikipedians are looking to get scared for Halloween's night. It is confirmed by the other pages in the WS with La Llorona, the Weeping Woman, or the ghost Bloody Mary for example. Notice how precise this WS is, starting on October 31 at 13h00 and ending on November 1 in the morning.

Our last example of WS in Fig. 5.1.C depicts the Charlie Hebdo terrorist attack in Paris on January 7th,2015. In this example, we show the influence of the different time scales on the same WS by coloring the nodes in three shades of gray from the day scale in light gray to the week scale in dark gray. The coarser scale includes the nodes from the finer scales. In the center, we can find key pages of the event, with Charlie Hebdo, Charlie Hebdo shooting as well as the famous Je Suis Charlie slogan that arose just after the shooting. On the right side, we can find most of the staff of the satirical magazine whereas on the left side we find pages related to terrorist organizations.

As we see in Fig. 5.1, it appears that Wiki-souvenirs naturally encode the context at a given point in time and reflect the collective interest of humanity for world-events/news. They also capture very precisely and in a sparse manner the complex relations between pages that have attracted the interest of users. The graph structure acts as a filter, connecting only simultaneously active pages that are related to a link.

Additionally, we note that the WS forms an efficient memory structure, since the collection of all Wiki-souvenirs for a whole year can be stored on a few megabytes whereas the memory requirement for the full dataset is two orders of magnitude greater.

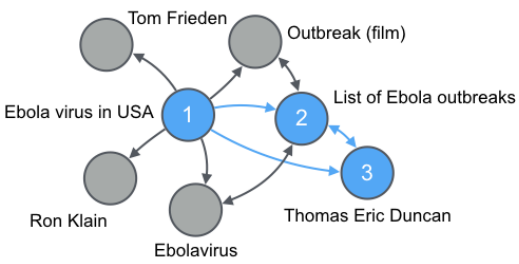
5.1.3 Deep recollection

By definition, a souvenir is an object that triggers a recollection of memories associated with it. By analogy a Wiki-souvenir is an object, representative of the average browsing behavior of users on past events, that can be used to bring back both semantic and contextual information when accessed. The deep-recollection leverages the complex graph structure of Wiki-souvenirs to fetch from the Internet all relevant information such as web-pages, news, images or social feeds that could describe what is encoded in a souvenir as we illustrate in Fig. 5.2. This process can thus automatically provide, based on user visits, a smart digest of what is relevant in the context. It also gives a deep semantic meaning to edges of a souvenir that cannot necessarily be explained without knowing the context. In this work, we use the deep recollection as a validation tool to verify by hand the accuracy of the relationships we extract in WS. We report 100% accuracy over the top-10 stories of the yearly review of Google Trends.

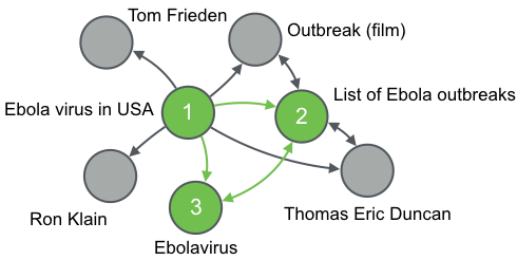
To perform the deep recollection, we first extract representative keywords that could describe the page for a regular Google search. In our case, we extract keywords from the page titles, by removing stop words and punctuation. Then, we identify the most visited cliques in the WS and order them by the number of views. These groups of nodes form the largest fully weakly-connected subgraphs of the Wiki-souvenir. We assume that they encode a particular aspect of the event. Finally, we perform the search for each group of keywords sorted by importance and restricted to date interval of the WS (Fig. 5.2).

We see that our deep recollection process takes into account context, network structure as well as the popularity to fetch every relevant aspect of the same WS. The list of results pinpoints exactly what one needs to read to get informed on this particular part of the subject.

Note that a more complex ranking scheme can also be adopted, taking into account the PageRank of nodes weighted by their number of views. However, it is difficult to assess if this more complex scheme yields significant changes in the results.



Ebola virus in USA List of Ebola outbreaks Thomas Eric Duncan
http://www.cdc.gov/vhf/ebola/outbreaks/2014-west-africa/united-states-imported-case.html
http://www.bbc.com/news/world-africa-28755033
http://www.cnn.com/2014/10/17/us/ebola-us-list/
http://www.cnn.com/2014/04/11/health/ebola-fast-facts/



Ebola virus in USA List of Ebola outbreaks Ebolavirus
http://www.cdc.gov/vhf/ebola/outbreaks/history/chronology.html
http://apps.who.int/ebola/ebola-situation-reports
http://healthintelligence.drupalgardens.com/content/chronology-ebola-virus-disease-outbreaks-1976-2014
http://ecdc.europa.eu/en/press/news/_layouts/forms/News_DispForm.aspx?List=8db7286c-fe2d-476c-9133-18ff4cb1b568&ID=1097

Figure 5.2 – **Some examples of deep recollection for the Ebola Wiki-souvenir.** Left: the WS and a few selected nodes highlighted (colored in blue and green). These nodes form the two largest fully weakly-connected subgraphs of the Wiki-souvenir. Right: table of search results on Google News associated with the keywords contained in the highlighted nodes.

5.2 Characterization of human curiosity on Wikipedia

Social science researchers are relentlessly seeking to improve the reproducibility of their research [227, 228, 229] as it is one of the key principles distinguishing science from non-science. Known as the demarcation problem [230], this eternal philosophical question has fueled advances in social science with the introduction of Grounded Theory, for example, [231].

In this era of big data, we see the recent advances of scientific tools and frameworks as a formidable opportunity to push forward social sciences by studying colossal samples of data. While there is a bias in our study of Wikipedia because it only represents people with an uncensored Internet access who speak English, the billions of visits are significant enough to be meaningful. More importantly, this work is reproducible.

Based on the analysis of Wiki-souvenirs we notice the existence of a common behavior characterizing human information retrieval: curiosity is instinctively driven by some underlying mechanisms. We focus now on the identification of these mechanisms and validate our initial hypothesis.

5.2.1 Media pressure bounds the space of curiosity

Intuition may suggest that random pages of Wikipedia would be visited in a significant manner as the result of pure random curiosity. We have shown, on the contrary, that all Wiki-souvenirs correspond to world events (e.g. Charlie Hebdo) or local news (e.g. a concert of Queen on New Year's Eve). Because they are directly correlated to the media coverage, we can affirm that the media pressure bounds the space onto which personal curiosity lives. The media coverage also influences the content of the Wikipedia page itself. As we see on the Charlie Hebdo WS or Germanwings example in Fig. 5.3, the Wikipedia community is very reactive to create or update pages as soon as something happens. Finally, the media are also driven by humans seeking to relay information based on what people want to know, hear and see creating a vicious circle.

5.2.2 Curiosity as a mix between lack of information and recreational roaming

Looking at the distributions of pages in Wiki-souvenirs, we see that inside a limited space around the main source of visits, different aspects of the same event are highlighted in people's visits. Indeed, WS extracted at the day scale are composed of 4.88 pages with only 56% of all visits on the most visited page. The diffusion of user clicks around the central page can be due to either the lack of information on the page or simply as a consequence of a "recreational roaming". In fact, it is due to both, as we show in the study of the Germanwings crash Wiki-souvenir in 2015 (Fig. 5.3).

At the beginning of the event on March 24, the Germanwings crash page does not exist. Hence, an unusual flow of users goes to the 'Accidents of A320' page seeking information about the crash. Indirectly, the page dedicated to the A320 Indonesia Flight crash in 2014 is also activated. During the first day, they keep a high number of visits while the main WS page is created and additional info is added.

On the contrary, we observe that pages about musicians M. Radner and O. Bryjak who died in the crash, did not take an active part in it. Because these pages contain no information allowing to understand the event, the significant number of visits here results from the recreational roaming.

The same behavior can also be witnessed in the activation of past films in the Paranormal Activity example of Fig. 5.1.B or the activation of previous movies of the Jurassic Park series following links from Jurassic World (the latest film to date).

5.2.3 Gossip and human relations drive curiosity

Despite a significant number of links presents in the pages, 60.62 links on average for the whole Wikipedia, only a few pages are highly visited as we have seen previously. To have better insights on visitors tastes, we characterize their topics of interests by performing a statistical

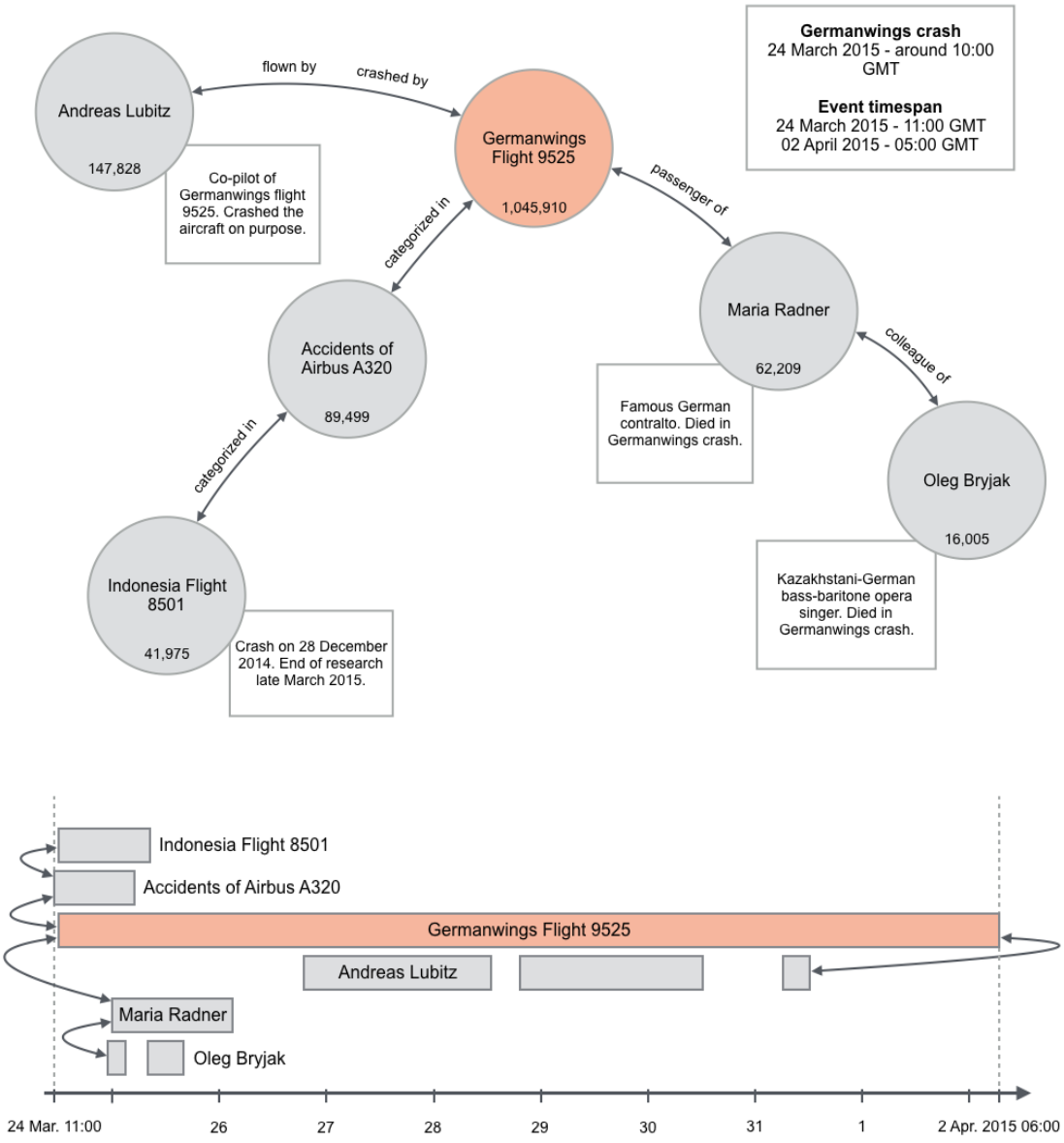


Figure 5.3 – **Germanwings crash Wiki-souvenir and its timeline.** Each node represents a Wikipedia page, the label of the node being the title of the page and the associated value being the total number of visits during the event time span from March 24 to April 2. Edges of the Wiki-souvenir represent the hyperlinks between pages enriched with ontologies from an external semantic database. The timeline of the crash shows the evolution of user activity. It is worth noting that the main page appeared at 11:02 UTC on the 24th and was highly visited less than two hours after the crash, which shows the reactivity of the Wikipedia community.

analysis on the Wiki-souvenirs. To do so, we choose keywords from selected themes and monitor their appearances in the categories associated with each page of the WS. To be as accurate as possible, we weight the importance of a page according to its number of visits so that the mixtures of topics of each WS is proportional to the popularity of each page. We present the results of this classification in Fig. 5.4.

