UNIVERSITÀ DEGLI STUDI DI MILANO

Department of Computer Science
Ph.D. in Computer Science – XXIX cycle

Modeling and Mining Complex Networks with Feature-Rich Nodes

Advisor and Ph.D. coordinator:
Prof. Paolo Boldi

Co-advisor:
Prof. Sebastiano Vigna

Doctoral dissertation of:
Corrado Monti

Academic year 2015/2016

Contents

Abstract

Real-world complex networks describe connections between objects; in reality, those objects are typically endowed with properties and attributes (hereby called features). How does the presence or absence of such features interplay with the network link structure? Although this situation is truly ubiquitous, literature on the subject is rather sparse. Moreover, of those who dealt with this kind of feature-rich graphs, many considered homophily as the only possible gear of transmission between nodes and features; others, instead, devised very complex models, unfit to scale to the size of common real-world networks. In this work, we focus on a model where interactions between features can foster or discourage link formation. The idea is to be able to represent a wide range of scenarios – not only homophily and heterophily, but a more general approach on how the interplay between all the features of two nodes can determine the link between them.

To this aim, we first define and analyze an ad-hoc statistical model for features. Features evolve through a modified Indian buffet process: every new node introduces new features, while also copying the popular ones from previous nodes, in a rich-get-richer fashion. The rich-get-richer effect is here moderated by the presence of a fitness parameter attached to each node: the idea is that some nodes are supposed to be more effective than others in transmitting their features. Directed links are then formed as a consequence of the features they bear. We show how this model displays the same global topological properties of a real-world complex network, in terms of degree distribution, distance distribution, connectivity, etc.

As a further type of experiment, we will assume to know the directed graph and which features characterize each node – we will show how this is the case in many real-world networks. From this data, we provide estimators to evaluate the parameters of our model; moreover, we show how the model can naturally define a likelihood-based technique to gain some information about which nodes were more successful in transmitting their features.

Streamlining the model, then, we devise a simple yet fast algorithm to build the feature-feature latent matrix describing the way features interact; in such a matrix W, the element $W_{i,j}$ is positive if feature i tends to connect to feature j. Specifically, we propose two algorithms, one assuming the independence of features and based on a Naïve Bayes approach; the other relaxes this assumption and is based on perceptron-like learning techniques. The latter (called *Llama*, Learning LAtent feature-feature MAtrix) can process hundreds of millions of links in minutes.

Finally, we propose an approach to solve a somehow dual problem: if a node in the network has unknown features, but we know its links, how can we discover the feature it may have? We

will show how the same feature-feature latent matrix can be the core of a shallow neural network which can help solve this problem.

We then proceed to employ these techniques in some concrete use cases, to show how they can help concretely in some graph-mining tasks. Many scenarios can be indeed described by a directed network with feature-rich nodes. We will take into consideration networks of different types: semantic networks and citation networks.

For the first kind of network, we will look at the Wikipedia link graph. Wikipedia is commonly used as an open knowledge base in many applications; the links from an entity to other entities are often of fundamental importance (e.g., in named entity disambiguation). Wikipedia provides a full taxonomy that ascribe entities to a set of Wikipedia categories; since this categorization is user-generated and extremely noisy, we propose a general technique to cleanse it, in order to consider only the k "most important" categories, and re-categorize the entities accordingly. Such cleansed categories will then serve as features for the nodes in the link graph. We investigate how much k features can explain the links, and discover that they provide an excellent way to mine "surprising" (a.k.a., *unexpected*) links between entities, much better than text-based approaches. We call this approach *LlamaFur*, Learning LAtent feature-feature MAtrix to Find Unexpected Relations.

For the second kind of network, we consider a citation network between papers: here, each paper can be described by many possible types of characteristics – e.g., its keywords and the affiliations of its authors. Each of these types have a different power of description with respect to the network: we show how our framework can be used to evaluate it.

Chapter 1

Introduction

1.1 Motivation

The necessity to model networks has been addressed by many works in the last decades. Its impact on many practical matters – from epidemiology [36] to financial stability [20] – has been invaluable. However, there has been, in comparison, far less work devoted to model networks where nodes present a set of properties of their own, despite this being the natural state of many real datasets: economical networks usually are accompanied with some data about each economic actor (e.g., the sectors in which it operates); social networks often have some indication about the interests of each user; in the interactome—i.e., the network describing interactions between genes—many properties for each gene are available, and so on.

In many tasks, empirical evidence has shown that the interconnection between the network and some property of its nodes is of primary practical importance. In epidemiology, it is common practice in recent years to study contact networks (i.e., networks of physical contacts among people) and to include, for each person involved, some of its social characteristics (for example, gender, age group [93] or profession [98]). Studying how these features interact with the physical contacts—and, therefore, with the spread of a disease—has lead to the so called *"Who Acquires Infection From Whom"* matrices [140]. However, connecting the presence of links in a network to some feature of its node has a longer history, that traces back to sociology; the way in which features (e.g. ethnicity or gender) intertwine with social connections has been called and studied as *"mixing pattern"* [131]. We will see a more complete review of these studies in section 2.3.

Many of these real cases backed with empirical data the role of node features in in the formation of links in a network; also, they showed that in many cases there are *different* features that interact within each other, fostering the creation of links. In other words, *homophily*—that is, similar features bring connections—is not the only valid mechanism behind link formation. Nonetheless, many models and data mining techniques for networks with feature-rich nodes assumed that links resonates with similarity between features. There is a need, therefore, for models able to represent interactions between different features; where pairs of different features could result in fostering or discouraging a link.

Moreover, many of the models and the techniques able to represent this scenario are tailored

to small and medium-sized networks. In this information age, instead, many datasets consist in very large networks, with millions or tens of millions of nodes—we will see some examples throughout this work. Our goal is therefore to develop models and data mining techniques able to work with large networks, considering the features their nodes may have, and representing interactions between pairs of features.

1.2 Problem and hypotheses

The problem we are going to deal with in this work is therefore that of modeling *large* networks with feature-rich nodes, with a model able to handle *overlapping* features (a node can have multiple features, as it is often the case in reality) and non-homophiliac relations (different features can foster the formation of the links). In order to simplify this scenario, we will assume that features of nodes are *binary*—that is, we are interested in cases where it is the presence or absence of features that influences the network. This assumption is analogous to the choice of dealing with *unweighted* graphs—in fact, the association between nodes and features can be represented exactly with an unweighted, bipartite graph. We will show that this assumption can give us algorithms that can scale to large networks, while capturing a plethora of scenarios—namely, all the examples we mentioned before, and many others we will see in this work.

We will show through numerical simulations and real data how such characteristics in a model can help in giving helpful insights in practice, leading to algorithms able to give us new information on many real cases of complex networks. Particularly, we will show how a simple feature-rich model of graphs is consistent with machine learning algorithms able to scale up to very large networks; moreover, we will show its practical significance in some different context, from network analysis—where we will use them to assess which types of features can *explain* the links in a network—to anomaly detection—where surprising information in a corpus of documents can easily be found as deviation from this model.

Our main hypothesis can therefore be summarized as follows: the analysis of networks can benefit from considering the features associated with the nodes, even in the case the network is of large size (up to tens of millions of nodes) and if the relations between features and links goes beyond simple homophily.

1.3 Main contributions and related publications

Now that we have clearly defined our playing field, let us sketch briefly the novel contributions of this work in this field.

Our first contribution is a refinement over previous models for feature-rich network models. In particular, we will conceive an innovative model for the evolution of features, by keeping in consideration the fact that different nodes can have different abilities to spread their features in time. We will use this fact to propose (and compare) different algorithms to estimate this ability for each node. Then, we will see how this model for feature evolution can lead to a graph model, by analyzing the global properties of the produced graphs. We will show how such an approach can lead to networks that resemble a given real-world complex network in terms of their global

properties (mainly, their degree distribution and their distance distribution). Such considerations resulted in a journal publication [30].

As we will see across this work, a natural framework for networks with feature-rich nodes can efficiently be connected to machine learning and data mining algorithms: connecting those two realms is one of the main contributions of this work. In particular, we will trace a formal link between perceptron-like algorithms and the estimation of parameters in our model for such networks. Such a connection is then exploited in different ways. First, it give us simple algorithms that are fast and can scale; second, it give us formal guarantees with respect to our model for such algorithms, by allowing us to reuse previous analysis in this context.

We will then see how similar frameworks can be applied on link prediction and on feature prediction, first by seeing the links as prediction targets when features are completely known, and then by seeing features as such when links are completely known.

Applying these algorithms to real world use cases is then another contribution of this work, explored in its last part. We will see how useful information can be extracted from citation networks, by applying our model. For example, we will be able to compare the ability of different types of characteristics of scientific works in predicting the citations they make. We present this example mainly as a showcase of how our model and techniques could be applied, in principle, to a wide range of social network (and content network) analysis. Determining the impact of features on the links of a network could be employed in many different applications: social scientists can have more scalable and well-tested tools to measure how social features determines our friendships, tracking with precision the phenomenon of "filter bubbles". Applying these tools on biological networks, like the aforementioned interactome, could give biologists clues on which properties a certain gene could have, basing on its interaction with other genes. These applications are promising future directions for our work.

Much of our attention will then be, in this work, devoted to semantic networks, exemplified by the case of the Wikipedia Link Network. Our technique for deriving a clean categorization of concepts within Wikipedia has enabled us to have a low amount of noise in the features we use, and has resulted in a conference paper [31]. The technique we propose is rooted in network analysis, and it is well-suited as a preprocessing phase to all the data mining tasks that make use of categories for Wikipedia articles.

The usage of such features, combined with our model, gives rise to an application in anomaly detection: that is, we analyze how our model can be read as a way of finding surprising links in a semantic network. We will see how our method outperforms both standard link prediction approaches and textual-based ones in this task. This result has been described in a conference paper [8].

We wish to highlight, finally, how a main contribution of this work is how many data mining algorithms for graphs—link prediction, label prediction, social network analysis, and node ranking—can be read in a unified way, thanks to a proper complex network model, able to represent feature-rich nodes. The link between modeling complex networks and designing learning techniques to mine information from them is, we believe, an exciting and promising field.

1.4 Organization

The present work is organized in three parts, of different nature.

In part I, we will devote our attention to mathematical models for networks and features. We first describe network models and particularly feature-rich network models already present in literature (chapter 2), highlighting their contributions, as well as their drawbacks, from our point of view. Then (chapter 3) we carefully describe the theoretical framework employed throughout this work, by detailing the mathematical relations we will assume to exist between links and features in a feature-rich network. Finally, we will have a closer look on a formal model for the generation of features (chapter 4), that in the end will give us a way to simulate from scratch realistic feature-rich networks.

In the second part of this work, we will describe algorithms that can be fruitfully applied to our model, and specifically to finding latent and unknown variables of the model given a set of available data; all the algorithms we propose in this part will be tested on synthetic data, thanks to the capability of our model to generate realistic feature-rich networks. By showing that these algorithms can be successfully applied to synthetic data, we can be confident in applying them (as we will see later) to a real case that our model can fit. Specifically, we will first look at how to infer the relation between all the pairs of features in chapter 5; we will see how this problem can be recast as a machine learning problem, and how it is fundamentally linked to link prediction. Then, we will look at the prediction of unknown features in chapter 6, and see how neural networks can be applied to this task to solve the issue of estimating non-homophiliac relations.