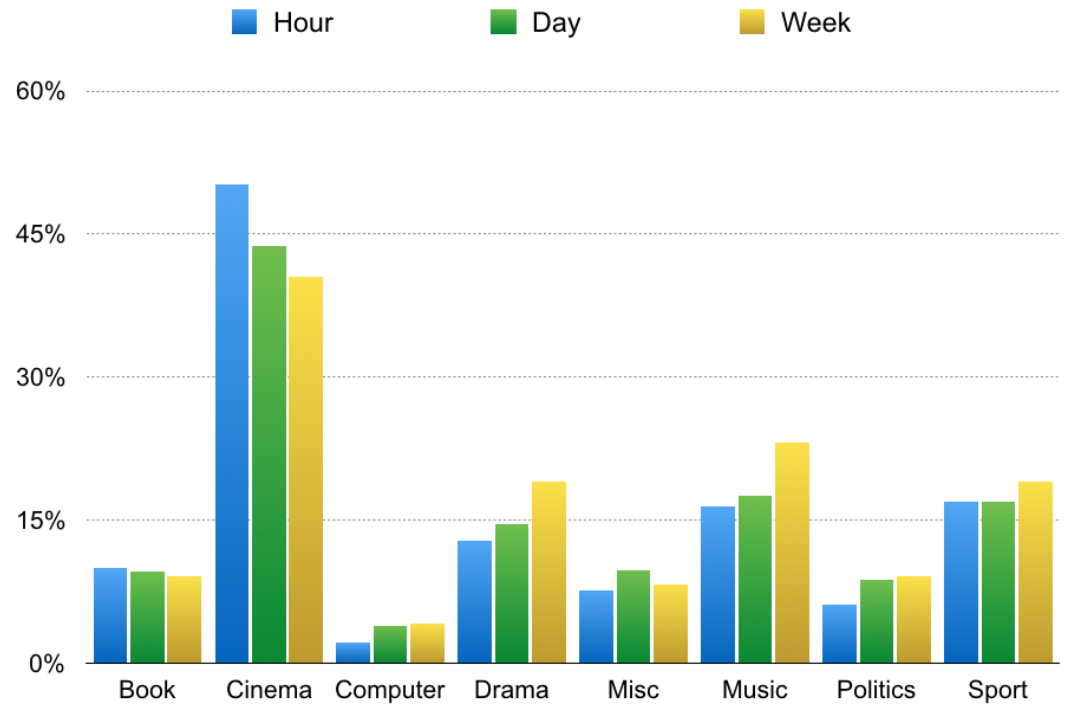


Figure 5.4 – **Topic repartition for Wiki-souvenirs.** Percentage of Wiki-souvenirs containing keywords related to selected themes, for different scales. A Wiki-souvenir can be classified in several themes. The misc category contains all the unclassified events.

Although it is presented as a source of general knowledge, strikingly, the majority of events on Wikipedia are in fact related to the entertainment industry. Wikipedia thus appears as a trusted site of choice to get the latest information on films and series. Each blockbuster has triggered an event by creating an enormous amount of visits on the page related to it, to its actors/actresses or the past aired episodes in case of TV series. As an example, our most visited WS from our period deals with “Fifty Shades of Grey” books and film with more than 25 M visits.

We also observe that people are fascinated with events related to death or sexual scandals with an extremely high number of visits on the “Celebrity photo hack” of 2014, the Ferguson shooting or the assassination of Boris Nemtsov for example. These findings are coherent with several behavioral studies on the subject [232, 233].

We bring further evidence of the vivid interest of humans for entertainment and tragedies in the analysis of the most common ontologies associated to the edges of Wiki-souvenirs. These

ontologies, extracted from the DBpedia database, describe relationships (if any) between connected pages of a Wiki-souvenir. Curated by humans or semi-automatically, the DBpedia database proves invaluable to access structured information of Wikipedia programmatically [234]. We additionally enrich the DBpedia with the content of Base KB, a state-of-the-art semantic database with more than 1.5 B ontologies built on top of FreeBase [235], mostly known as the open version of the Google knowledge graph.

Overall the ontologies visited by users on all scales, around 43% are devoted to entertainment. The peculiar repartition of Wiki-souvenirs confirms the social studies [236, 237] stating that most discussions are dedicated to social topics. In fact, close to 50% of the pages of all Wiki-souvenirs are about people independently of the scale (hour, day, week) or the main topic of the event. The ontologies related to family relationships such as "married to, son of, spouse, sibling" also represent around 20% of all ontologies highlighting the importance of social relations for humans.

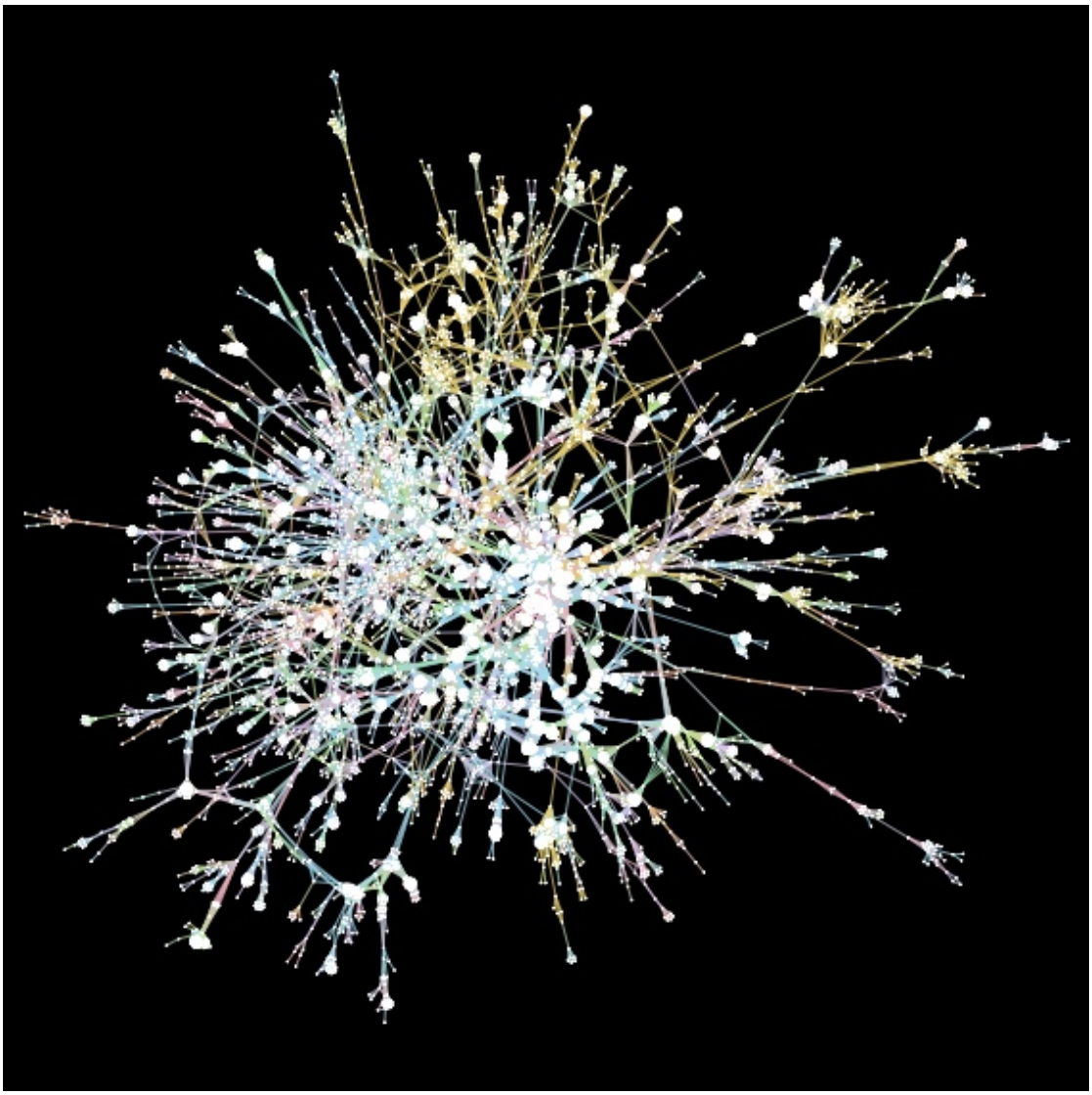
5.3 Emergence of a new kind of collectively-structured network

To summarize what we have uncovered so far, human curiosity is bounded by the media coverage and alternates between the search for factual information and recreational roaming over the pages of the network. Lastly, we have shown that visits are not random over the main page of an event. Indeed, they are driven by a conscious or unconscious intent to seek social information that includes gossip and human relationships.

Recalling that Wiki-souvenirs capture very precisely the context at a given time and correspond to what is collectively remembered from an event, we suspect that human curiosity also shapes the way WS relate to another. More precisely, we hypothesize that human curiosity tends to form links between past events to organize and structure information.