In the third part of this work, we will apply our model and our mining algorithms to real-world examples. In chapter 7 we will look at citation networks, and see how their adherence to our model heavily depends on the chosen set of features. In chapter 8, instead, we will analyze the case of semantic networks, and prove how our techniques can be efficiently applied to this case, even if their size is very large.

Finally, in the last chapter, we will outline the conclusions of this work.

Part I

Models

Contents

Chapter 2

State of the art

2.1 Introduction

Complex networks are a unifying theme that emerged in the last decades as one of the most important topics in many areas of science; the starting point is the observation that many networks arising from different types of interactions (e.g., in biology, physics, chemistry, economics, technology, on-line social activity) exhibit surprising similarities that are partly still unexplained. The quest for a model that is able to explain, describe, analyze and simulate those real-world complex networks is of uttermost practical, as well as theoretical, interest.

The general idea of a network model can be seen as a function that, given a set of parameters, returns a probability distribution over graphs. The most classical random graph is the Erdős-Rényi model [67]. In this model, the parameters are $n \in \mathbb{N}$ and $p \in [0, 1]$, respectively the number of nodes and the probability of an edge. Then, the model considers a set N of n vertices, and creates each edge (x, y) with $x \in N, y \in N$, independently, with probability p. This model was developed by Paul Erdős-Rényi and Alfred Rényi in 1959; a very similar model was proposed in the same year by Edgar Gilbert [81].

This classical probabilistic model, however, soon revealed itself unfit to describe complex networks because, for example, it fails to produce a power-law degree distribution. The general approach of the subsequent attempts is to produce probabilistic frameworks (typically with one or more parameters) giving rise to networks with statistical properties that are compatible with the ones that are observed in real-world graphs: degree distribution is just one example; other properties are degree-degree correlation, clustering coefficients, distance distribution, etc. [99].

2.2 Complex networks models

The Barabási-Albert model. The Barabási-Albert model [17] was one of the first attempts to try to obtain more realistic models, where the idea of *preferential attachment* was first introduced: nodes tend to attach themselves more easily to other nodes that are already very popular, i.e. with an high number of links.

The idea is that nodes come to the network each at a different time step; then, when a new

node joins the network, it is more likely to do so by linking to a well-connected node. Barabási and Albert assumed that, in fact, the probability of linking a certain node is directly proportional to its degree. Thinking to a web graph, for example, it means that when a new page is created, a link on that page is selected uniformly among *existing links*, leading thus to this proportionality assumption. This rule establishes a "rich-get-richer" effect.

To express formally this basic model, let us consider the formation of the network as a sequence of discretized time steps $0, 1, \ldots, t$. At time step 0, the graph G_0 is composed by a fully-connected clique of n_0 nodes: its set of edges L_0 contains all the possible edges among the n_0 nodes contained in the initial set of nodes N_0. The number n_0 is a parameter of the model: we can assume e.g. $n_0 = 2$, and start with two connected nodes. At every time step, we add a new node. We will therefore define the set of nodes at time step $(t + 1)$ as:

$$N_{t+1} = N_t \cup \{i_t\}$$

At time step $(t + 1)$, the set of edges L_{t+1} is obtained by adding to the set of edges at the previous time step, L_t, a set of new edges incident to the new node i_t with this probability:

$$\mathbb{P}\Big((i_t, j) \in L_{t+1}\Big) = \frac{\deg(j, t)}{\sum_{j \in N_t} \deg(j, t)}$$

Where we indicated with the function $\deg(j, t)$ the degree of node j at time step t.

This model results in a power-law distribution of degrees, and in a significantly more realistic average distance than the Erdős-Rényi model.

Many other models were developed to study and analyze graphs with a power-law distribution of degrees. Among the first, Aiello *et al.* [4] and Kumar *et al.* [119] both studied this class of graphs. Many described different strategies to obtain a small-world, realistic complex network; for example Kleinberg [118] and Kumar *et al.* [120] devised networks that evolved by "copying nodes".

The Jackson-Rogers model. We will now describe a similar strategy, showcasing well the approach dubbed "friends of friends". This model was developed in 2007 by Jackson and Roger [100]. Their model obtains a "rich-get-richer" effect by observing what happen in a real social network: when a new person is introduced to the network, it will form some connections for random causes—i.e., external to the social graph, an aspect they summarize in the phrase "meeting strangers"—and some other connections by getting introduced by a friend to their own friends—i.e., a "friend of friends" aspect.

The friend-of-friends aspect naturally leads to the preferential attachment behavior of the Barabási-Albert model. However, this model is richer: the amount of positive feedback mechanism with respect to degrees is in fact tunable, by means of controlling the amount of "strangers met" over the amount of "friends of friends".

Let us encapsulate this decision in the parameter $a \in [0, 1]$. This model as well can be seen as a sequence of discretized time steps. Again, we add one node at a time, so that the set of nodes at time step $(t + 1)$ is $N_{t+1} = N_t \cup \{i_t\}$. Let us then define a parameter k, that will represent

the number of arcs that will be added at each time step. Then, when a node i_t joins the set of nodes N_t, it will add to the set of links L_t the following connections:

- $k \cdot a$ links of the form (i_t, j), where j is chosen from N_t uniformly at random (the "strangers met").
- $k \cdot (1 - a)$ links of the form (i_t, j), where j is chosen uniformly at random from the set $\{j \mid \exists l \text{ s.t. } (j, l) \in L_t\}$ ("friends of a friend").

As Jackson and Roger [100] point out, this model leads to an exponential degree distribution when $a = 1$, and to a power law distribution when $a = 0$; in this sense, it permits a finer-grained fit to data. This is particularly useful when considering that recent works are increasingly showing that, despite being so-called "fat-tails" distribution, many complex networks fit other distributions (e.g., log-normal) better than the classical power law [83, 99].

In the past two decades, many other models of complex network (dynamic or not, and directed or not) have been proposed in literature. A complete and accurate overview was authored by Jackson [99]. However, in this work we wish to focus on a different type of graphs models: models for feature-rich networks.

2.3 Feature-rich complex networks

Complex networks arise, in reality, between concrete objects, not characterized solely by their connections – as in the pure graph models we have seen – but also by some kind of properties and attributes. In the past, this fact has been long neglected in the literature of graph models, for computational reasons and for the absence of such large and rich dataset. Nowadays, however, when a graph is emerging from data, very often some kind of information is also available for its nodes. For example, social networks originating from virtual communities can mark their user with locations, with interests or with some content they generated; network obtained from academic works (e.g., citation networks) can describe each node with keywords, affiliations of authors, and so on. The same consideration can be done for many biological networks: the interactome among molecules can be enriched by chemical properties, or metabolic information. We will see many other examples throughout this work.

Empirical evidence. The interplay between such features and links has been investigated separately in different fields. In many cases, interpreting links in a network as a result of features of each node has in fact a solid empirical background. For example, the dualism between "persons and groups" as an underlying mechanism for social connections was investigated by Breiger [35] in 1974. Within sociology, the simple phenomenon of homophily – "similarity breeds connection" – has received a great deal of attention: McPherson *et al.* [131] presented evidence and investigated the role of homophily in social ties; considered features included race and ethnicity, social status, and geographical location. Bisgin *et al.* [25] studied instead the role of interests in online social media (specifically, Last.fm, LiveJournal, and BlogCatalog), finding however that the role of interests as features is weak on those online networks—at least when considering homophily.

In some fields, more complex behavior than homophily has been studied as well. Tendencies of such kind, where nodes with certain features tend to connect to other type of nodes, are called *mixing patterns* in sociology and are often described by a matrix, where element (i, j) describe the relationship between a feature i and a feature j. In epidemiology, they have proven to be greatly beneficial in analyzing the spread of contagions. For example, they appeared to be a crucial factor in tracking the spread of sexual diseases [12] as well as in modeling the transmission of respiratory infections [140]. For this reason, such matrices are also called "Who Acquires Infection From Whom" (WAIFW) matrices, and have been empirically assessed in the field through surveys [93] and with wearable sensors [98]. In biology and bioinformatics, a seminal study by Menche *et al.* [133] highlighted the connections between the interactome, the network of the physical and metabolical interactions within a cell, and the diseases each component was associated with, observing a clustering of disease-associated proteins.

The empirical evidence presented in various fields, combined with the availability of large datasets in the web, and the increase of computational resources, fostered some investigation of models of graph endowed with features.

Fitness models. Earliest models incorporating some kind of features characterized each node with a single, real-valued feature. This feature was variously called the "fitness" of nodes, or more generally a "hidden variable" of each node. An early example is the Bianconi-Barábasi model [24] from 2001. This model sparkled from the necessity to fix a behavior of the Barabási-Albert model that did not match common experience [55]: in the BA model, only the first nodes have a chance to become hubs (i.e., nodes with a high degree), while in many practical examples (e.g., the World Wide Web) many of the largest hubs have joined the network late in time. To fix this, Bianconi and Barabási introduced the *fitness value* into the BA model: the fitness value acts as a multiplier in the previous model, and the effect is that a node with a high fitness value will prevail (in terms of degree) on the others, also if it joined the network later. According to Pastor-Satorras and Vespignani [150], this behavior takes inspiration from evolutionary models: the intrinsic fitness value that is assigned to each node embodies all the properties other than the degree.

However, this BB model still relies on the BA model principle (rich-get-richer) in order to generate a small-world network; the additional feature of the nodes is grafted on it. In 2002, Caldarelli *et al.* [39] proposed a model entirely based on the fitness values, where the entirety of links are caused by the feature of the nodes; this means removing the Preferential Attachment from the previous models and relying only on a *"good-get-richer"* principle. They propose to consider the probability of a link between two nodes i and j as the result of a function $\phi(x_i, x_j)$ of their fitness values x_i and x_j. We will see how such a function will be similar to our context, of *multiple* features appearing in each node.

One of the main results of this work was that, surprisingly, even when fitness values do not follow a power-law, the model can generate scale-free networks. In other words, the feature-based model is sufficient *per se* in emulating the degree distributions of real networks. In this light, the Bianconi-Barábasi model model can be therefore thought of as a hybrid model, between the classic scale-free models and the new feature-based one.

Garlaschelli and Loffredo [75] applied the model to the World Trade Web, a weighted, directed graph describing commercial relationship between countries: a striking result is that using the GDP of each country as its real-valued feature fits the data for this network. It was also showed [74] that the same model can also fit a network of large market investments between shareholders.

A recent application of this model [51] proposed a mathematical framework which, assuming to know each node fitness values, and the in- and out-degrees of a small subset of nodes, it is able to reconstruct global properties of the network. Building upon the previously investigated good fit between the fitness model and the World Trade Web, they use their framework to show how different properties of this network can be estimated in this way. Notably, they are able to estimate the DebtRank [20], a measure proposed by Battiston *et al.* to evaluate system risk caused by financial leverage in an economical system.

Class models. However, in many contexts a single real-valued feature is not the most natural choice to represent existing data. A popular framework has been that of *latent class models*: there, every node belongs to exactly one "categorical" class, and this class influences the links it may have. The best-known example is the stochastic block model [147, 171]: it assumes each pair of classes has a certain probability of creating a link, and in the work they study how to infer those probability; they also investigate how to determine the class assignments, leading to a sort of community detection algorithm. Hofman *et al.* [97] devised a variation of this schema, by specifying only within-class probabilities and between-class probabilities. Another useful variation involve sharing only the between-class probability and specifying instead the within-class probabilities separately for each class, allowing to characterize each with a certain degree of homophily. Both these approaches exemplify the need to reduce the number of parameters of the original block model, in order to facilitate the estimation of its parameters. Kemp [111] and Xu [192] studied and applied a non-parametric generalization of the model which allows for an infinite number of classes (therefore called *infinite relational model*). It permits application on data where the information about class is not provided directly. They use a Gibbs sampling technique to infer model parameters. Among the examples they cite, Kemp *et al.* [111] put a strong focus on semantic networks, a topic which we will treat carefully in Chapter 8.