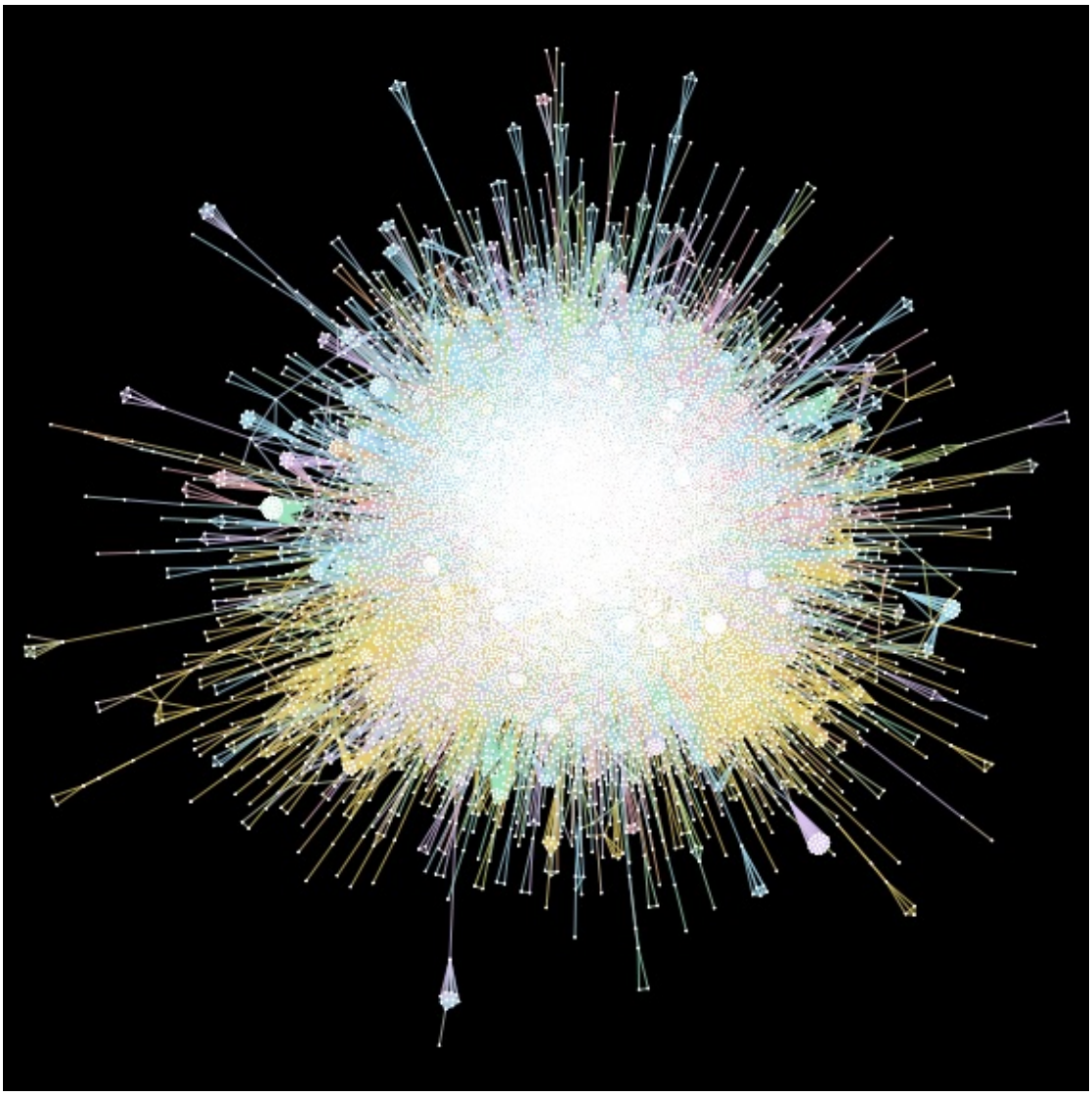
To verify our hypothesis, we build a graph of Wiki-souvenirs, called Wiki-memory, where WS are connected if they share at least a common Wikipedia page. This "artificial memory" created in a natural manner should exhibit unique properties derived from the very structured nature of WS. Additionally, we create another graph called random-memory based on generated Wiki-souvenirs. These fake WS are created by mimicking the shape of a real Wiki-souvenir while picking connected nodes around the most visited page of a WS at random. They thus simulate random curiosity around the main page while having the same distribution of nodes and time span. We expect the random-memory to follow the properties of the Wiki-graph which is known to be a scale-free network [225] as the random-memory is created by randomly sampling neighbors.

To give the reader intuition, we display in Fig. 5.5 visualizations of the two memories using the same layout algorithm. We first notice the differences between the hairball shape of the random-memory, a classic visualization of random networks, compared to the very dense packs of nodes connected sparsely together over a wider surface in the Wiki-memory.



(a) Wiki-memory

Looking at the evolution of the number of nodes in both memories in Fig. 5.6a, we observe striking differences in the growth of the giant connected component over time. Contrary to the random-memory that grows progressively over time revealing a typical behavior of real-world networks [238], our Wiki-memory is scarcely connected at the beginning but then, suddenly, encounters a critical phenomenon after four months. The evolution of random graphs is known to have sharp transitions on their statistical properties depending on their construction rules [239, 240]. Depending on the probability of connection between nodes, asymptotically, the graph can become a set of small unconnected components or witness the appearance of a giant component. However in our case, the probability of connecting a node stays constant, and the number of nodes and edges grows almost linearly as we show in Fig. 5.6a and Fig. 5.6d, revealing that the Wiki-memory does not follow the properties of a random network. Strikingly, the Wiki-memory possesses always more edges than the fake one,



(b) Random memory

Figure 5.5 – **Wiki-memory and random-memory at the hour scale colored by main topic** The colors are as follows: cinema pale blue, sport yellow, music pink, drama purple, book, green, politics blue, computer olive. **a,** Wiki-memory. **b,** Random-memory.

yet the giant component appears much later.

Another piece of evidence for the emergence of a new very organized structure can be found both in the evolution of the diameter of the giant component in Fig. 5.6b. Instead of slowly decreasing as expected similar to the random-memory [238], the diameter of Wiki-memory jumps to 37 hops after the critical phenomenon. Finally, the evolution of the average betweenness centrality of both memories in Fig. 5.6c confirms the particular structure of the Wiki-memory. The average number of shortest paths passing through the nodes of the giant

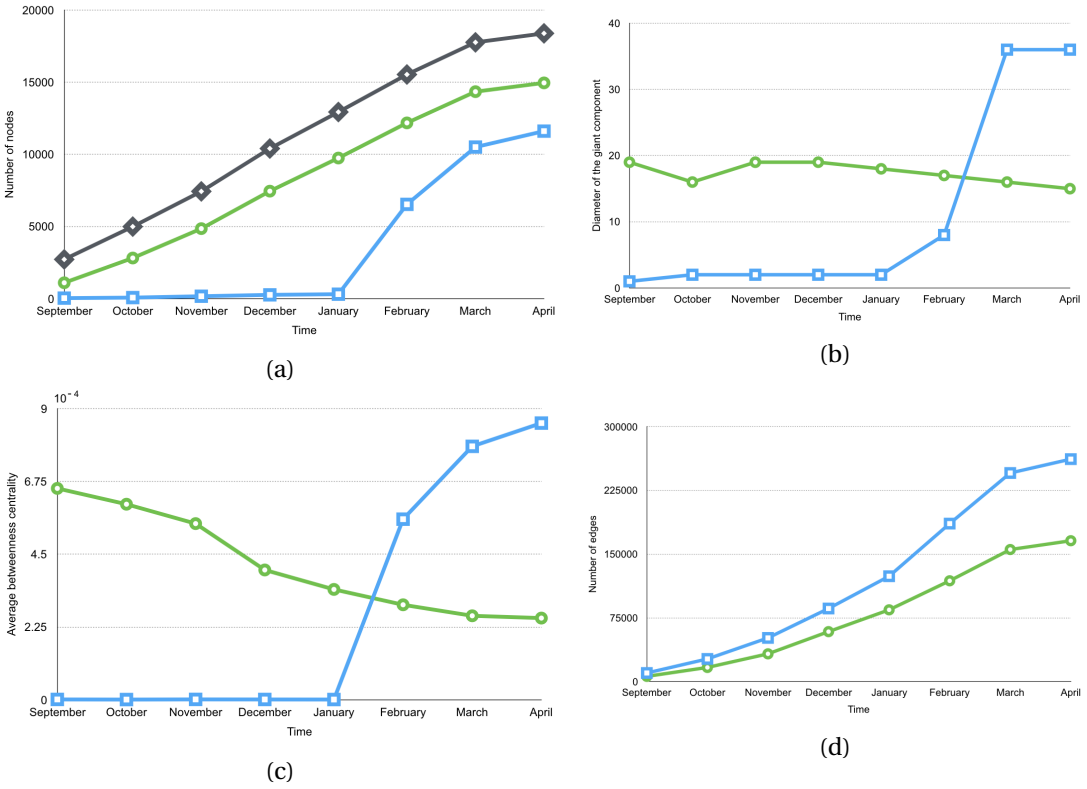


Figure 5.6 – **Evolution of Wiki-memory and random-memory at the hour scale through time.** In blue, the Wiki-memory collectively created by users. In green, the random memory. **a,** evolution of the number of nodes of the giant connected component. The grey curve shows the total number of nodes in both memories. **b,** evolution of the diameter of the giant component. **c,** evolution of the average node betweenness centrality. **d.** growth of the number of edges.

component of the Wiki-memory is larger than in the random-memory.

From what we have demonstrated so far, it seems that human curiosity is unconsciously and collectively driven towards the creation of a well-structured knowledge network of memories that optimizes the storage, organization, and retrieval of information.

Seeking to know the cause of this mechanism triggers another set of questions that remain to be solved. We suspect that our peculiar interest for social topics depends on our human nature with its set of qualities (compassion, altruism, emulation) and defaults (greed, jealousy, libido). Also, we hypothesize that our common interest in human relations is also driven by biological considerations as it may trigger a richer set of signals in the brain calling to both episodic and semantic memory [241, 242]. In layman's terms, we hypothesize that recalling a memory involving humans, like a birthday, for instance, makes it more vivid as it triggers both contextual and semantic signals. These data could be the date and time, the people involved, the location, as well as the sounds, colors, emotions, and taste related to the event. This last part definitively deserves to be investigated more, but it offers news perspectives in

the understanding of human nature.

5.4 Discussion

In this work, we have combined the activity of millions of visitors with the graph of Wikipedia to extract Wiki-souvenirs, small contextual subgraphs of Wikipedia that encode what was collectively browsed from world events. The analysis of these Wiki-souvenirs revealed that visitors were mostly interested in social topics and human relations as opposed to pure knowledge on random facts. In building the Wiki-memory, the graph of Wiki-souvenirs, we have shown that it was significantly different from existing network topologies such as random or small-world networks and could constitute a new class of network on its own. Moreover, we have uncovered that human curiosity is instinctively led towards the creation of a well-structured and well-connected network of memories. Surprisingly, this network possesses more edges than the random one but has a larger diameter and a larger average centrality. Human curiosity seems to be directed towards information that creates more connections between souvenirs. We can conjecture that it makes them easier to remember and understand. The social facts and topics seem to act as hubs in the memory network, relating different events and catalyzing connections. However, the network does not become a small-world as one could expect: it seems that a small-world network may not be the best structure to store and retrieve information.

On a more practical side, we could now imagine how to use Wiki-souvenirs to build a new kind of search engine emphasizing on the context and data exploration [243]. Indeed, search engines are the core systems behind the World Wide Web, and despite their technical improvements of the last 20 years regarding hardware or algorithms, the presentation of the results is still represented as a list for all types of queries. Focusing on the news or events, the core of Wiki-souvenirs, we observe that existing systems fail to capture the essence of the information as they simply point to online news site ranked by popularity. Instead, we propose to present a digest or overview glance of what is important, what are the relations between the different protagonists or how the event in being unfolded. While it is possible to filter the Google results between a date range, for instance, it is also tough to extract contextual information as we do not necessarily know the bounds of the event. Worse, the different aspects of the same event cannot be adequately captured, and the user needs to browse several results to have a less biased opinion of the event. All theses steps, necessary to go beyond the superficial understanding of a given event to take time and efforts to be answered with existing tools.

On the contrary, by indexing the content of WS, we propose to retrieve all the key aspects of an event collectively curated by the activity of visitors searching for a particular page. Additionally, we propose to navigate between similar events by following the links of the Wiki-memory that can be filtered by a given range to pinpoint precisely the period of interest. We present our vision for the interface of the contextual search engine in Fig. 5.7.

On the technical side, many challenges remain to be solved, notably with the construction

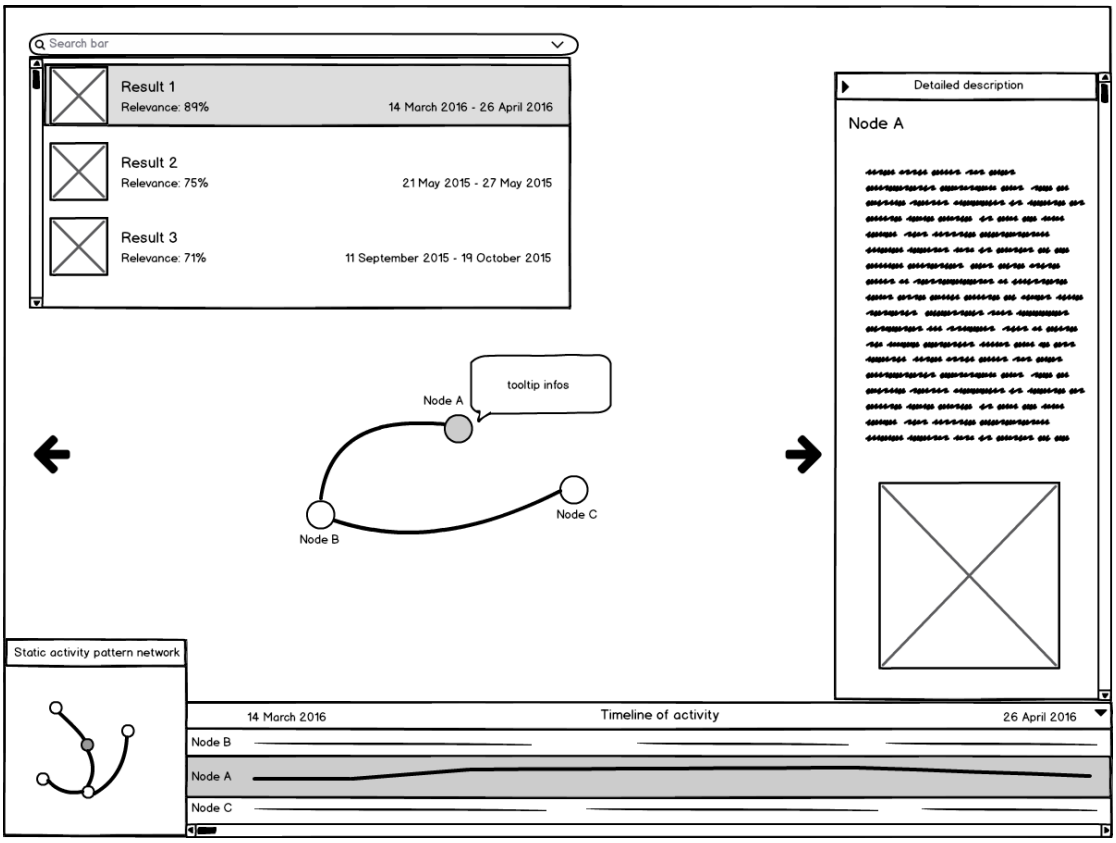


Figure 5.7 – **GUI wireframe of our contextual search engine.** On a given query, we present a list of matching Wiki-souvenirs in the form of a graph. Clicking on a node displays a foldable panel with the Wikipedia page (on the right). The usage of the big arrows simply allows iterating through results. On the bottom left, we display the node corresponding to the current Wiki-souvenir in the Wiki-memory as well as a short list of neighbors. A click on this minimap loads the corresponding Wiki-souvenir. Finally, on the bottom, we display the activity time-series of the nodes of the WS similarly to what is presented in Fig. 5.3, allowing to track the activity of each page for the whole duration of the event.

of the pipeline to ingest in real-time the visits on Wikipedia on all languages. On the design side, it is essential to building an informative interface that is user-friendly yet powerful for analytics. We hope that once live, our system will be used by a large sample of the Wikipedia visitors as the exploration of our collective memories concerns us all.

6 Discussion

Throughout this thesis, we have seen that networks are fundamental to model the world's complexity. By studying various applications in both academia and industry, we have demonstrated that the field of practice of network science combining data mining, computer science, machine learning, graph theory, signal processing and data visualization is virtually limitless. In a general manner, we have tried to argue for a more open and reproducible science along the chapters of this thesis with the creation and study of publicly available datasets like the Free Music Archive or Wikipedia for example. One of our core contribution in the area is the creation of a practical guide introducing tools and techniques to make network science more formally defined and reproducible. We hope that what we have presented will be refined and adapted to be still relevant for new-comers in the field as time goes by.

6.1 The causal multilayer graph of activity

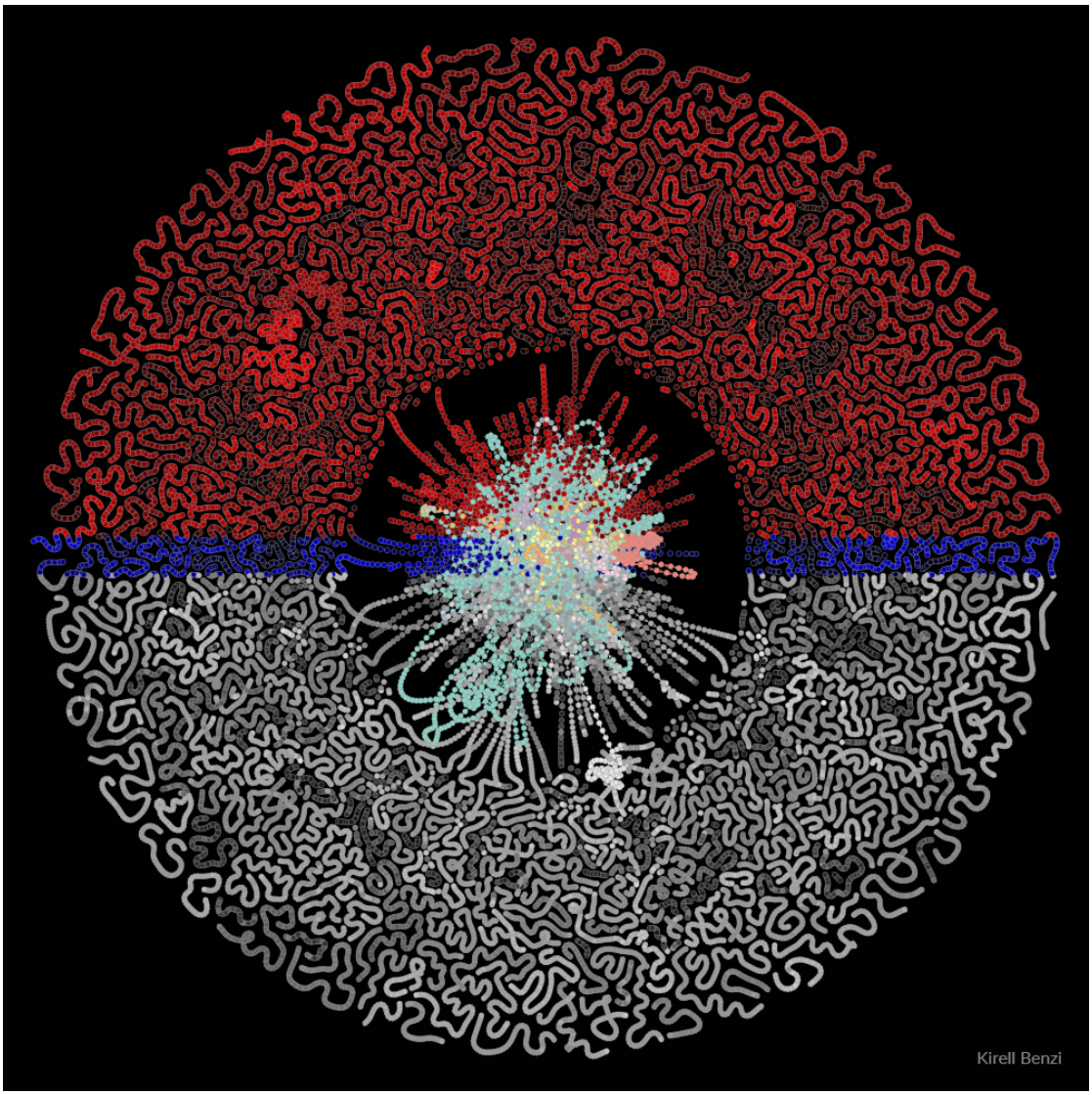
We have shown that the emergence of new graph methods like the causal multilayer graph of activity is the result of a data-driven, inter-disciplinary and practical work focusing on today's large-scale challenges. With applications in audio recommender systems, neuroscience, crowd tracking, rumor spreading and social behavior analysis we have only surfaced the powerful combination of graph and signals over its vertices (or edges). Our studies have revealed how to harness the dynamical aspects of information with evidence of the formation of spatio-temporal communities on Twitter or the visualization of pedestrian movements in a train station. We can also now highlight the dynamic activity of the brain in resting state and explore our collective memory of past events using the Wiki-memory. Future work on the algorithmic part would concern the threshold that defines what is considered active or not. While it is application dependent, we hope to rely on machine learning to learn how to tune the parameters optimally, given a set of constraints. For instance, in our brain example, we know that the hemodynamic response function signaling a brain activation is between 6 and 16 seconds. We could then learn what is the optimal threshold to obtain patterns of this size, instead of discarding patterns that do not fit these criteria.

Another limiting factor of the application of our method is related to the data availability as it often represents more than 50% of the total work. We underline the fact that finding a suitable source of data still remains the hardest part of the job as it requires both intuition and experience. Coming back to our arguments in favor of open research, we hope that more and more APIs will be accessible to researchers in the future, especially to fetch dynamic data. In the meantime, we already have a few starting projects related to the application of our causal multilayer graph of activity.

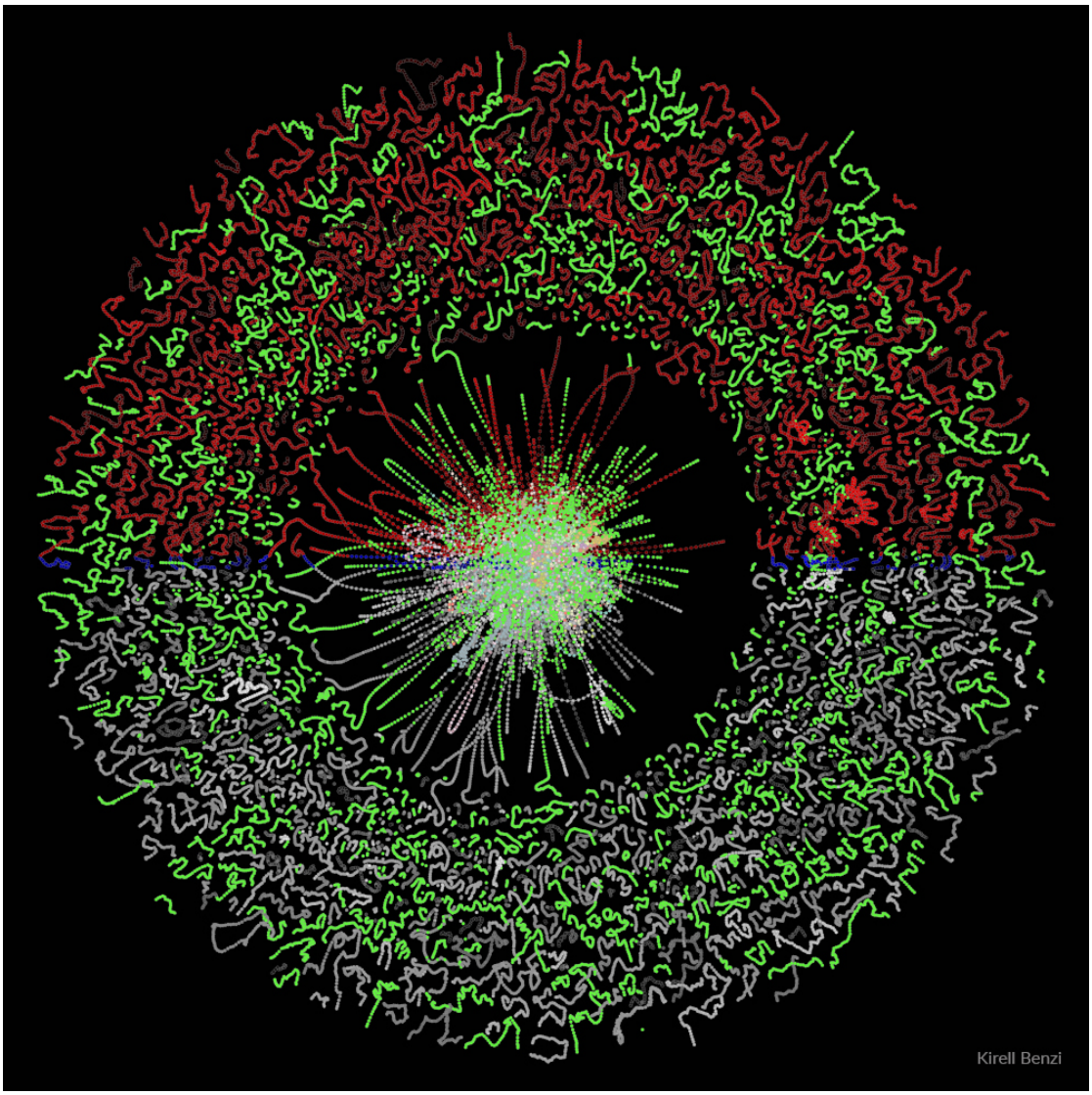
First, we plan to use the causal multilayer graph of activity on Youtube to shed some light on viral phenomena and the evolution of dynamic graphs in general. The idea behind this study is to try to understand how information spreads on a Youtube video network where each node is a video and edges in between are given if they follow each other in a playlist. In this regard we have collected time series on the evolution of views, likes and dislikes on all playlists related to the mobile game Pokemon Go during several weeks as the viral phenomenon took the world by storm in July 2016 with over 100 million downloads in a month. Our dataset, composed of millions of data points, promises to be fascinating to study as the evolution of the video network through time in Fig. 6.1 reveals. In the figure, we show a snapshot of the video graph and its evolution fifteen days later. The last figure highlights the nodes that have been added to the network. We clearly see that we have a well-connected core and many disconnected tendrils around it. More than 30% of the new videos are connected to the core. We also notice an exponential increase in the total number of views for Pokemon videos in the core. At the beginning of September, the total views reached 7.6 billion with 15% of views concentrated in only the first 100 most popular videos. This particular distribution of views and the topology of the graph raise interesting questions about virality. What makes a video viral, is it only its content or also its connectivity in the network? Looking at the graph of creators, how does the number of followers influences the diffusion of the video? We hope to investigate these aspects in the near future.