A well-known shortcoming of the class-based models is the proliferation of classes [136], since dividing a class according to a new feature leads to two different classes: if we have a class for "students" and then we wish to account for the gender too, we will have to split that class in "female students" and "male students". This approach is impractical and in many cases it leads to overlook significant dynamics. To overcome this limitation, classical class-based models have been extended to allow mixed membership [5]. Here, the model of classes remains, but with a fuzzy approach: each node can be "split" among multiple classes, and in practice represents class assignments become a probability distribution.

Feature models. Contrary to class-based models, *feature-based models* propose a more natural approach for nodes with multiple attributes: in those models, each node is endowed with a whole vector of features. Therefore, they can be seen as a generalization of class-based models: in fact,

in the special case where all the vectors have exactly one non-zero component, the model has the same expressive power of class-based ones. Features can be real-valued – as in the latent factor model [96] – or binary, where the set of nodes exhibiting a feature is sharp, and not fuzzy [132].

Many works in this direction proposed models that only allow for homophily, forbidding any other interaction among features. A seminal example are *affiliation networks* [122] by Lattanzi and Sivakumar; in that work, a social graph is produced by a latent bipartite network of *actors* and *societies*; links among actors are fostered by a connection to the same society. Gong *et al* [83] analyzed a real feature-rich social network – Google+ – through a generative, feature-based network model based on homophily.

Our attention will be focused instead on models able to grasp more complex behavior than homophily, following the aforementioned empirical evidence from social networks, epidemiology and bioinformatics.

MAG model family. Within this realm, an important line of work has been explored by Kim, Leskovec and others [112], under the name of *multiplicative attribute graphs*. There, every feature is described by a two-by-two matrix, with real-valued elements. Those elements describe the probabilities of the creation of a link in all the 4 possible cases of that feature appearing or not appearing on a given pair of nodes. An example would be the following: let us think of a citation network, where each node (author) is tagged with some features (the fields she is working in); let us suppose the total number of tags is m. This model describes the interaction between those features with m different matrices of size 2×2; one of these matrices could be related for example to the feature "sociology"; the elements of this matrix would represent the probability of a link going from a sociology paper to another sociology paper; the probability of one from a sociology paper to a non-sociology paper; and so on.

Because of this description made by a number of so-called "link affinity matrices", this model can be thought of as a feature-rich special case of their previous Kronecker model [124]. A very expressive model, it has been further extended to include many other factors; notably, they modify it to be a dynamic model [9]: features can be born and die, and only alive features can have effects. However, the complexity of this model – and its large number of parameters – prevents it from being used on large-scale networks. They have proposed [113] an expectation-maximization algorithm to estimate the parameters of their base model; reported experiments are on graphs with thousands of nodes. In the dynamic version, they report examples on hundreds of nodes; e.g., they find that by fitting the interactions of characters in a Lord of the Ring movie, their features effectively model the different subplots. Instead, in our work we wish to treat networks of much larger cardinality: in the experimental part, we will show examples with many millions of nodes, for which we wish to estimate model parameters in minutes.

MGJ model family. In 2009, Miller, Griffiths and Jordan [136] proposed a feature-based model to describe the link probability between two nodes by considering interactions between all the pairs of features of the two nodes. In the same example of the last paragraph, we would have one matrix of size m^2; one of these elements would correspond, for example, to the feature pair "sociology" and "statistics", expressing how likely would be a link from a sociology paper to

a statistics paper. We will explore this theme in depth in the rest of this work.

The authors show how by inferring features and their interactions on small (hundreds of nodes) graphs, they are able to predict links with a very high accuracy (measured through the Area Under ROC curve). The estimate technique they propose is not exact (since it would be intractable [86]), but it is based on a Markov Chain Monte Carlo method [79].

Their model can be interpreted as a generative model; they chose, however, not to investigate its structural properties in terms of the resulting network structure. Subsequent work [152] focused on this goal, being able to generate feature-rich graphs with realistic features, but they will not try to estimate the latent variables of the model necessary to predict links. In this work we will use the same model to do both (in particular, we will evaluate both purposes empirically in Chapter 7).

Despite the capabilities of the model, however, the choice of using a MCMC technique revealed itself inadequate to work on datasets larger than some hundreds of nodes. As noted by Griffiths *et al.* in 2010 [86], there is a need for computationally efficient models as well as reliable inference schemas for modeling multiple memberships. Menon *et al.* [134] noted how the inadequacy in handling large graphs underpinned this work, and many similar ones, and ascribed this flaw to the MCMC method.

There has been, since, a certain amount of work on how to apply this model on larger graphs. The two aforementioned works, for example, tried to solve this problem in different ways. A simpler model was described by Griffiths and Ghahramani [86]: they removed from the original model [136] the ability to describe negative interaction between features; also, they fixed the activation function of the model (a component which we will carefully explain in Chapter 3); in this way, they obtained a more computationally efficient model, able to run on graphs of about ten thousand nodes.

Menon *et al.* [134] slightly enriched the model, by introducing a bias term for each node; then, they propose a new estimation technique, based on stochastic gradient descent. A main focus is on avoiding undersampling non-links to overcame class imbalance, since despite it being "the standard strategy to overcome imbalance in supervised learning" it has the "disadvantage of necessarily throw[ing] out information in the training set". To overcome this problem (that we solved instead in the standard way of undersampling, see chapter 5), they design a sophisticated approach centered on the direct optimization of the area under the ROC curve.[1] With this technique, they can handle graphs with several thousands of nodes. Similar results [63] were obtained, with a different approach, on graphs of the same cardinality of nodes, where the authors propose an SVM-based estimation of parameters, and report it being able to run on graphs as large as two thousands of nodes in 42 minutes.

In this work, we too will propose a model fundamentally identical to the Miller-Griffiths-Jordan model, that we will thoroughly describe in Chapter 3. There we will also propose some further considerations on that model. In the subsequent chapters, then, we will propose in a single framework a generative model, able to generate realistic, feature-rich complex network, and estimation techniques for its parameters, able to run on graphs of millions of nodes. In

[1] Recent works [193] have indicated empirically as well as theoretically how employing this measure in link prediction leads to severely misleading results.

particular:

- In chapter 4 we will outline a generative model for features and for the graphs, and we will test how realistic are the graphs it can generate.
- In chapter 5 we will propose various techniques (mostly based on online learning and perceptrons) to estimate the main parameter of the model, i.e., the feature-feature matrix. Across the experimental part (specifically, chapters 7 and 8) we will show how these techniques are able to work on graphs of millions to about ten million nodes in less than 10 minutes.
- In Chapter 6 we will propose a neural-network method to infer features if they are partially known.
- We will integrate our framework for feature-rich graphs within the context of node ranking in section 4.6.

Finally, across chapters 7 and 8, we will try our methods on real networks whose size is unmatched by previous literature.

Chapter 3

Our framework

3.1 Introduction

As a member of the feature-based family, our model of choice stems from the observation that, in reality, complex networks are typically endowed with features on their nodes: since networks describe links among real-world objects, those indeed have properties and attributes. Very often, those attributes can be simply treated as a set of *features* a node can have or not, without any middle ground; therefore, we choose to consider features with binary values. This assumption allows for better tractability, both computational and mathematical, while still covering a vast number of concrete scenarios. Examples include friendship networks with topics of interest for each person; semantic relations with categories of each concept; economic exchanges between agents with the products produced by each; and – as we will see throughout this work – many others.

In many scenarios, a reasonable hypothesis is that the links in a network arise in some way from a complex interweaving of some features of the nodes. For example, in a co-authorship network, a link stems more easily between authors with similar interests; similarly, in a genetic regulatory network, links are affected by the different biological functions of the regulators.

In this regard, we would like our model to be able to capture not only homophily – where a link stems between nodes sharing *the same* features – but also more complex behaviors. For example, feature h could foster links to feature k also with $h \neq k$: e.g., in the case of semantic relations, categories "Movies" and "Directors" will often link to each other. If we consider directed networks, we would like this relationship to be directed as well: feature h could foster links towards feature k but not the other way around. For example, in a citation network, we could easily expect a paper within the sociology realm to cite a statistics paper, but a link in the opposite direction will be much harder to find. Finally, a pair of features could inhibit link formation: as *"Romeo and Juliet"* narrates, belonging to rival families could discourage the creation of a link in a long-term romantic relationship graph.

In section 3.2 we will detail a statistical model for building a graph structure from feature-rich nodes. In section 3.3, we will present alternatives for it by hybridizing it with well-known small-world graph models.

3.2 Feature-Feature probability model

The model we are going to describe here is able to represent all the aforementioned kinds of behavior, while at the same time being simple enough to be computationally useful and scalable, as we will show in the second part of this work. It is fundamentally the model proposed by Miller, Griffiths and Jordan [136]; however, we will use it extensively as a common frame of reference throughout this work, presenting new techniques – scalable methods for its parameters estimation, insights and experiments about the kind of graphs it is able to generate by simulation, concrete application in graph mining tasks, and so on.

Let us briefly present the main actors in our model. We will typically think, throughout this thesis, of the following objects as (at least partially) observable:

- The (possibly directed) graph $G = (N, L)$, where N is the set of n vertices, while $L \subseteq N \times N$ is the set of links.
- A set of m features F.
- A node-feature association $Z \subseteq N \times F$.

We will often treat these objects through their matrix equivalent representation. More precisely, $G = (N, L)$ will be represented as $\mathbf{L} \in \{0,1\}^{n \times n}$; Z as $\mathbf{Z} \in \{0,1\}^{n \times m}$, using an (arbitrary but fixed) ordering for nodes and features. Moreover, $A_{i,j}$ will refer to the element in the i-th row and j-th column of the matrix $\mathbf{A}$.

The – typically unobservable – objects that will define our network model will be the following:

- A matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$, that represent how features interact with each other. The idea is that a high value for $W_{h,k}$ means that the presence of feature h in a node i and of feature k in node j will foster the creation of a link from i to j. Conversely, a negative value will indicate that such a link will be inhibited by h and k. Naturally, the magnitude of $|W_{h,k}|$ will determine the force of these effects.

  We will refer to $\mathbf{W}$ as the *latent feature-feature matrix*.

- A monotonically increasing function $\phi : \mathbb{R} \to [0,1]$ that will assign a probability to a link (i, j), given the real number resulting from applying $\mathbf{W}$ to the features of i and j; we will call such a function our *activation function*, in analogy with neural networks [95].

The relationship between those actors is described formally by the following equation, that fully defines our model:

$$\mathbb{P}\Big((i,j) \in L\Big) = \phi\Big(\sum_h \sum_k Z_{i,h} W_{h,k} Z_{j,k}\Big) \tag{3.1}$$

In other words, the probability of a link is higher when the sum of $W_{h,k}$ is higher, where h, k are all the (ordered) pairs of features appearing in the considered pair of nodes. We will carefully analyze this equation in the following sections. The rest of this work will deal with how this framework can be useful in modeling a real network and in extracting useful information from it.

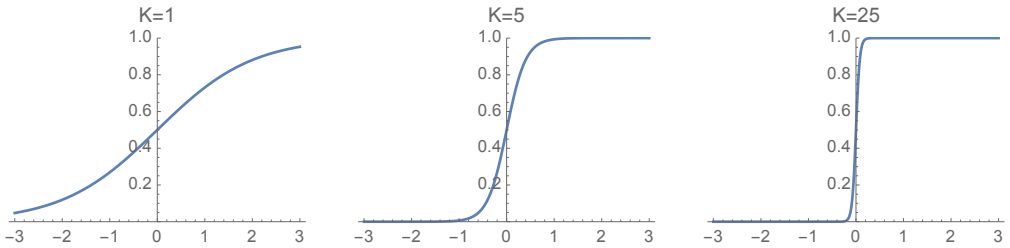


Figure 3.1: The activation function ϕ as a sigmoid, with different choice for K. K regulates its smoothness, and for $K \to \infty$ it approaches a step function.

3.2.1 Model parameters

Analysis of the latent feature-feature matrix. Let us point out how different construction for $\mathbf{W}$ can lead to many different kinds of interplay between links and features. The simplest case is $\mathbf{W} = I$ (the identity matrix). Since its only non-zero elements are those in the form (k, k), the only non-zero elements in the summation are those with $Z_{i,k} = Z_{j,k} = 1$. Therefore, the behavior of the model in this case is of pure *homophily*: the more features in common, the higher the probability of a link (remember that ϕ is monotonic).

More generally, elements of $\mathbf{W}$ like $W_{h,k} > 0$ will indicate a positive influence on the formation of a link between nodes with feature h and nodes with feature k. For example, we could reasonably expect a high $W_{nations,capitals}$ for a semantic graph. In the particular case of an undirected graph, we will have a symmetric matrix – that is, $W_{h,k} = W_{k,h}$ for all h and k.

$\mathbf{W}$ can be used to express also other concepts. If $\sum_k W_{h,k}$ is high, this fact will indicate that nodes with feature h will be highly connected – specifically, they will have a large number of out-links. A large sum for a column of $\mathbf{W}$, that is a large value for $\sum_k W_{k,h}$, will cause instead nodes with feature h to have many in-links.

Choice of the activation function. The activation function ϕ will regulate how the real numbers resulting from $\sum_h \sum_k Z_{i,h} W_{h,k} Z_{j,k}$ will be translated into a probability for the event $\big\{(i, j) \in L\big\}$.

Since we require ϕ to be monotonically increasing, its role is just to *shape* the resulting distribution – the "responsibility" for the outcome of $\big\{(i, j) \in L\big\}$ will heavily lie on $\mathbf{W}$. Nonetheless, we will show in section 4.5 that the specific distribution shaped by ϕ will have an impact on the link structure of the resulting graph.

Throughout this work, we will focus on activation functions that can be expressed as a sigmoid:

$$\phi(x) = \left(e^{K(\vartheta - x)} + 1\right)^{-1} \tag{3.2}$$

$\vartheta \in \mathbb{R}$ is the center of the sigmoid; $K \in (0, \infty)$ regulates its smoothness. Figure 3.1 depicts how K influences the resulting probabilities (when $\vartheta = 0$). We will look at both these quantities as *a priori* parameters of the model. The choice of ϑ affects the number of created links: we will see in section 4.5 how the density of the resulting graph can be controlled with a good approximation by controlling ϑ, thus shifting the assumption we make on ϑ to an easy-to-interpret assumption

on the number of links we want to create. In section 4.5, we will also show the effect of different K on the resulting network, for the specific case of pure homophily. We will see that in that case, a larger K results in more disconnected graph, and also in a cleaner power-law of the degree distribution.

For many applications, we will also extend the domain of K to the special value $K = \infty$, for which ϕ is the step function[1] $\chi_{(\vartheta,\infty)}$. This will make our model fully deterministic—all the probabilities become either 1 or 0. We will see how this simplification can become an important framework for mining information from a complex network.

3.2.2 An algebraic point of view

For some applications, it will be useful to consider the model expressed by (3.1) as a matrix operation. As introduced in the previous section, $\mathbf{Z}$ is the $n \times m$ node-feature indicator matrix.

With this notation, we can express (3.1) as

$$\mathbf{P} = \phi\Big(\mathbf{Z}\mathbf{W}\mathbf{Z}^T\Big) \tag{3.3}$$

Where ϕ here denotes the natural element-wise generalization of our activation function – i.e., it simply applies it to all the elements of the matrix. The resulting matrix $\mathbf{P}$ is a matrix that describe the probabilities of $\mathbf{L}$: that is, its element $P_{i,j}$ define the probability that $L_{i,j} = 1$ or equivalently that $(i, j) \in L$.

While this view is simple and concise, it may be of little use from a computational perspective. In concrete applications n will be very large; also, algorithms that could be of use in dealing *directly* with this representation do not run in linear time—the most notable example being matrix factorization (e.g., computing the SVD [183]).

It is useful, however, to view (3.3) separately for each row of the matrix. In practice, this means computing the set of out-links of a single node. This operation allows us to treat a single node at a time, permitting the design of *online* algorithms, requiring a single pass on all the nodes.

Moreover, this interpretation renders $\mathbf{W}$ a (possibly asymmetric) similarity function: if we represent nodes i and j through their corresponding rows in $\mathbf{Z}$ – indicating them as $\mathbf{z}_i$ and $\mathbf{z}_j$ – then our feature-feature matrix can be seen as a function that given these two vectors then returns a real number representing a weight for the pair (i, j). In the special case of $\mathbf{W} = I$ this is the standard inner product $\langle \mathbf{z}_i, \mathbf{z}_j \rangle$; in this case the similarity of those two vectors is just the number of features they share, thus implementing homophily. Instead, for a general $\mathbf{W}$ this similarity is $\langle \mathbf{z}_i, \mathbf{z}_j \rangle_{\mathbf{W}}$ (although $\mathbf{W}$ is not necessarily symmetric or positive definite). In this sense $\mathbf{W}$ can be seen as a function $W : 2^F \times 2^F \to \mathbb{R}$, that acts as a *kernel* for sets of features. We will embrace this perspective for some algorithm in the second part of this work.

[1] We will use the notation

$$\chi_I(x) = \begin{cases} 1 & \text{if } x \in I, \\ 0 & \text{if } x \notin I. \end{cases}$$

3.2.3 Intrinsic dimensionality and explainability

Fixing a specific graph G, it is obvious that the adherence of this model to it heavily depends on Z; that is, on the choice of the features that we associate with every node, and ultimately on the set of features F we choose.

Different choices for F will bring the graph from being perfectly explained by these features to not being explained at all. We could then say that the *explainability* is a property of a set of feature F for a certain graph G. We will measure it in practice in many scenarios in the third, experimental, part of this work.

For the moment, let us point out that the cardinality $|F|$ can be seen as an *intrinsic dimensionality* of the graph G: if the graph could be explained by our model without any error at all, then the same information of G could be contained by $\mathbf{Z}$ and $\mathbf{W}$. In that case, we could say that the out-links of node i – described in the graph by ℓ_i, the i-th row of the adjacency matrix $\mathbf{L}$ – could be equally represented by $\mathbf{z}_i$, thus with a much smaller dimension: specifically, with a vector of $|F|$ elements.

In fact, $|V|$ is a natural upper bound for $|F|$. If we set $F = V$ and associate to each node $i \in V$ only a feature $i \in F$ – so that $Z_{i,\hat{i}} = 1$ iff $i = \hat{i}$ and $\mathbf{Z} = I$ – then the graph will be always perfectly explained by these features: it would be enough to set ϕ as the step function $\chi_{(0,\infty)}$ and $\mathbf{W} = \mathbf{L}$ to make the results of our model identical to the graph, since

$$\mathbb{P}\Big((i,j) \in L\Big) = \phi\bigg(\sum_{\hat{i}}\sum_{\hat{j}} Z_{i,\hat{i}} W_{\hat{i},\hat{j}} Z_{j,\hat{j}}\bigg) = \phi(W_{i,j}) = \begin{cases} 1 \text{ if } (i,j) \in L \\ 0 \text{ otherwise} \end{cases}$$

Naturally this choice of F is not very useful; in practice, we obviously want $F \ll V$. For this, we allow for the introduction of some degree of approximation; some links will be falsely predicted by our model, because it will expect their categories to link each other. We shall call this effect our *generalization error*. We will see in experiments how it can be measured and how it is intimately connected with the explainability of a set of features in a graph.

3.2.4 Adding normalization

Let us now present some interesting variations of the proposed model. In many real scenarios, we can speculate that not all features are created equal. For example, in the forming of a friendship link between two people, discovering that they both have seen *"Gone with the Wind"*, the highest-grossing film ever made[2], may not give us much information; knowing instead that they both have seen the underground cult movie by Kevin Smith *"Clerks"* could give to their friendship link a more solid background. Similarly, in a network of economic exchanges, a computer manufacturer will form a more solid link with one of the few agents which develop operating systems, than with a specific screw factory out of all in the world. In other words, for some cases rarest features matter more.

[2]Adjusted for inflation.

Column normalization. Translating this to our model realm, what we want is that the impact of a feature k in the establishing of a link to be inversely weighted by the number of nodes with that feature—that is $\sum_l Z_{l,k}$. To implement this effect, we just need to use the row-normalized version of $\mathbf{Z}$ in our equation:

$$\overleftarrow{Z}_{i,h} = \frac{Z_{i,h}}{\sum_l Z_{l,k}}$$

where we used the notation $\overleftarrow{\mathbf{Z}}$ since it is a left stochastic matrix (i.e., column-normalized). Each column can be seen as a probability distribution among nodes, uniform on nodes having that feature and zero on the others. If we employ $\overleftarrow{\mathbf{Z}}$ in place of $\mathbf{Z}$ in (3.1), we obtain

$$\mathbb{P}\Big((i,j) \in L\Big) = \phi\Big(\sum_h \sum_k \overleftarrow{Z}_{i,h} W_{h,k} \overleftarrow{Z}_{j,k}\Big) = \phi\Big(\sum_h \sum_k \frac{Z_{i,h} W_{h,k} Z_{j,k}}{(\sum_l Z_{l,h})(\sum_l Z_{l,k})}\Big) \tag{3.4}$$

thus reaching the effect we wanted: inside the summation, rare features will bear more weight, and common features will be of lesser importance. This can also be seen as an adaptation of a tf-idf-like schema [191] to our context, where nodes are documents and features are words: connecting documents containing the word *"avuncologratulazione"* is more certain than connecting documents containing the word *"the"*.

Row normalization. It comes natural to ask what happens when we normalize by row instead of by column; that is, by using

$$\overrightarrow{Z}_{i,h} = \frac{Z_{i,h}}{\sum_q Z_{i,q}}$$

Here we used the notation $\overrightarrow{\mathbf{Z}}$ since it is a right stochastic matrix. This time, each row defines a distribution, uniform on the features possessed by that node. By substituting $\overrightarrow{Z}$ in our main equation, we get:

$$\mathbb{P}\Big((i,j) \in L\Big) = \phi\Big(\sum_h \sum_k \overrightarrow{Z}_{i,h} W_{h,k} \overrightarrow{Z}_{j,k}\Big) = \phi\Big(\frac{\sum_h \sum_k Z_{i,h} W_{h,k} Z_{j,k}}{(\sum_q Z_{i,q})(\sum_q Z_{j,q})}\Big) \tag{3.5}$$

Thus, the effect we get is that a node with a high number of features will require a higher sum to form a link. If two pairs of nodes have the same value for $\sum_h \sum_k Z_{i,h} W_{h,k} Z_{j,k}$, the pair with less features will be more likely to form a link than the other one. This behavior is natural in many contexts: a higher number of features will set a higher bar for the sum of their effects; in other words, nodes belonging to few categories provide stronger signals than those that belong to many categories.