Using the same framework, we hope to tackle a new research area by working with linguists and sociologists to create a meme tracker, following the previous work of Leskovec in [244]. The challenges to be solved to develop a meme tracker are numerous as to understand how memes are created and transformed by users, one first needs to reveal how the information spreads among them. Here, our Twitter analysis in chapter 4 seems like a good starting point. We would first need to apply it for another given corpus of documents and visualize the evolution of interactions of users from a given set of hashtags. Building on these patterns of user activity on various social networks, we could move them on to the textual analysis of the content generated by each user. In the case a meme is just repeated without modification, it is trivial to track them in the pattern as we only need to follow the edges of the dynamic activation component. However, when the text is modified by users, for example by quoting a part of the original content, more sophisticated methods could be used such as the word2vec and paragraph2vec described in [245, 246, 247]. In these methods, a deep neural network is trained on a large corpus of documents to learn syntactic and semantic relationships between words or sentences. Because words are now represented as a set of features, our meme tracker

can now draw on robust machine learning techniques to perform similarity comparison between user generated content and eventually classify memes together. The classification could be done using unsupervised techniques such as k-means or DBscan [248] or resorting to semi-supervised methods such as SVM [249] and deep neural networks using the knowledge of linguists to learn meaningful classes.



(a)



(b)

Figure 6.1 – **Evolution of the Pokemon video graph. Copyright Kirell Benzi.** Each node represents a Youtube video with the word Pokemon in its title. They are connected if they follow each other in Youtube playlists also containing the world Pokemon. The red, white and blue color mimics the Pokeball, virtual item used to catch Pokemons. The core of the network is colored by communities. The shades of the different colors depend on the number of views of the weakly connected subgraph. **a,** Pokegraph on the 17th August 2016. **b,** Pokegraph on the 5th September 2016, nodes added between the two snapshots are colored in green.

6.2 Building real-world applications

This thesis has also revealed the difficulty of switching from an academic (yet practical) project to a real-world distributed implementation as we have done with Genezik. As we have shown, these implementation aspects are nonetheless vital to delivering robust data science products

in the industry. In our opinion, software engineering is, for now, not considered as a first-class citizen in research. As a result, it leads to poorly written code that limits code reuse and the validity of models [250]. In life science, Fang *et al.* reported that 21.2% of retracted scientific publications were attributed to errors [251]. While we don't have hard data for our field, we can use this as a warning or reminder to further develop the testability of models.

Recalling the peculiar hybrid architecture of Genezik, we have shown that the usage of networks is perfectly adapted to model user taste and its evolution on a personal collection of audio files. We have demonstrated that our method could improve the state-of-the-art in the performance of our global recommender system as well as the usability of the final application. We have proposed novel playlist recommendation strategies based on traversals of a graph of musical similarity such as "Take Me On A Tour" which tries to fill the gaps between 3 or 4 songs in an optimal manner. We also showed how we could renew playlists while keeping the original intent of its creator with *play-points*.

A fascinating and logical evolution of Genezik would be to apply the same idea for pictures. Contrary to music where the songs are well-defined with metadata, personal pictures only contain geo-localization and camera technical specifications at best. Photos organization is exceedingly difficult with the current tools as they only propose to use face detection, date, and geo-localization to sort or filter them (see Apple Photos, Google Picasa or Facebook). Leveraging the recent advances in image captioning using deep neural networks [252], a graph of image keywords could be employed instead to explore and structure the user's personal library. User requests similar to: "generate a slideshow of all the pictures of me at the beach" would result on a customized graph traversal similarly to what we propose in Genezik. We feel that the integration of such technologies in a mobile app would disrupt the market significantly and overcome the challenges of their integration.

Finally, we hope to transfer some of the experience we have gained in building distributed recommender systems to the realization of our contextual search engine based on Wikisouvenirs. Here, our plan of action is to start to build a proof-of-concept regarding the web interface allowing visitors to explore the data we have already extracted. We then plan to tackle the robust implementation of the causal multilayer graph in an online fashion outputting dynamic activation components on the fly using the streaming connected components for example [253].

6.3 Diffusing science to the general public

Raising scientific awareness in the mind of taxpayers is a serious concern for those who finance scientific projects [254]. Perhaps the most long-term contribution induced by the writing of this thesis relates to the diffusion of science to the general public via network visualizations.

Indeed, we think that science should be approachable by anyone, especially those who are not naturally inclined to study it. Showing that science can also be "cool" seems like a great

way to motivate youngsters [255, 256]. It could also maybe help changing women's perception of having a career in STEM [257, 258].

Our approach mixing art and science tries to show both the complexity and inner beauty of physical phenomena based on the datasets presented in this thesis as we show in Fig. 6.2. Currently displayed in an exhibition called "Singular Networks" [12] in Lausanne, Switzerland, we are actively seeking to expand its visibility to other countries. We hope that our passion of networks will invite more people to dive into this fascinating universe where each node, beyond calling for imagination, tells a story.

Figure 6.2 – **Secret Knowledge - 2016. Copyright Kirell Benzi.** Who would have thought that Wikipedia could be so structured? Each node represents a group of pages visited by large numbers of people on a specific subject related to world news or events. Nodes are connected if they share one or more common pages. Petals are formed by grouping together all nodes from a given topic and stacking them on top of each other. Displayed in the shape of a ring, the different topics are well connected because humans tend to be pretty curious and visit many different pages! Do you think you can spot the petal that contains the hottest news about your favorite TV shows?

Bibliography

[1] A.-L. Barabasi, "Linked: How everything is connected to everything else and what it means," *Plume Editors*, 2002.

[2] T. G. Lewis, *Network science: Theory and applications*. John Wiley & Sons, 2011.

[3] I. Data and S. Division, "Ict facts and figures." https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf, 2015.

[4] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future*, vol. 2007, pp. 1–16, 2012.

[5] M. Loukides, *What is data science?* " O'Reilly Media, Inc.", 2011.

[6] R. Schutt and C. O'Neil, *Doing data science: Straight talk from the frontline*. " O'Reilly Media, Inc.", 2013.

[7] B. Baesens, *Analytics in a big data world: The essential guide to data science and its applications*. John Wiley & Sons, 2014.

[8] K. Benzi, B. Ricaud, and P. Vandergheynst, "Principal patterns on graphs: Discovering coherent structures in datasets," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 160–173, 2016.

[9] A. Alahi, P. Vandergheynst, and K. Benzi, "System and method for media library navigation and recommendation," 2014. EPFL Patent WO 2014/002064A1.

[10] K. Benzi, V. Kalofolias, X. Bresson, and P. Vandergheynst, "Song recommendation with non-negative matrix factorization and graph total variation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2439–2443, IEEE, 2016.

[11] A. Griffa, K. Benzi, B. Ricaud, P. Vandergheynst, J.-P. Thiran, and P. Hagmann, "Mapping resting-state dynamics on causal multilayer graphs: a combined functional and diffusion MRI approach," *NeuroImage. In review.*, 2016.

[12] K. Benzi, "Singular Networks." http://kirellbenzi.com/art/, 2016.

[13] V. Stodden, F. Leisch, and R. D. Peng, *Implementing reproducible research*. CRC Press, 2014.

[14] K. Lynch, "Star wars day: The force is strong in these world records." http://www.guinnessworldrecords.com/news/2015/5/star-wars-day-the-force-is-strong-in-these-world-records-378041, 2015.

[15] D. K. Elson, N. Dames, and K. R. McKeown, "Extracting social networks from literary fiction," in *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 138–147, Association for Computational Linguistics, 2010.

[16] J. Rydberg-Cox, "Social networks and the language of greek tragedy," in *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, vol. 1, 2011.

[17] A. Agarwal, A. Corvalan, J. Jensen, and O. Rambow, "Social network analysis of alice in wonderland," in *Workshop on Computational Linguistics for Literature*, pp. 88–96, 2012.

[18] E. Vanhoutte, *Defining digital humanities: a reader*. Ashgate Publishing, Ltd., 2013.

[19] P. Cohen, "Digital keys for unlocking the humanities' riches," *New York Times*, vol. 16, 2010.

[20] M. Kirschenbaum, "What is digital humanities and what's it doing in english departments?," *Debates in the digital humanities*, vol. 3, 2012.

[21] S. Schreibman, R. Siemens, and J. Unsworth, *A companion to digital humanities*. John Wiley & Sons, 2008.

[22] D. Berry, *Understanding digital humanities*. Springer, 2012.

[23] C. Warwick, C. Warwick, M. Terras, and J. Nyhan, *Digital humanities in practice*. Facet Publishing, 2012.

[24] M. K. Gold, *Debates in the digital humanities*. U of Minnesota Press, 2012.

[25] F. Albertin, A. Astolfo, M. Stampanoni, E. Peccenini, Y. Hwu, F. Kaplan, and G. Margaritondo, "X-ray spectrometry and imaging for ancient administrative handwritten documents," *X-Ray Spectrometry*, vol. 44, no. 3, pp. 93–98, 2015. XRS-14-0064.R1.

[26] B. Wagner, E. Bulska, A. Hulanicki, M. Heck, and H. Ortner, "Topochemical investigation of ancient manuscripts," *Fresenius' journal of analytical chemistry*, vol. 369, no. 7-8, pp. 674–679, 2001.

[27] G. Joutel, V. Eglin, S. Bres, and H. Emptoz, "Curvelets based feature extraction of handwritten shapes for ancient manuscripts classification," in *Electronic Imaging 2007*, pp. 65000D–65000D, International Society for Optics and Photonics, 2007.

[28] Y. Leydier, A. Ouji, F. LeBourgeois, and H. Emptoz, "Towards an omnilingual word retrieval system for ancient manuscripts," *Pattern Recognition*, vol. 42, no. 9, pp. 2089–2105, 2009.

[29] M.-F. Moens, J. Li, and T.-S. Chua, *Mining user generated content*. CRC Press, 2014.

[30] G. Van Rossum *et al.*, "Python programming language.," in *USENIX Annual Technical Conference*, vol. 41, 2007.

[31] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

[32] B. W. Kernighan and D. M. Ritchie, "The c programming language," 2006.

[33] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[34] E. Jones, T. Oliphant, P. Peterson, *et al.*, "Open source scientific tools for python," 2001.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[36] L. Pappano, "The year of the mooc," *The New York Times*, vol. 2, no. 12, p. 2012, 2012.