ℓ^2-norm. In both cases, we can also use the ℓ^2-norm instead of the ℓ^1 we just presented, to obtain a smoother effect. In the row-normalization case, e.g, nodes with many features will have lower weight, but it would be slightly higher than in the previous ℓ^1 case, since the normalization

would be[3]:

$$\overrightarrow{\overrightarrow{Z}}_{i,h} = \frac{Z_{i,h}}{\sqrt{\sum_q Z_{i,q}}}$$

This normalization is in fact a compromise between the non-normalized case $\mathbf{Z}$ and the previous normalization $\overrightarrow{\mathbf{Z}}$, since their normalization factors are in the order $1 \leq \sqrt{\sum_q Z_{i,q}} \leq \sum_q Z_{i,q}$. Our model in this case is almost the same:

$$\mathbb{P}\Big((i,j) \in L\Big) = \phi\Big(\sum_h \sum_k \overrightarrow{\overrightarrow{Z}}_{i,h} W_{h,k} \overrightarrow{\overrightarrow{Z}}_{j,k}\Big) = \phi\Big(\frac{\sum_h \sum_k Z_{i,h} W_{h,k} Z_{j,k}}{\sqrt{(\sum_q Z_{i,q})(\sum_q Z_{j,q})}}\Big) \tag{3.6}$$

We will employ it in practice in some learning algorithms presented in the second part of this work.

In principle, there are many other forms of normalization that are applicable to our model. In fact, every well-defined norm could lead to a well-defined model. Beside the most obvious choices of $2, 3, \ldots$-norm or max-norm, we wish to highlight that also the normalization proposed by Adamic and Adar [1] for link prediction, that in our case would be

$$\widehat{Zi,h} = \frac{Z_{i,h}}{\log \sum_q Z_{i,q}}$$

could be of interest in practical applications of our model. This could constitute an interesting approach, and it is left as future work.

3.3 Hybrid graph models

Let us conclude this chapter by presenting some hybrid models. The idea is to crossbreed our model with some small world models already known from literature; specifically, we will look at the two models presented in section 2.2: the Barabási-Albert model [17] and the Jackson-Rogers model [100]. The idea of hybridizing a small world model with a feature-based one is not new: as we mentioned in section 2.3, one of the first hidden variables model, the one designed by Bianconi and Barabási [24], was itself grafting hidden features (in that case, a single real-valued number for each node) into the BA model.

We will test how our own hybrid models behave in simulation, with respect to our main model, in section 4.5.

3.3.1 Feature-feature Barabási-Albert model

Our first variant of the feature-feature probability model takes into account the fact that some edges exist independently of the features that the involved nodes exhibit, but they are there simply because of the popularity of a node, as in the traditional "preferential attachment" model

[3]Please note that $Z_{i,q} = {Z_{i,q}}^2$ since $Z_{i,q} \in \{0, 1\}$

by Barabási and Albert [17]. To take this into consideration, instead of using (3.1), we define (for $1 \leq j < i \leq n$):

$$\mathbb{P}(G_{i,j} = 1) = \delta\phi\left(\sum_{h,k} Z_{i,h} W_{h,k} Z_{j,k}\right) + (1-\delta)\frac{d_j(i-1)}{2D(i-1)}, \tag{3.7}$$

where $d_j(k)$ and $D(k)$ are, respectively, the degree of node j and the overall number of edges just after node k was added. The parameter δ controls the mixture between the pure feature-feature model and the preferential-attachment model (degenerating to the former when $\delta = 1$, and to the latter when $\delta = 0$).

3.3.2 Feature-feature Jackson-Rogers model

Finally, Jackson and Rogers [100] observed that preferential attachment can be induced also injecting a "friend-of-friend" approach in the creation of edges. Their behavior can be mimicked in our model as follows: we first generate a graph with adjacency matrix $\mathbf{G}'$ using the pure FF model, as expressed by (3.1).

After this, every node i looks at the set of the neighbors of its neighbors, according to $\mathbf{G}'$. If this set is not empty, it then selects one node from the set uniformly at random; the resulting node is chosen as an "extra" friend of i with some probability $1-\delta$ (for suitably chosen $\delta \in [0,1]$). The adjacency matrix obtained in this way is A. Once more, if $\delta = 0$ we have $\mathbf{G} = \mathbf{G}'$ so we get back to the pure version of our model.

We will analyze those models by simulation in the next chapter.

3.4 Discussion

In this chapter, we outlined the framework in which we will operate throughout this thesis. In particular, we explored and motivated how our complex network model works.

In section 3.1, we motivated the necessity for a model for feature-rich graphs, highlighting that explaining links through features in nodes could be a promising assumption in many contexts. We sketched the behaviors that such a model should implement: homophily (i.e., the presence of the same feature in different nodes can foster links between them), of course, but also heterophily (different features could foster links) and their opposites (some pairs of features could *prevent* links).

Then, we presented our model in section 3.2, by defining its ingredients, dividing them in the observable and latent ones, and by presenting its equation (3.1). This equation essentially connects the probability of a link between two nodes with the sum of all the elements of a (latent) feature-feature matrix relative to the pairs of features possessed by those nodes. To connect this sum and the probability, the model postulates an activation function ϕ, assumed to be monotonically increasing: higher sums will mean higher probabilities, no matter the specific shape of the chosen function.

In section 3.2.1, we analyzed the different aspects of the model: we understood how the latent feature-feature matrix operates, and we described the role of the activation function, as well as

indicating the sigmoid (3.2) as the most common choice for such a function. We then reformulated our model as a pure linear algebra equation in section 3.2.2 and we discussed possible application of this view.

In section 3.2.3 we showed how the number of features in our model for a certain graph can be thought as the intrinsic dimensionality of that graph. In other words, if a correct feature-feature matrix exists, then the features of the nodes become a compact representation of their links.

We discussed then how different normalization heuristics could be implemented naturally in our framework: in section 3.2.4, we saw how both node-based normalization (nodes with fewer features have stronger effects) and feature-based normalization (rare features matter the most) can be though as a normalization layer on the node-feature matrix. Also, different norms can be used in practice to implement this behavior.

Finally, we also described how our model can be also be crossbred with previous small-world models, in order to produce different complex network models that incorporate both previously known dynamics with a feature-rich aspect.

After having outlined how our model can trace a link between features and graphs, in the next chapter we will focus on how can we model the features themselves, and how it affects the resulting network.

Chapter 4

Modeling the evolution of features

4.1 Introduction

In this chapter, we will detail how we can model the creation of the node-feature matrix $\mathbf{Z}$ and how the proposed model for $\mathbf{Z}$ affects the creation of the graph G.

This model, inspired by a generalization of the Indian buffet process, is simple and contains a small number of parameters, with a clear and intuitive role. Each node is characterized by a number of features and the probability of the existence of an edge between two nodes depends on the features they have; the number of possible features is not fixed a priori and can grow indefinitely. Moreover, a random *fitness parameter* is introduced for each node in order to determine its ability to transmit its own features to other nodes; this behavior is added for the first time on top of a process of Indian-Buffet type. Because of the fitness property, a node's connectivity does not depend on its age alone, so that also "young but fit" nodes are able to compete and succeed in propagating their features and acquiring links. We also show how, considering the resulting bipartite node-feature matrix, it is possible to gain some insight about which nodes were originally the most "fit".

Our model for the evolution of features depends on few parameters, that are characterized by their straightforward interpretation and by the availability of proper estimators. We provide some theoretical as well as experimental results regarding the power-law behavior of the model and the proposed tools for the estimation of the parameters. We also show, through a number of experiments, how the proposed model naturally captures most local and global properties (e.g., degree distributions, connectivity and distance distributions) real networks exhibit.

The rest of the chapter is organized as follows. In section 4.2 we will review relevant literature. In section 4.3 we will present a statistical model for the generation of the node-feature matrix $\mathbf{Z}$ from a small set of parameters, for which we will provide theoretical results, estimators, and analysis. These methods are then tested by simulations in Section 4.4. The properties of the generated graphs are studied experimentally in section 4.5. In section 4.6 we will discuss how to rank nodes from the influence they had on the evolution of features, within our model. In section 4.7, finally, we will discuss the work presented in this chapter, with respect to the state of the art.

The contents of this chapter were published in the journal *"Information Sciences"* [30] (with minor differences in notation and form). Some parts were expanded, after further investigation on this matter.

4.2 Related works

As we have seen in chapter 2, the task of modeling a network is often undertaken directly, but recently some authors proposed to split it into two steps [121,122]. This proposal stems from the observation that many complex networks contain two types of entities: actors on one hand, and groups (or features) on the other; every actor belongs to one, or more, groups (or can exhibit one, or more, features), and the common membership to groups (or the sharing of features) determines a relation between actors. Among the many different families of feature-rich network models we described, we found the one proposed by Miller, Griffiths and Jordan [136] to be the most useful for describing large scale real data, and on that we developed our framework, described in chapter 3.

A natural model for the evolution of such binary bipartite graphs comes from Bayesian statistics and it is known as the Indian Buffet process, introduced by Griffiths *et al.* [10, 80, 86] and, subsequently extended and studied by many authors [23,37,181,182]. The process defines a plausible way for features to evolve, always according to a *rich-get-richer* principle: because of this, it represents a promising model for affiliation networks. Since the Indian Buffet process provides *a prior distribution* in Bayesian statistics, these models have been used to reconstruct affiliation networks, with an unknown number of features, from data where only friendship relations between actors are available [136].

In this paper we propose and analyze a model that combines two characteristic characterizing the evolution of a network: first, behind the adjacency matrix of a network there is a *latent attribute structure* of the nodes; second, not all nodes are equally successful in transmitting their own attributes to the new nodes, but each node n is characterized by a *random fitness parameter* describing its ability to transmit the node's attributes. We refer to this aspect as *competition*.

We were inspired by the recent generalization of the Indian buffet process [23]. However, the model presented here is in some sense simpler since the parameters (that will be introduced and analyzed in the next sections) play a role that is clearer and more intuitive. Specifically, we have two parameters (α and β) that control the number of new attributes each new node exhibits (in particular $\beta > 0$ tunes the power-law behavior of the overall number of different observed attributes), whereas the random fitness parameters R_i impact on the probability of the new nodes to inherit the attributes of the previous nodes. With respect to previous models [23], we lose some mathematical properties, but we will show that some important results still hold true and they allow us to estimate the parameters and, in particular, the exponent of the power-law behavior.

Regarding the use of fitness parameters, we recall the work by Bianconi and Barabási [24] that introduced some fitness parameters describing the ability of the nodes to compete for links. In this work, we integrate this behavior into a feature-based model. While in previous work [24] the fitness parameters appear explicitly in the edge-probabilities, in our model they affect the

evolution of the attribute matrix and then play an implicit role in the evolution of the connections.

4.3 A model for feature generation

Let us assume that the nodes enter the network sequentially so that node i represents the node that comes into the network at time i. Let $\mathcal{X}$ be an unbounded collection of possible features that a node can exhibit. (This means that we do not specify the total number of possible features *a priori*.) Each node is assumed to have only a finite number of features.