[37] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: Abstraction and reuse of object-oriented design," in *European Conference on Object-Oriented Programming*, pp. 406–431, Springer, 1993.

[38] S. McConnell, *Code complete*. Pearson Education, 2004.

[39] B. Lonsdorf, "Mostly adequate guide to functional programming." https://drboolean.gitbooks.io/mostly-adequate-guide/content/, 2016.

[40] M. Ragan-Kelley, F. Perez, B. Granger, T. Kluyver, P. Ivanov, J. Frederic, and M. Bussonier, "The jupyter/ipython architecture: a unified view of computational research, from interactive exploration to communication and publication.," in *AGU Fall Meeting Abstracts*, vol. 1, p. 07, 2014.

[41] J. Loeliger and M. McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc.", 2012.

[42] M. Masse, *REST API design rulebook*. " O'Reilly Media, Inc.", 2011.

[43] D. Crockford, "The application/json media type for javascript object notation (json)," 2006.

[44] D. Kouzis-Loukas, *Learning Scrapy*. Packt Publishing Ltd, 2016.

[45] B. Bollobás, *Modern graph theory*, vol. 184. Springer Science & Business Media, 2013.

[46] D. Borthakur, "Hdfs architecture guide," *HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf*, p. 39, 2008.

[47] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An overview of the hdf5 technology suite and its applications," in *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, pp. 36–47, ACM, 2011.

[48] R. P. Padhy, M. R. Patra, and S. C. Satapathy, "Rdbms to nosql: reviewing some next-generation non-relational database's," *International Journal of Advanced Engineering Science and Technologies*, vol. 11, no. 1, pp. 15–30, 2011.

[49] K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. Capretz, "Data management in cloud environments: Nosql and newsql data stores," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, p. 1, 2013.

[50] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[51] P. J. Sadalage and M. Fowler, *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education, 2012.

[52] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica, "Graphx: A resilient distributed graph system on spark," in *First International Workshop on Graph Data Management Experiences and Systems*, p. 2, ACM, 2013.

[53] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein, "Graphlab: A new framework for parallel machine learning," *arXiv preprint arXiv:1408.2041*, 2014.

[54] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014.

[55] J. G. Siek, L.-Q. Lee, and A. Lumsdaine, *Boost Graph Library: User Guide and Reference Manual, The*. Pearson Education, 2001.

[56] D. A. Schult and P. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, vol. 2008, pp. 11–16, 2008.

[57] L. Boytsov and B. Naidan, "Engineering efficient and effective non-metric space library," in *Similarity Search and Applications - 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings*, pp. 280–293, 2013.

[58] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *arXiv preprint arXiv:1408.2927*, 2014.

[59] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," tech. rep., Citeseer, 2002.

[60] M. Bastian, S. Heymann, M. Jacomy, *et al.*, "Gephi: an open source software for exploring and manipulating networks.," *ICWSM*, vol. 8, pp. 361–362, 2009.

[61] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software," *PloS one*, vol. 9, no. 6, p. e98679, 2014.

[62] S. Martin, W. M. Brown, R. Klavans, and K. W. Boyack, "Openord: an open-source toolbox for large graph layout," in *IS&T/SPIE Electronic Imaging*, pp. 786806–786806, International Society for Optics and Photonics, 2011.

[63] Y. Frishman and A. Tal, "Multi-level graph layout on the gpu," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1310–1319, 2007.

[64] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[65] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proceedings of the 19th international conference on World wide web*, pp. 631–640, ACM, 2010.

[66] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web.," 1999.

[67] R. Wang, Y. Perez-Riverol, H. Hermjakob, and J. A. Vizcaíno, "Open source libraries and frameworks for biological data visualisation: A guide for developers," *Proteomics*, vol. 15, no. 8, pp. 1356–1374, 2015.

[68] J. Leskovec and A. Krevl, "{SNAP Datasets}:{Stanford} large network dataset collection," 2015.

[69] R. A. Rossi and N. K. Ahmed, "An interactive data repository with visual analytics," *ACM SIGKDD Explorations Newsletter*, vol. 17, no. 2, pp. 37–41, 2016.

[70] T. Li, M. Ogihara, and Q. Li, "A comparative study on content-based music genre classification," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 282–289, ACM, 2003.

[71] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content: a survey," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 133–141, 2006.

[72] L. Lu, D. Liu, and H.-J. Zhang, "Automatic mood detection and tracking of music audio signals," *IEEE Transactions on audio, speech, and language processing*, vol. 14, no. 1, pp. 5–18, 2006.

[73] Y.-H. Yang, C.-C. Liu, and H. H. Chen, "Music emotion classification: a fuzzy approach," in *Proceedings of the 14th ACM international conference on Multimedia*, pp. 81–84, ACM, 2006.

[74] E. Gómez, A. Klapuri, and B. Meudic, "Melody description and extraction in the context of music content processing," *Journal of New Music Research*, vol. 32, no. 1, pp. 23–40, 2003.

[75] A. Krishna and T. V. Sreenivas, "Music instrument recognition: from isolated notes to solo phrases," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 4, pp. iv–265, IEEE, 2004.

[76] C. Cao, M. Li, J. Liu, and Y. Yan, "Singing melody extraction in polyphonic music by harmonic tracking.," in *ISMIR*, pp. 373–374, 2007.

[77] V. Rao and P. Rao, "Vocal melody extraction in the presence of pitched accompaniment in polyphonic music," *IEEE transactions on audio, speech, and language processing*, vol. 18, no. 8, pp. 2145–2154, 2010.

[78] A. Klapuri and M. Davy, *Signal processing methods for music transcription*. Springer Science & Business Media, 2007.

[79] G. Nierhaus, *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media, 2009.

[80] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.

[81] E. R. Miranda and J. Al Biles, *Evolutionary computer music*. Springer, 2007.

[82] D. Schwarz, "Concatenative sound synthesis: The early years," *Journal of New Music Research*, vol. 35, no. 1, pp. 3–22, 2006.

[83] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[84] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[85] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[86] R. Sinha and K. Swearingen, "The role of transparency in recommender systems," in *CHI'02 extended abstracts on Human factors in computing systems*, pp. 830–831, ACM, 2002.

[87] L. Barrington, R. Oda, and G. R. Lanckriet, "Smarter than genius? human evaluation of music recommender systems.," in *ISMIR*, vol. 9, pp. 357–362, Citeseer, 2009.

[88] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[89] P. Hartono and R. Yoshitake, "Automatic Playlist Generation from Self-Organizing Music Map," *Journal of Signal Processing*, vol. 17(1), pp. 11–19, 2013.

[90] K. Budhraja, A. Singh, G. Dubey, and A. Khosla, "Probability based Playlist Generation based on Music Similarity and User Customization," *Proceedings of National Conference on Computing and Communication Systems*, pp. 1–5, 2012.

[91] S. Chen, J. Moore, D. Turnbull, and T. Joachims, "Playlist Prediction via Metric Embedding," *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 714–722, 2012.

[92] J. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, 1998.

[93] Z. Chedrawy and S. Abidi, "A Web Recommender System for Recommending, Predicting and Personalizing Music Playlists," in *Proceedings of Web Information Systems Engineering*, pp. 335–342, 2009.

[94] N. Hariri, B. Mobasher, and R. Burke, "Context-Aware Music Recommendation based on Latent Topic Sequential Patterns," in *Proceedings of ACM conference on Recommender systems*, pp. 131–138, 2012.

[95] N. Liu, S. Lai, C. Chen, and S. Hsieh, "Adaptive Music Recommendation based on User Behavior in Time Slot," *International Journal of Computer Science and Network Security*, vol. 9(2), pp. 219–227, 2009.

[96] O. Celma, "Music recommendation," in *Music Recommendation and Discovery*, pp. 43–85, Springer Berlin Heidelberg, 2010.

[97] G. Bonnin and D. Jannach, "Automated Generation of Music Playlists: Survey and Experiments," *ACM Computing Surveys (CSUR)*, vol. 47, pp. 1–35, nov 2014.

[98] M. Schedl, P. Knees, B. McFee, D. Bogdanov, and M. Kaminskas, "Music recommender systems," in *Recommender Systems Handbook*, pp. 453–492, Springer, 2015.

[99] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 8, pp. 30–37, 2009.

[100] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *In IEEE International Conference on Data Mining (ICDM 2008*, pp. 263–272, 2008.

[101] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, pp. 452–461, 2009.

[102] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 287–296, ACM, 2011.

[103] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix Completion on Graphs," *arXiv*, vol. preprint arXiv:1408.1717, 2014.

[104] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 8, pp. 1548–1560, 2011.

[105] P. Combettes and J. Pesquet, "Proximal Splitting Methods in Signal Processing," *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212, 2011.

[106] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear Total Variation Based Noise Removal Algorithms," *Physica D*, vol. 60(1-4), pp. 259 – 268, 1992.

[107] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[108] S. A. Vavasis, "On the complexity of nonnegative matrix factorization," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1364–1377, 2009.

[109] F. Yanez and F. Bach, "Primal-Dual Algorithms for Non-negative Matrix Factorization with the Kullback-Leibler Divergence," *arXiv:1412.1788*, 2014.

[110] X. Bresson, T. Laurent, D. Uminsky, and J. von Brecht, "Multiclass Total Variation Clustering," *Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1421–1429, 2013.

[111] A. Chambolle and T. Pock, "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging," *Journal of Mathematical Imaging and Vision*, vol. 40(1), pp. 120–145, 2011.

[112] D. Donoho, "De-Noising by Soft-Thresholding," *IEEE Transactions on Information Theory*, vol. 41(33), pp. 613–627, 1995.

[113] B. McFee and G. R. Lanckriet, "Hypergraph models of playlist dialects.," in *ISMIR*, pp. 343–348, Citeseer, 2012.

[114] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset.," in *ISMIR*, vol. 2, p. 10, 2011.

[115] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[116] A. Schindler and A. Rauber, "Capturing the temporal domain in echonest features for improved classification effectiveness," in *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation*, pp. 214–227, Springer, 2014.

[117] B. McFee and G. R. Lanckriet, "The Natural Language of Playlists.," in *ISMIR*, pp. 537–542, 2011.

[118] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, pp. 707–710, 1966.

[119] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, pp. 1473–1480, 2005.

[120] C. Boutsidis and E. Gallopoulos, "SVD Based Initialization: A Head Start For Nonnegative Matrix Factorization," *Pattern Recognition*, vol. 41, no. 4, pp. 1350–1362, 2008.

[121] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A review of audio fingerprinting," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 41, no. 3, pp. 271–284, 2005.

[122] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Transactions on Multimedia*, vol. 7, pp. 96–104, Feb. 2005.

[123] D. Jang, C. D. Yoo, S. Lee, S. Kim, and T. Kalker, "Pairwise Boosted Audio Fingerprint," *IEEE Transactions on Information Forensics and Security*, vol. 4, pp. 995–1004, Dec. 2009.

[124] J. J. McCarthy, "An introduction to harmonic serialism," *Language and Linguistics Compass*, vol. 4, no. 10, pp. 1001–1018, 2010.

[125] M. Müller, *Information retrieval for music and motion*, vol. 2. Springer, 2007.

[126] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, "Yaafe, an easy to use and efficient audio feature extraction software.," in *ISMIR*, pp. 441–446, 2010.

[127] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, 2015.

[128] T. Liu, A. W. Moore, K. Yang, and A. G. Gray, "An investigation of practical approximate nearest neighbor algorithms," in *Advances in neural information processing systems*, pp. 825–832, 2004.

[129] M. Muja and D. G. Lowe, "Flann, fast library for approximate nearest neighbors," in *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, INSTICC Press, 2009.

[130] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification," in *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pp. 9–15, ACM, 2011.

[131] F. Spitzer, *Principles of random walk*, vol. 34. Springer Science & Business Media, 2013.