Let us represent $\mathbf{Z}$ as the set of its rows $\mathbf{z}_1, \dots, \mathbf{z}_n$; specifically, $\mathbf{z}_n$ represents the features of the node n: $Z_{i,k} = 1$ if node i has feature k, $Z_{i,k} = 0$ otherwise. We assume that each $\mathbf{z}_i$ remains unchanged in time, in the sense that every node decides its own features when it arrives and then it will never change them thereafter. This assumption is quite natural in many contexts.

In all the sequel of this chapter we postulate that $\mathbf{Z}$ is left-ordered. This means that in the first row the columns for which $Z_{1,k} = 1$ are grouped on the left and so, if the first node has N_1 features, then the columns of $\mathbf{Z}$ with index $k \in \{1, \dots, N_1\}$ represent these features. The second node could have some features in common with the first node (those corresponding to indices k such that $k = 1, \dots, N_1$ and $Z_{2,k} = 1$) and some, say N_2, new features. The latter are grouped on the right of the sets for which $Z_{1,k} = 1$, i.e., the columns of Z with index $k \in \{N_1 + 1, \dots, N_2\}$ represent the new features brought by the second node. This grouping structure persists throughout the matrix Z.

Here is an example of a Z matrix with $n = 4$ nodes; observe that, for every node i, the i-th row contains 1's for all the columns with indices $k \in \{N_1 + \cdots + N_{i-1} + 1, \dots, N_1 + \cdots + N_i\}$ (they represent the new features brought by i); moreover some elements of the columns with indices $k \in \{1, \dots, N_1 + \cdots + N_{i-1}\}$ are also 1's (features that were brought by previous nodes and that also node i decided to adopt):

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Explanation of the model. We will describe the dynamics using a culinary metaphor (similarly to what some authors do for other models, see Chinese Restaurant [153], Indian Buffet process [10, 86, 181] and their generalizations [19, 23]). We identify the nodes with the customers of a restaurant and the features with the dishes, so that the dishes tried by a customer represent the features that a node exhibits; therefore, in the matrix above $Z_{i,j} = 1$ means that customer i tried dish j. The general idea in this metaphor is that a customer will try both some *new* dishes (precisely, N_i new dishes), but also some dishes already tasted by previous customers. More popular dishes will have larger probability to be tasted again, with the proviso that, in establishing the "popularity" of a dish, some people are naturally better influencers than others.

Let us explain the model formally, through this metaphor. Fix $\alpha > 0$ and $\beta \in (-\infty, 1]$. Also, let $\mathrm{Poi}(\lambda)$ denote the Poisson distribution with mean $\lambda > 0$. Customer (node) n is attached

a random weight (that we call, in accordance with the usage in network science [23], *fitness parameter*) r_n. We assume that each r_n is independent of $r_1, \dots, r_{n-1}$ and of the dishes (features) experimented by customers $1, \dots, n$. The fitness parameter r_n affects the choices of the future customers (those after n), while the choices of customer n are affected by the fitness parameters and the choices of the previous ones. Indeed, it may be the case that different customers have different relevance, due to some random cause, that does not affect their choices but is relevant to the choices of future customers (i.e., their capacity of being followed).

The dynamics is as follows. Customer (node) 1 tries N_1 dishes (features), where N_1 is Poi(α)-distributed. For a customer $n+1$ after the first one, we have that:

- They select a number of dishes already tried by others. The inclusion probability of dish k is

$$P_n(k) = \frac{\sum_{i=1}^{n} r_i Z_{i,k}}{\sum_{i=1}^{n} r_i}\,, \tag{4.1}$$

  where $Z_{i,k} = 1$ if {customer i has selected dish k} and $Z_{i,k} = 0$ otherwise. It is a preferential attachment rule: the larger the weight of a dish k at time n (given by the numerator of (4.1), i.e., the total value of the random variables r_i associated to the customers that have chosen it until time n), the greater the probability that it will be chosen by the future customer $n+1$.

  Please note that we can further generalize our model by introducing another parameter $c \geq 0$ in the inclusion probabilities so that

$$P_n(k) = \frac{\sum_{i=1}^{n} r_i Z_{i,k}}{c + \sum_{i=1}^{n} r_i}\,,$$

  In other words, the bigger c, the smaller the inclusion probabilities and so the sparser the features. This can allow to obtain feature matrices $\mathbf{Z}$ that are sparser. For the moment – and through the rest of this chapter – we set $c = 0$.

- In addition to the dishes already tried by others, customer $n+1$ also tries N_{n+1} new dishes, where N_{n+1} is Poi(Λ_n)-distributed with

$$\Lambda_n = \frac{\alpha}{\left(\sum_{i=1}^{n} r_i\right)^{1-\beta}}. \tag{4.2}$$

Note that we assume that, for each single customer, each choice is made independently from the others. Besides this assumption of independence, we also assume that the random parameters r_n are identically distributed with $r_n \geq v$ for each n and a certain number $v > 0$, and $E[r_n^2] < +\infty$. The particular distributions for which $E[r_n] = 1$ allow for a simplification of the following formulas; however, we will not require this assumption for generality.

Role of the parameters. We set $E[r_n] = m_r$ – i.e., the expected values for the fitness parameters; then, let us define $L_n = \sum_{i=1}^{n} N_i$, i.e. the overall number of different observed features for the first n nodes (i.e., customers).

The meaning of the parameters is the following. The random fitness parameters r_n fundamentally control the probability of transmitting the features to the new nodes. The main effect of β is that it regulates the asymptotic behavior of the random variable L_n (see Theorem 1). In particular, $\beta > 0$ is the power-law exponent of L_n. The main effect of α is the following: the larger α, the larger the total number of new tried dishes by a customer. It is worth to note that β fits the asymptotic behavior of L_n (in particular, the power-law exponent of L_n) and, separately, α fits the number of observed features.

Choice of the Poisson distribution. The Poisson distribution is common in modeling counts in stochastic processes [109, 156]; examples include the number of decays in radioactive substance, or the number of mutations in in a strand of DNA of given length. In a number of circumstances, this is the case also for human processes: a well-known example is phone calls arriving in a call center [156], but recent results include also scores in sports [108].

In the case of Indian Buffet Processes, the choice is also motivated by a particular ease in mathematical analysis of the Poisson distribution [10]. It is in fact characterized by a range of convenient mathematical properties, which makes it particularly suited for modeling the building blocks of the process (e.g., when dealing with sum of Poisson random variables). However, other authors [103] have tried to generalize the Indian Buffet Process to cases where the number of new dishes is not Poisson-distributed, but follows—for example—a negative binomial.

In its study of models of human dynamics, Barabási [16] pointed out that in a number of cases, human processes are better modeled by a power-law distribution than by the classic Poisson distribution; one of his main examples is the time distribution of the emails sent by a given user. His point is that, while the underlying choices (e.g., in the case of emails, the priority of the single email) may be distributed in a non-power-law fashion, the *emerging behavior* will be, nonetheless, a power-law, as a consequence of how these choices are carried out as a whole.

As we will see in the next sections, this is also what happens with the Indian Buffet Process: while the single choices—the number of new dishes to try—are not distributed with a power-law (but instead with a Poisson distribution), the resulting total number of features (see section 4.3.1, or fig. 4.2), or the number of nodes per feature (see experiments in fig. 7.4), ends up following a power-law distribution. Previous works [181] have shown that, also in the Indian Buffet Process, the power-law is an emerging properties of the system, and it can explain the power-law distribution of features in real phenomena.

It is, however, intriguing to ask ourselves what would happen if we would employ a power-law to model, from the beginning, the number of new features per node. For sure, the model would be dominated by a small number of nodes. In fact, what would happen is that a few nodes would introduce a very large number of new features. In particular, since the sum of two power laws with the same exponent tends towards a power-law with the same exponent [190], we can see that in the end the total number of features will be again distributed as a power-law, but in that case almost all of the appearing features will be introduced by a small number of nodes.

Since these nodes will have a very large number of features, the computational aspect of such a model would be more difficult to handle. However, it would have the effect of introducing *implicitly* the competition dynamics that we have introduced *explicitly* with the fitness values $r_1, \ldots, r_n$. Whether it would lead to a better model to fit empirical data is a matter that needs more investigation.

4.3.1 Theoretical results about estimation of α and β

In this section we prove some properties regarding the asymptotic behavior of L_n. In particular, the first result shows a logarithmic behavior for $\beta = 0$ and a power-law behavior for $\beta \in (0, 1]$. These results allow us to define suitable estimators for β and α.

The mathematical formalization of the above model can be performed by means of random measures [114] with atoms corresponding to the tried dishes (observed features), similarly to previous works [23,37,182]. More precisely, besides the sequence of positive real random variables (r_n), we can define a sequence of random measures (M_n), such that each M_{n+1} is, conditionally on the past $(M_i, r_i : i \leq n)$, a Bernoulli random measure with a hazard measure ν_n, having a discrete part related to the features k belonging to a node and their weights $P_n(k)$ and a diffuse part with total mass equal to Λ_n.

Theorem 1. *Using the previous notation, the following statements hold true:*

- *a)* $\sup_n L_n = L < +\infty$ *a.s. for* $\beta < 0$*;*
- *b)* $L_n/\ln(n) \xrightarrow{a.s.} \alpha/m_r$ *for* $\beta = 0$*;*
- *c)* $L_n/n^\beta \xrightarrow{a.s.} \alpha/(\beta\, m_r^{1-\beta})$ *for* $\beta \in (0, 1]$.

Proof. Let us prove assertion a), first. Let $\mathcal{F}_i$ be the natural σ-field associated to the model until time i and set $\Lambda_0 = \alpha$. Since, conditionally on $\mathcal{F}_i$, the distribution of N_{i+1} is $\mathrm{Poi}(\Lambda_i)$, we have

$$P(N_{i+1} \geq 1) = E\big[P(N_{i+1} \geq 1 \mid \mathcal{F}_i)\big] \leq E[\Lambda_i].$$

Since $r_i \geq v > 0$, we obtain

$$\sum_i P(N_{i+1} \geq 1) \leq \alpha \sum_i \frac{1}{(vi)^{(1-\beta)}} < +\infty$$

(where the convergence of the series is due to the assumption $\beta < 0$). By the Borel-Cantelli lemma, we conclude that

$$P\big(N_i > 0 \text{ infinitely often }\big) = P\big(N_i \geq 1 \text{ infinitely often}\big) = 0.$$

Hence, if $\beta < 0$, there is a random index N such that $L_n = L_N$ a.s. for all $n \geq N$, which concludes the proof of a).

The assertion c) is trivial for $\beta = 1$ since, in this case, L_n is the sum of n independent random variables with distribution $\mathcal{P}(\alpha)$ and so, by the classical strong law of large numbers, $L_n/n \xrightarrow{a.s.} \alpha$.