[132] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[133] M. Holzer, F. Schulz, and D. Wagner, "Engineering multilevel overlay graphs for shortest-path queries," *Journal of Experimental Algorithmics (JEA)*, vol. 13, p. 5, 2009.

[134] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[135] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[136] B. L. Sturm, "An analysis of the gtzan music genre dataset," in *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, pp. 7–12, ACM, 2012.

[137] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging.," in *ISMIR*, pp. 387–392, 2009.

[138] M. Slaney, K. Weinberger, and W. White, "Learning a metric for music similarity," in *International Symposium on Music Information Retrieval (ISMIR)*, 2008.

[139] K. Benzi, M. Defferrard, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2016.

[140] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[141] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[142] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*, pp. 217–253, Springer, 2011.

[143] J. M. Beggs and D. Plenz, "Neuronal avalanches in neocortical circuits," *The Journal of neuroscience*, vol. 23, no. 35, pp. 11167–11177, 2003.

[144] O. Sporns, "Contributions and challenges for network models in cognitive neuroscience," *Nature neuroscience*, vol. 17, no. 5, pp. 652–660, 2014.

[145] M. J. Keeling and K. T. Eames, "Networks and epidemic models," *Journal of the Royal Society Interface*, vol. 2, no. 4, pp. 295–307, 2005.

[146] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani, "The role of the airline transportation network in the prediction and predictability of global epidemics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 7, pp. 2015–2020, 2006.

[147] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi, "The anatomy of a scientific rumor," *Scientific reports*, vol. 3, 2013.

[148] L. Kovanen, M. Karsai, K. Kaski, J. Kertész, and J. Saramäki, "Temporal motifs in time-dependent networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2011, no. 11, p. P11005, 2011.

[149] U. Redmond and P. Cunningham, "Temporal subgraph isomorphism," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, (New York, NY, USA), pp. 1451–1452, ACM, 2013.

[150] P. Holme, "Modern temporal network theory: a colloquium," *The European Physical Journal B*, vol. 88, no. 9, pp. 1–30, 2015.

[151] M. Salehi, R. Sharma, M. Marzolla, M. Magnani, P. Siyari, and D. Montesi, "Spreading processes in multilayer networks," *Network Science and Engineering, IEEE Transactions on*, vol. 2, pp. 65–83, April 2015.

[152] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.

[153] K. Benzi, "Code repository for the causal multilayer graph." https://github.com/epfl-lts2/sptgraph, 2015.

[154] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *Science*, vol. 328, no. 5980, pp. 876–878, 2010.

[155] S. Gómez, A. Diaz-Guilera, J. Gomez-Gardeñes, C. J. Perez-Vicente, Y. Moreno, and A. Arenas, "Diffusion dynamics on multiplex networks," *Physical review letters*, vol. 110, no. 2, p. 028701, 2013.

[156] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *Journal of Complex Networks*, vol. 2, pp. 203–271, jul 2014.

[157] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, vol. 519, pp. 97–125, oct 2012.

[158] E. Valdano, L. Ferreri, C. Poletto, and V. Colizza, "Analytical computation of the epidemic threshold on temporal networks," *Physical Review X*, vol. 5, no. 2, p. 021005, 2015.

[159] M. De Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, and A. Arenas, "Mathematical formulation of multilayer networks," *Physical Review X*, vol. 3, no. 4, p. 041022, 2013.

[160] G. Sabidussi, "Graph multiplication," *Mathematische Zeitschrift*, vol. 72, no. 1, pp. 446–457, 1959.

[161] U. Kang, C. E. Tsourakakis, and C. Faloutsos, "PEGASUS: A Peta-Scale Graph Mining System Implementation and Observations," in *2009 Ninth IEEE International Conference on Data Mining*, IEEE, dec 2009.

[162] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, mar 1982.

[163] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243–256, 2013.

[164] H. Bischof, A. Leonardis, and A. Selb, "Mdl principle for robust vector quantisation," *Pattern Analysis & Applications*, vol. 2, no. 1, pp. 59–72, 1999.

[165] D. Pelleg, A. W. Moore, *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters.," in *ICML*, pp. 727–734, 2000.

[166] C. A. Sugar and G. M. James, "Finding the number of clusters in a dataset," *Journal of the American Statistical Association*, vol. 98, no. 463, 2003.

[167] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.

[168] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[169] R. Lletı, M. C. Ortiz, L. A. Sarabia, and M. S. Sánchez, "Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the silhouettes," *Analytica Chimica Acta*, vol. 515, no. 1, pp. 87–100, 2004.

[170] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-Aware Large-Scale Crowd Forecasting," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2014.

[171] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2274–2282, nov 2012.

[172] G. Bonnin and D. Jannach, "Automated Generation of Music Playlists: Survey and Experiments," *ACM Computing Surveys (CSUR)*, vol. 47, pp. 1–35, nov 2014.

[173] B. McFee and G. R. Lanckriet, "The Natural Language of Playlists.," in *ISMIR*, pp. 537–542, 2011.

[174] F. X. Castellanos and et al., "Clinical applications of the functional connectome," *NeuroIm.*, vol. 80, no. 0, pp. 527–540, 2013. Mapping the Connectome.

[175] A. Griffa and et al., "Structural connectomics in brain diseases," *NeuroIm.*, vol. 80, no. 0, pp. 515–526, 2013. Mapping the Connectome.

[176] B. Thomas Yeo and et al., "The organization of the human cerebral cortex estimated by intrinsic functional connectivity," *J. Neurophys.*, vol. 106, pp. 1125–1165, Sep 2011.

[177] D. M. Cole, S. M. Smith, and C. F. Beckmann, "Advances and pitfalls in the analysis and interpretation of resting-state FMRI data," *Front. Syst. Neurosci.*, 2010.

[178] J. Goni and et al., "Resting-brain functional connectivity predicted by analytic measures of network communication," *PNAS*, vol. 111, no. 2, pp. 833–838, 2014.

[179] M. D. Greicius and et al., "Resting-state functional connectivity reflects structural connectivity in the default mode network," *Cer. Cortex*, vol. 19, no. 1, pp. 72–78, 2009.

[180] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nature Reviews Neuroscience*, vol. 10, no. 3, pp. 186–198, 2009.

[181] S. M. Smith, D. Vidaurre, C. F. Beckmann, M. F. Glasser, M. Jenkinson, K. L. Miller, T. E. Nichols, E. C. Robinson, G. Salimi-Khorshidi, M. W. Woolrich, *et al.*, "Functional connectomics from resting-state fmri," *Trends in cognitive sciences*, vol. 17, no. 12, pp. 666–682, 2013.

[182] A. Griffa, "The topology of structural brain connectivity in diseases and spatio-temporal connectomics," 2015.

[183] P. Hagmann, M. Kurant, X. Gigandet, P. Thiran, V. J. Wedeen, R. Meuli, and J.-P. Thiran, "Mapping Human Whole-Brain Structural Networks with Diffusion MRI," *PLoS ONE*, vol. 2, p. e597, Jul 2007.

[184] R. S. Desikan and et al., "An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest," *NeuroIm.*, vol. 31, no. 3, pp. 968–980, 2006.

[185] V. Wedeen and et al., "Diffusion spectrum magnetic resonance imaging (DSI) tractography of crossing fibers," *NeuroIm.*, vol. 41, no. 4, pp. 1267–1277, 2008.

[186] M. P. van den Heuvel and O. Sporns, "Rich-Club Organization of the Human Connectome," *J. Neurosci.*, vol. 31, no. 44, pp. 15775–15786, 2011.

[187] J. Richiardi, S. Achard, H. Bunke, and D. Van De Ville, "Machine Learning with Brain Graphs: Predictive Modeling Approaches for Functional Imaging in Systems Neuroscience," *IEEE Sig. Proc. Mag.*, vol. 30, no. 3, pp. 58–70, 2013.

[188] J. D. Power and et al., "Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion," *NeuroIm.*, vol. 59, no. 3, pp. 2142–2154, 2012.

[189] K. Murphy, R. M. Birn, and P. A. Bandettini, "Resting-state fMRI confounds and cleanup," *NeuroIm.*, vol. 80, no. 0, pp. 349–359, 2013. Mapping the Connectome.

[190] A. Weissenbacher and et al., "Correlations and anticorrelations in resting-state functional connectivity MRI: A quantitative comparison of preprocessing strategies," *NeuroIm.*, vol. 47, no. 4, pp. 1408–1416, 2009.

[191] A. Abraham, E. Dohmatob, B. Thirion, D. Samaras, and G. Varoquaux, "Extracting Brain Regions from Rest fMRI with Total-Variation Constrained Dictionary Learning," *Lect. Notes in Comp. Sci.*, pp. 607–615, 2013.

[192] V. Michel and et al., "Total Variation Regularization for fMRI-Based Prediction of Behavior," *IEEE Trans. Med. Imaging*, vol. 30, no. 7, p. 1328–1340, 2011.

[193] S. Strother, "Evaluating fMRI preprocessing pipelines," *IEEE Eng. Med. Bio. Mag.*, vol. 25, no. 2, pp. 27–41, 2006.

[194] M. H. Lee and et al., "Clustering of Resting State Networks," *PLoS ONE*, vol. 7, p. e40370, Jul 2012.

[195] S. M. Smith and et al., "Temporally-independent functional modes of spontaneous brain activity," *PNAS*, vol. 109, no. 8, pp. 3131–3136, 2012.

[196] E. A. Allen and et al., "Tracking Whole-Brain Connectivity Dynamics in the Resting State," *Cer. Cortex*, vol. 24, pp. 663–676, Mar 2014.

[197] C. F. Beckmann and et al., "Investigations into resting-state connectivity using independent component analysis," *Phil. Trans. Royal Soc. B*, vol. 360, pp. 1001–1013, May 2005.

[198] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection." http://snap.stanford.edu/data, June 2014.

[199] A.-L. Barabasi, "The origin of bursts and heavy tails in human dynamics," *Nature*, vol. 435, no. 7039, pp. 207–211, 2005.

[200] M. Karsai, K. Kaski, A.-L. Barabási, and J. Kertész, "Universal features of correlated bursty behaviour," *Scientific reports*, vol. 2, 2012.

[201] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech.*, vol. 2008, p. P10008, oct 2008.

[202] M. D. MacLaren, "The Art of Computer Programming. Volume 2: Seminumerical Algorithms (Donald E. Knuth)," *SIAM Rev.*, vol. 12, pp. 306–308, apr 1970.

[203] S. E. Anderson, "Bit twiddling hacks." http://graphics. stanford. edu/s̃eander/bithacks. html, 2005.

[204] D. Lazer, R. Kennedy, G. King, and A. Vespignani, "The parable of google flu: traps in big data analysis," *Science*, vol. 343, no. 6176, pp. 1203–1205, 2014.

[205] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," *Rev. Mod. Phys.*, vol. 87, pp. 925–979, Aug 2015.

[206] A. Guille, H. Hacid, C. Favre, and D. A. Zighed, "Information diffusion in online social networks: A survey," *ACM SIGMOD Record*, vol. 42, no. 2, pp. 17–28, 2013.

[207] D. M. Romero, B. Meeder, and J. Kleinberg, "Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter," in *Proceedings of the 20th international conference on World wide web*, pp. 695–704, ACM, 2011.

[208] B. Min, K.-I. Goh, and A. Vazquez, "Spreading dynamics following bursty human activity patterns," *Physical Review E*, vol. 83, no. 3, p. 036102, 2011.

[209] R. M. Bond, C. J. Fariss, J. J. Jones, A. D. Kramer, C. Marlow, J. E. Settle, and J. H. Fowler, "A 61-million-person experiment in social influence and political mobilization," *Nature*, vol. 489, no. 7415, pp. 295–298, 2012.

[210] S. A. Myers, C. Zhu, and J. Leskovec, "Information diffusion and external influence in networks," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 33–41, ACM, 2012.

[211] K. Lerman and R. Ghosh, "Information contagion: An empirical study of the spread of news on digg and twitter social networks.," *ICWSM*, vol. 10, pp. 90–97, 2010.

[212] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, "Identification of influential spreaders in complex networks," *Nature physics*, vol. 6, no. 11, pp. 888–893, 2010.

[213] A. Baronchelli, R. F. i Cancho, R. Pastor-Satorras, N. Chater, and M. H. Christiansen, "Networks in cognitive science," *Trends in Cognitive Sciences*, vol. 17, no. 7, pp. 348 – 360, 2013.

[214] G. Rodi, V. Loreto, V. Servedio, and F. Tria, "Optimal learning paths in information networks," *Scientific reports*, vol. 5, 2015.

[215] M. Mesgari, C. Okoli, M. Mehdi, F. Å. Nielsen, and A. Lanamäki, ""the sum of all human knowledge": A systematic review of scholarly research on the content of wikipedia," *Journal of the Association for Information Science and Technology*, vol. 66, no. 2, pp. 219–245, 2015.

[216] A. Lih, "Wikipedia as participatory journalism: Reliable sources? metrics for evaluating collaborative media as a news resource," *Nature*, 2004.

[217] D. Fallis, "Toward an epistemology of wikipedia," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 10, pp. 1662–1674, 2008.

[218] R. Schroeder and L. Taylor, "Big data and wikipedia research: social science knowledge across disciplinary divides," *Information, Communication & Society*, vol. 18, no. 9, pp. 1039–1056, 2015.

[219] C. Okoli, M. Mehdi, M. Mesgari, F. Å. Nielsen, and A. Lanamäki, "The people's encyclopedia under the gaze of the sages: A systematic review of scholarly research on wikipedia," *Available at SSRN 2021326*, 2012.

[220] R. West, A. Paranjape, and J. Leskovec, "Mining missing hyperlinks from human navigation traces: A case study of wikipedia," in *Proceedings of the 24th international conference on World Wide Web*, pp. 1242–1252, International World Wide Web Conferences Steering Committee, 2015.

[221] A. Paranjape, R. West, L. Zia, and J. Leskovec, "Improving website hyperlink structure using server logs," *arXiv preprint arXiv:1512.07258*, 2015.

[222] Wikimedia, "Wikipedia article dumps," 2016. [Online; accessed 16-May-2016].

[223] Wikimedia, "Wikipedia page counts," 2015. [Online; accessed 16-May-2016].

[224] M. A. Porter, "Small-world network," vol. 7, no. 2, p. 1739, 2012. revision 153247.

[225] A. Capocci, V. D. Servedio, F. Colaiori, L. S. Buriol, D. Donato, S. Leonardi, and G. Caldarelli, "Preferential attachment in the growth of social networks: The internet encyclopedia wikipedia," *Physical Review E*, vol. 74, no. 3, p. 036116, 2006.

[226] T. P. Peixoto, "Efficient monte carlo and greedy heuristic for the inference of stochastic block models," *Physical Review E*, vol. 89, no. 1, p. 012804, 2014.

[227] O. S. Collaboration *et al.*, "An open, large-scale, collaborative effort to estimate the reproducibility of psychological science," *Perspectives on Psychological Science*, vol. 7, no. 6, pp. 657–660, 2012.

[228] D. Lakens and E. R. Evers, "Sailing from the seas of chaos into the corridor of stability practical recommendations to increase the informational value of studies," *Perspectives on Psychological Science*, vol. 9, no. 3, pp. 278–292, 2014.

[229] E. Miguel, C. Camerer, K. Casey, J. Cohen, K. M. Esterling, A. Gerber, R. Glennerster, D. P. Green, M. Humphreys, G. Imbens, D. Laitin, T. Madon, L. Nelson, B. A. Nosek, M. Petersen, R. Sedlmayr, J. P. Simmons, U. Simonsohn, and M. Van der Laan, "Promoting transparency in social science research," *Science*, vol. 343, no. 6166, pp. 30–31, 2014.

[230] D. B. Resnik, "A pragmatic approach to the demarcation problem," *Studies in History and Philosophy of Science Part A*, vol. 31, no. 2, pp. 249 – 267, 2000.

[231] A. Strauss and J. Corbin, *Basics of qualitative research: Techniques and procedures for developing grounded theory.* Sage Publications, Inc, 1998.

[232] G. Gorer, "The pornography of death," *Encounter*, vol. 5, no. 4, pp. 49–52, 1955.

[233] M. Zuckerman and P. Litle, "Personality and curiosity about morbid and sexual events," *Personality and Individual Differences*, vol. 7, no. 1, pp. 49–56, 1986.

[234] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*, pp. 722–735, Springer, 2007.

[235] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, ACM, 2008.

[236] R. I. Dunbar, "Gossip in evolutionary perspective.," *Review of general psychology*, vol. 8, no. 2, p. 100, 2004.

[237] E. Anderson, E. H. Siegel, E. Bliss-Moreau, and L. F. Barrett, "The visual impact of gossip," *Science*, vol. 332, no. 6036, pp. 1446–1448, 2011.

[238] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 177–187, ACM, 2005.

[239] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[240] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[241] E. Tulving, "Episodic and semantic memory 1," *Organization of Memory. London: Academic*, vol. 381, no. 4, 1972.

[242] E. Tulving, "Episodic memory: From mind to brain," *Annual review of psychology*, vol. 53, no. 1, pp. 1–25, 2002.

[243] K. Benzi, B. Ricaud, and P. Vandergheynst, "System and method for contextual knowledge retrieval and display," 2016. Pending patent.

[244] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 497–506, ACM, 2009.

[245] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[246] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[247] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents.," in *ICML*, vol. 14, pp. 1188–1196, 2014.

[248] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[249] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*, pp. 137–142, Springer, 1998.

[250] M. Zhang, T. Hall, and N. Baddoo, "Code bad smells: a review of current knowledge," *Journal of Software Maintenance and Evolution: research and practice*, vol. 23, no. 3, pp. 179–202, 2011.

[251] F. C. Fang, R. G. Steen, and A. Casadevall, "Misconduct accounts for the majority of retracted scientific publications," *Proceedings of the National Academy of Sciences*, vol. 109, no. 42, pp. 17028–17033, 2012.

[252] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *arXiv preprint arXiv:1502.03044*, vol. 2, no. 3, p. 5, 2015.

[253] A. McGregor, "Graph stream algorithms: a survey," *ACM SIGMOD Record*, vol. 43, no. 1, pp. 9–20, 2014.

[254] T. E. F. P. for Research and Innovation, "Communicating EU research and innovation guidance for project participants - Horizon 2020." http://ec.europa.eu/research/participants/data/ref/h2020/other/gm/h2020-guide-comm_en.pdf, 2014.

[255] R. A. Duschl, H. A. Schweingruber, A. W. Shouse, *et al., Taking science to school: Learning and teaching science in grades K-8*. National Academies Press, 2007.

[256] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[257] D. Milgram, "How to recruit women and girls to the science, technology, engineering, and math (stem) classroom.," *Technology and engineering teacher*, vol. 71, no. 3, pp. 4–11, 2011.

[258] E. W. Foundation, "How girls hold themselves back in Computer Science." http://www.epflwishfoundation.org/news/how-girls-hold-themselves-back-in-computer-science, 2014.

[259] K. Benzi, B. Ricaud, and P. Vandergheynst, "Unveiling collective knowledge retrieval processes," 2016. In preparation.

List of contributions

Publications

1. K. Benzi, B. Ricaud, and P. Vandergheynst, "Principal patterns on graphs: Discovering coherent structures in datasets," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 160–173, 2016

2. K. Benzi, V. Kalofolias, X. Bresson, and P. Vandergheynst, "Song recommendation with non-negative matrix factorization and graph total variation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2439–2443, IEEE, 2016

3. A. Alahi, P. Vandergheynst, and K. Benzi, "System and method for media library navigation and recommendation," 2014. EPFL Patent WO 2014/002064A1

4. K. Benzi, B. Ricaud, and P. Vandergheynst, "System and method for contextual knowledge retrieval and display," 2016. Pending patent

5. K. Benzi, M. Defferrard, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2016

6. A. Griffa, K. Benzi, B. Ricaud, P. Vandergheynst, J.-P. Thiran, and P. Hagmann, "Mapping resting-state dynamics on causal multilayer graphs: a combined functional and diffusion MRI approach," *NeuroImage. In review.*, 2016

7. K. Benzi, B. Ricaud, and P. Vandergheynst, "Unveiling collective knowledge retrieval processes," 2016. In preparation

Press

- **Exploring the Star Wars expanded universe.**
  http://kirellbenzi.com/blog/exploring-the-star-wars-expanded-universe/.
  Featured in more than 150 websites in 10 languages, including: *Gizmodo, Engadget, Daily Mail, The MarketPlace, Wired.it, TechRadar, Co.Design, Phys.org, Le Temps, 20minutes, Muy Interesante, EPFL press*. 2016.

- **Evolving Pokemon network.**
  http://kirellbenzi.com/blog/evolving-pokemon-network/
  Featured in: *VICE, Phys.org, 20minutes, EPFL press.* 2016.
- Le court du jour “Imaginez un monde”: les réseaux. *RTS, Swiss national TV.* 2015.
- EPFL software is able to trace paths amidst a music jungle. *EPFL press.* 2013.

Awards

- Innovation by Design 2016 Finalist. Exploring the Star Wars Expanded Universe.
- CVPR 2012 Open source winner for FREAK: Fast Retina Keypoints implementation in OpenCV.

KIRELL BENZI

DATA SCIENCE, ART

kirell.benzi@gmail.com
http://kirellbenzi.com
28 years old

Profile

My primary interests revolve around data science, particularly in knowledge discovery and data mining using networks, the core part of my Ph.D. Looking for opportunities as data valorization specialist focusing on brand image, API valorization, and marketing using algorithms. I am currently running an exhibition called "Singular Networks" in between art and science. Combining both data analysis and network visualization, I try to captivate the audience by showing that algorithms, apart from their scientific necessity, also generate emotions.

Skills

Data science	Python	NoSQL
Data visualization	C++	Recommendation systems
Network analysis	Javascript	Deep learning

Experience

LTS2 - École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Data Scientist **2012-present**

Combined with my Ph.D. in Network Science. Implementation of large-scale algorithms applied to recommendation, knowledge mining, neuroscience, pedestrians tracking and social network analysis.

Genezik, Lausanne, Switzerland

Co-founder & CTO **2012-2016**

Genezik is a distributed smart playlist generator based on personal musical tastes. By analyzing the "DNA" of each song such as rhythm, timbre, pitch, energy combined with machine learning algorithms, Genezik creates a musical journey adapted to your mood from your library.

Metamedia - EPFL, Lausanne, Switzerland

R&D engineer **2011-2012**

Development of immersive and interactive data visualizations for the Montreux Jazz Festival archive.

Education

LTS2 - École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Ph.D. in Network Science **2012-2016**

Obtention of a Ph.D. in the Signal Processing Laboratory 2 directed by Prof. Pierre Vandergheynst. My Ph.D. thesis is called "From Recommender Systems to Spatio-temporal Dynamics with Network Science".

ECE Paris, Paris, France

BSc and MSc in Communication Systems **2006-2011**

French engineering school specialized in IT and software engineering. Obtained the French title: "Ingénieur en Systèmes d'Information et Réseaux" after a Bachelor and Master's degree.

Recognition

Exploring the Star Wars expanded universe

Featured in 150+ articles worldwide including Gizmodo, Engadget, Daily Mail, The MarketPlace, Wired.it, TechRadar, Co.Design, Phys.org **2016**

Evolving Pokemon network

Featured in: VICE, Digital Trends, Phys.org, 20minutes **2016**

Innovation by Design Awards 2016 finalist

Organized every year by Co.Design @ Fast Company, selected over 1800 submissions **2016**

Interests

Music, Salsa, Digital art, Snowboard, Golf