Now, let us prove assertions b) and c) for $\beta \in [0,1)$. Define

$$\lambda(\beta) = \frac{\alpha}{m_R} \text{ if } \beta = 0 \quad \text{and} \quad \lambda(\beta) = \frac{\alpha}{\beta\, m_R^{1-\beta}} \text{ if } \beta \in (0,1),$$
$$a_n(\beta) = \log n \text{ if } \beta = 0 \quad \text{and} \quad a_n(\beta) = n^\beta \text{ if } \beta \in (0,1).$$

We need to prove that

$$\frac{L_n}{a_n(\beta)} \xrightarrow{a.s.} \lambda(\beta).$$

First, we observe that we can write

$$\frac{\sum_{i=1}^{n-1} \Lambda_i}{a_n(\beta)} = \alpha \frac{\sum_{i=1}^{n-1} i^{\beta-1} \left(\overline{R}_i\right)^{\beta-1}}{a_n(\beta)},$$

where, by the strong law of the large numbers,

$$\overline{r}_i = \frac{\sum_{j=1}^{i} r_j}{i} \xrightarrow{a.s} m_r.$$

Therefore, since $\sum_{i=1}^{n-1} i^{\beta-1}/a_n(\beta)$ converges to 1 when $\beta = 0$ and to $1/\beta$ when $\beta \in (0,1)$, we get

$$\frac{\sum_{i=1}^{n-1} \Lambda_i}{a_n(\beta)} \xrightarrow{a.s.} \lambda(\beta). \tag{4.3}$$

Next, let us define

$$T_0 = 0 \quad \text{and} \quad T_n = \sum_{i=1}^{n} \frac{N_i - E[N_i \mid \mathcal{F}_{i-1}]}{a_i(\beta)} = \sum_{i=1}^{n} \frac{N_i - \Lambda_{i-1}}{a_i(\beta)}.$$

Then, (T_n) is a martingale with respect to $(\mathcal{F}_n)$ and

$$E[T_n^2] = \sum_{i=1}^{n} \frac{E\left[(N_i - \Lambda_{i-1})^2\right]}{a_i(\beta)^2} = \sum_{i=1}^{n} \frac{E\left\{E\left[(N_i - \Lambda_{i-1})^2 \mid \mathcal{F}_{i-1}\right]\right\}}{a_i(\beta)^2} = \sum_{i=1}^{n} \frac{E[\Lambda_{i-1}]}{a_i(\beta)^2}.$$

Since $r_i \geq v > 0$, it is easy to verify that $E[\Lambda_i] = \mathrm{O}(i^{-(1-\beta)})$ and so $\sup_n E[T_n^2] = \sum_{i=1}^{\infty} \frac{E[\Lambda_{i-1}]}{a_i(\beta)^2} < \infty$. Thus, (T_n) converges a.s., and the Kronecker's lemma implies

$$\frac{1}{a_n(\beta)} \sum_{i=1}^{n} a_i(\beta) \frac{(N_i - \Lambda_{i-1})}{a_i(\beta)} \xrightarrow{a.s.} 0,$$

so finally

$$\lim_n \frac{L_n}{a_n(\beta)} = \lim_n \frac{\sum_{i=1}^{n} N_i}{a_n(\beta)} = \lim_n \frac{\sum_{i=1}^{n} \Lambda_{i-1}}{a_n(\beta)} = \lim_n \frac{\Lambda_0 + \sum_{i=1}^{n-1} \Lambda_i}{a_n(\beta)} = \lambda(\beta) \quad \text{a.s.} \tag{4.4}$$

□

The above result entails that $\ln(L_n)/\ln(n)$ is a strongly consistent estimator of $\beta \in [0,1]$. In fact:

- if $\beta = 0$ then $L_n \overset{a.s.}{\sim} \frac{\alpha}{m_r}\ln(n)$ as $n \to +\infty$; hence $\ln(L_n) \overset{a.s.}{\sim} \ln(\alpha/m_r) + \ln(\ln(n))$, therefore $\ln(L_n)/\ln(n) \overset{a.s.}{\sim} \ln(\alpha/m_r)/\ln(n) + \ln(\ln(n))/\ln(n) \overset{a.s.}{\to} 0 = \beta$;
- if $\beta > 0$, we have $L_n \overset{a.s.}{\sim} \lambda(\beta) n^\beta$ as $n \to +\infty$ so $\ln(L_n) \overset{a.s.}{\sim} \ln(\lambda(\beta)) + \beta \ln(n)$, hence $\ln(L_n)/\ln(n) \overset{a.s.}{\sim} \ln(\lambda(\beta))/\ln(n) + \beta \overset{a.s.}{\to} \beta$.

Remark 4.3.1. In practice, the value of $\ln(L_n)/\ln(n)$ may be quite far from the limit value β when n is small. Hence, it may be worth trying to fit the power-law dependence of L_n as a function of n with standard techniques [53] and use the slope $\widehat{\beta}_n$ of the regression line in the log-log plot as an effective estimator for β.

Finally, assuming that $\beta \in [0,1]$ and m_r are known, we can get a strongly consistent estimator of α, as:

$$m_r \frac{L_n}{\ln(n)} \quad \text{for } \beta = 0 \qquad \text{and} \qquad m_r^{1-\beta}\beta \frac{L_n}{n^\beta} \quad \text{for } 0 < \beta \leq 1.$$

In practice, we assume β equal to the estimated value $\widehat{\beta}_n$ (as defined before) and we take m_r equal to the estimated value $\overline{r}_n = \sum_{i=1}^n r_i/n$, if the random parameters r_i are known. In Section 4.6.3, we will discuss the case when the random variables r_i are unknown.

Remark 4.3.2. Once more, it may be better in practice to estimate α as

$$\begin{aligned} \widehat{\alpha}_n &= m_r\, \widehat{\gamma}_n \qquad \text{when } \beta = 0 \\ \widehat{\alpha}_n &= \beta\, m_r^{1-\beta}\, \widehat{\gamma}_n \qquad \text{when } 0 < \beta \leq 1, \end{aligned} \tag{4.5}$$

where $\widehat{\gamma}_n$ is the slope of the regression line in the plot $\big(\ln(n), L_n\big)$ or in the plot $\big(n^\beta, L_n\big)$ according to whether $\beta = 0$ or $\beta \in (0,1]$.

We complete this section with a central-limit theorem that gives the rate of convergence of $L_n/a_n(\beta)$ to $\lambda(\beta)$ when $\beta \in [0,1]$.

Theorem 2. *If $\beta \in [0,1]$, then we have the following convergence in distribution*[1]*:*

$$\sqrt{a_n(\beta)}\left\{\frac{L_n}{a_n(\beta)} - \lambda(\beta)\right\} \xrightarrow{d} \mathcal{N}\big(0,\, \lambda(\beta)\big).$$

Proof. The result for $\beta = 1$ follows from the classical central limit theorem, since, in this case, L_n is the sum of n independent random variables with distribution $\mathcal{P}(\alpha)$. Assume now $\beta \in [0,1)$ and set $\Lambda_0 = \alpha$. We first prove that

$$\sqrt{a_n(\beta)}\left\{\frac{\sum_{i=1}^n \Lambda_{i-1}}{a_n(\beta)} - \lambda(\beta)\right\} \xrightarrow{P} 0. \tag{4.6}$$

[1] Actually, the convergence is in the sense of the *stable* convergence, which is stronger than the convergence in distribution. Indeed, stable convergence is a form of convergence intermediate between convergence in distribution and convergence in probability.

By some calculations, condition (4.6) is equivalent to

$$\frac{\sum_{i=1}^{n-1}\left\{\left(\sum_{j=1}^{i} r_j\right)^{\beta-1} - (m_r\, i)^{\beta-1}\right\}}{\sqrt{a_n(\beta)}} \xrightarrow{P} 0. \tag{4.7}$$

Since $r_j \geq v > 0$, we have $m_r \geq v > 0$ and we obtain

$$\begin{aligned} E\left[\left|(m_r\, i)^{\beta-1} - \left(\sum_{j=1}^{i} r_j\right)^{\beta-1}\right|\right] &\leq \frac{E\left[\left|\left(\sum_{j=1}^{i} r_j\right)^{1-\beta} - (m_r\, i)^{1-\beta}\right|\right]}{(v\, i)^{2(1-\beta)}} \\ &\leq \frac{1}{(v\, i)^{2(1-\beta)}}\,\frac{1-\beta}{(v\, i)^{\beta}}\, E\left[\left|\sum_{j=1}^{i} r_j - m_r\, i\right|\right] \\ &= \frac{1-\beta}{v^{2-\beta}}\,\frac{1}{i^{1-\beta}} E\left[\,|\overline{r}_i - m_r|\,\right] \\ &\leq \frac{1-\beta}{v^{2-\beta}}\,\frac{1}{i^{1-\beta}}\sqrt{Var[\overline{r}_i]} = \frac{(1-\beta)\sqrt{Var[r_1]}}{v^{2-\beta}}\,\frac{i^{\beta-1}}{\sqrt{i}}. \end{aligned}$$

This proves condition (4.7) (and so (4.6)). Indeed, we have

$$\begin{aligned} &\frac{1}{\sqrt{a_n(\beta)}} E\left[\left|\sum_{i=1}^{n-1}\left\{\left(\sum_{j=1}^{i} r_j\right)^{\beta-1} - (m_r\, i)^{\beta-1}\right\}\right|\right] \leq \\ &\frac{1}{\sqrt{a_n(\beta)}}\sum_{i=1}^{n-1} E\left[\left|(m_r\, i)^{\beta-1} - \left(\sum_{j=1}^{i} r_j\right)^{\beta-1}\right|\right] \leq \\ &\frac{(1-\beta)\sqrt{Var[r_1]}}{v^{2-\beta}}\,\frac{1}{\sqrt{a_n(\beta)}}\sum_{i=1}^{n-1}\frac{1}{i^{1-(\beta-1/2)}} \to 0. \end{aligned}$$

Next, define

$$T_n = \sqrt{a_n(\beta)}\left\{\frac{L_n}{a_n(\beta)} - \frac{\sum_{i=1}^{n}\Lambda_{i-1}}{a_n(\beta)}\right\} = \frac{\sum_{i=1}^{n}(N_i - \Lambda_{i-1})}{\sqrt{a_n(\beta)}}.$$

In view of (4.6), it suffices to show that $T_n \xrightarrow{d} \mathcal{N}\big(0,\, \lambda(\beta)\big)$.
To this end, for $n \geq 1$ and $i = 1, \ldots, n$, define

$$T_{n,i} = \frac{N_i - \Lambda_{i-1}}{\sqrt{a_n(\beta)}}, \quad \mathcal{G}_{n,0} = \mathcal{F}_0 \quad \text{and} \quad \mathcal{G}_{n,i} = \mathcal{F}_i,$$

where $\mathcal{F}_i$ is the natural σ-field associated to the model until time i. Then, we have $E[T_{n,i} \mid \mathcal{G}_{n,i-1}] = 0$, $\mathcal{G}_{n,i} \subseteq \mathcal{G}_{n+1,i}$ and $T_n = \sum_{i=1}^{n} T_{n,i}$. Thus, by a martingale central limit theorem [88], $T_n \xrightarrow{d} \mathcal{N}\big(0,\, \lambda(\beta)\big)$ provided

$$\text{(i)}\ \sum_{i=1}^{n} T_{n,i}^2 \xrightarrow{P} \lambda(\beta), \quad \text{(ii)}\ \max_{1\leq i\leq n} |T_{n,i}| \xrightarrow{P} 0, \quad \text{(iii)}\ \sup_n E\left[\max_{1\leq i\leq n} T_{n,i}^2\right] < \infty.$$

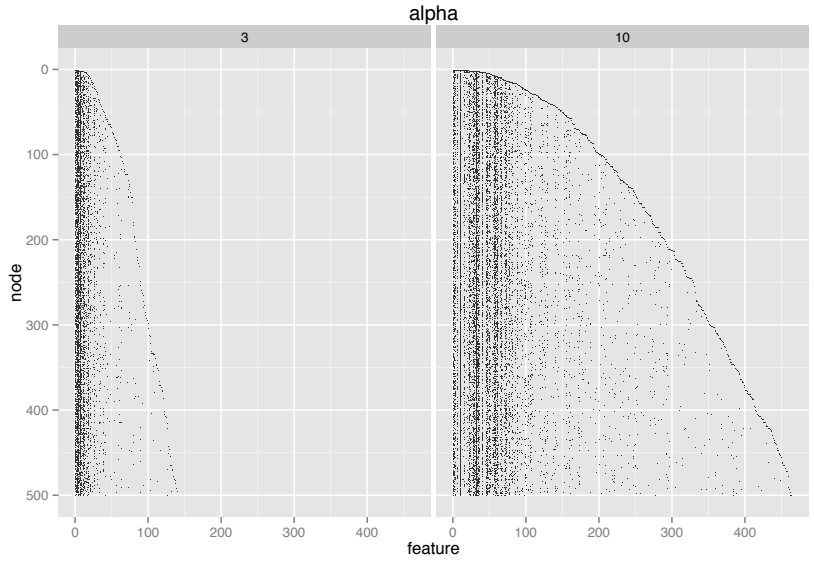


Figure 4.1: The Z matrix for $n = 500$, two different values of α ($\alpha = 3$ and $\alpha = 10$) and a fixed $\beta = 0.5$. The random variables r_i are uniformly distributed on the interval $[0.25, 1.75]$.

Let

$$D_i = (N_i - \Lambda_{i-1})^2 \quad \text{and} \quad U_n = \frac{\sum_{i=1}^n \{D_i - E[D_i \mid \mathcal{F}_{i-1}]\}}{a_n(\beta)} = \frac{\sum_{i=1}^n (D_i - \Lambda_{i-1})}{a_n(\beta)}.$$

By the same martingale argument used in the proof of the previous theorem and by Kronecker's lemma, $U_n \xrightarrow{a.s.} 0$. Then, by (4.3),

$$\sum_{i=1}^n T_{n,i}^2 = \frac{\sum_{i=1}^n D_i}{a_n(\beta)} = U_n + \frac{\sum_{i=1}^n \Lambda_{i-1}}{a_n(\beta)} \xrightarrow{a.s.} \lambda(\beta).$$

This proves condition (i). As to (ii), fix $k \geq 1$ and note that

$$\max_{1 \leq i \leq n} T_{n,i}^2 \leq \frac{\max_{1 \leq i \leq k} D_i}{a_n(\beta)} + \max_{k < i \leq n} \frac{D_i}{a_i(\beta)} \leq \frac{\max_{1 \leq i \leq k} D_i}{a_n(\beta)} + \sup_{i > k} \frac{D_i}{a_i(\beta)} \quad \text{for } n > k.$$

Hence, $\limsup_n \max_{1 \leq i \leq n} T_{n,i}^2 \leq \limsup_n \frac{D_n}{a_n(\beta)}$ and condition (ii) follows since

$$\frac{D_n}{a_n(\beta)} = \frac{\sum_{i=1}^n D_i}{a_n(\beta)} - \frac{\sum_{i=1}^{n-1} D_i}{a_n(\beta)} \xrightarrow{a.s.} 0.$$

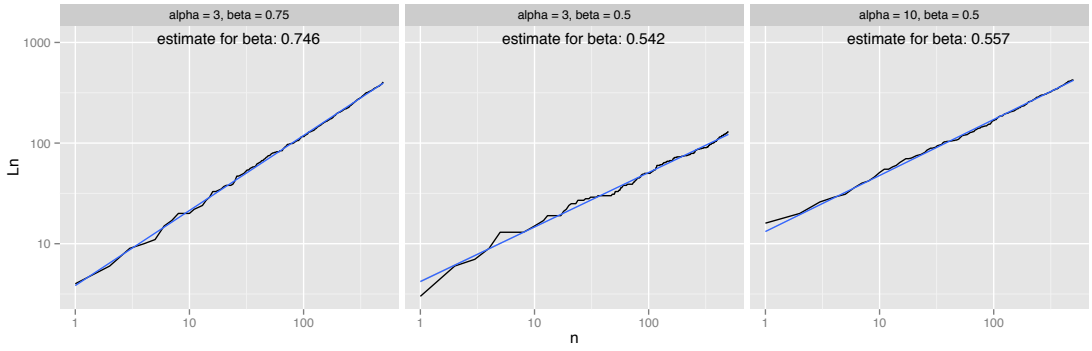


Figure 4.2: Correspondence between the parameter β and the power-law exponent of L_n as a function of n. The estimate of β is the slope of the regression line. Here, we have 500 nodes and the random variables r_i are uniformly distributed on the interval $[0.25, 1.75]$. Values for α and β are indicated above; we can see how different values for α do not affect the power-law behavior.

Finally, condition (iii) is a consequence of

$$E\left[\max_{1\le i\le n} T_{n,i}^2\right] \le \frac{\sum_{i=1}^n E[D_i]}{a_n(\beta)} = \frac{\sum_{i=1}^n E[\Lambda_{i-1}]}{a_n(\beta)} =$$
$$= \frac{\Lambda_0 + \sum_{i=1}^{n-1} E[\Lambda_i]}{a_n(\beta)} \le \frac{\alpha\left(1 + \sum_{i=1}^{n-1}(vi)^{\beta-1}\right)}{a_n(\beta)}.$$

□

4.4 Simulations for the evolution of features

In this section, we shall present a number of simulations we performed in order to illustrate the role of the parameters of the model for Z and also to see how good the proposed tools turn out to be.

4.4.1 Estimating α and β

Firstly, we aim at pointing out the role played by the model parameters α and β. Therefore, we fix a distribution for the random fitness parameters with $E[r_n] = 1$ and we simulate the matrix Z for different values of α and β (fixing one and making the other one change). More precisely, we assume that the random variables r_n are uniformly distributed on the interval $[0.25, 1.75]$.

In Figure 4.1, we visualize the effect of α: a larger α yields a larger number of new features per node.

In Figure 4.2, instead, we visualize how different positive values of β yield a different power-law (asymptotic) behavior of L_n. Indeed, in this figure, we have the log-log plot of L_n as a function of n. In the first two panels, we present two different positive values of β (0.75 and 0.5), showing the correspondence with the power-law exponent of L_n, estimated by the slope of the

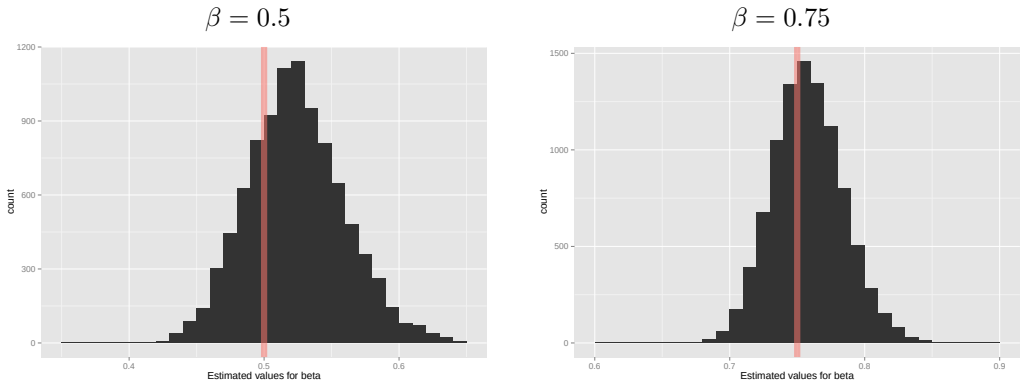


Figure 4.3: Distribution of the estimator $\widehat{\beta}_n$ of β over 10 000 experiments, each with $n = 2000$ and $\alpha = 3$. The random variables r_i are uniformly distributed on the interval $[0.25, 1.75]$. The red line indicates the true value of β.

regression line. Moreover, in the third panel, we point out that the parameter α do not affect the power-law exponent of L_n.

Figure 4.3 underlines that the estimator proposed in remark 4.3.1 works better (i.e. with more precision) for large values of β; in fact, L_n reaches the power-law behavior more quickly for larger values of β. We can see, in fact, that the estimator shows a small bias for $\beta = 0.5$, but the bias becomes negligible for $\beta = 0.75$.

Similarly, we evaluated the estimator $\widehat{\alpha}_n$ of α, obtained by using the slope of the regression line in the plot of L_n as a function of n^β, as said[2] in Remark 4.3.2. Results are illustrated in Figure 4.4.

We also checked how the shape of the matrix Z is influenced by the distribution of the random parameters r_n. More precisely, we analyzed the effect of ε on the shape of Z when the random variables r_i are uniformly distributed on the interval $[\varepsilon, 2 - \varepsilon]$, with $0 < \varepsilon \leq 1$; in this way, for every ε we have that $E[r_i] = 1$ and the variance of r_i is $Var[r_i] = (1 - \varepsilon)^2/3$, which goes to zero as $\varepsilon \to 1$. Hence, when ε is smaller, the variance of the r_n's is larger, so that also a "young" node i have some chance of transmitting their features to the other nodes (recall that a larger r_i makes i more successful in transmitting its own features). This is witnessed (see Figure 4.5) by the number of blackish vertical lines, that are more or less widespread in the whole spectrum of nodes; whereas for larger ε they are more concentrated on the left-hand side (i.e., only the first nodes successfully transmit their features).

4.5 Simulations for the graph structure

The purpose of the following collection of experiments is to determine the topology of a graph generated with the models described above. We will examine through simulations a more restricted case than the general model expressed in section 3.2, by making the following assumptions:

[2]Note that we have $E[r_n] = 1$ and so α coincides with α'

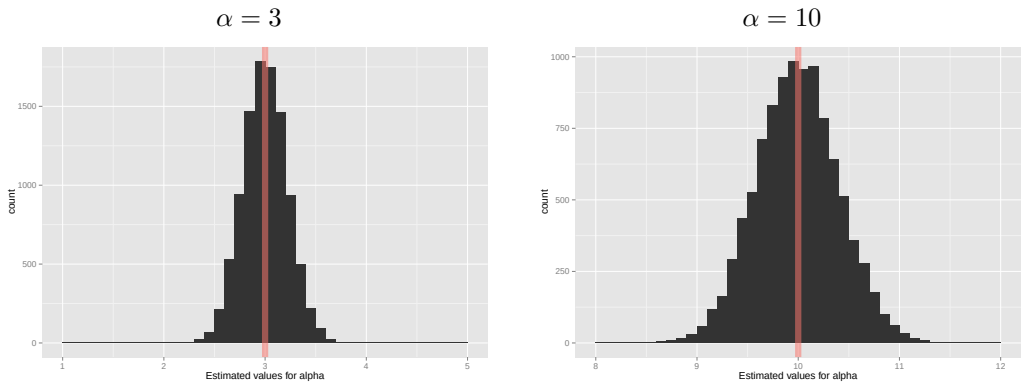


Figure 4.4: Distribution of the estimator $\widehat{\alpha}_n$ of α over 10 000 experiments, each with $n = 2000$ and $\beta = 0.5$. The random variables r_i are uniformly distributed on the interval $[0.25, 1.75]$. The red line indicates the true value of α. Please note we employed the true value of β in order to estimate α.

- We consider *undirected* graphs: a link (i, j) will exist if at least one link among (i, j) and (j, i) exist.
- We will focus, in this section, on pure *homophily*: therefore we will set $\mathbf{W} = I$. In this way the only pairs of features that affect the existence of a link will be those in the form (k, k).
- We also omit self-loops, i.e., edges of type (i, i).

We fix *a priori* the number of nodes n and the (approximate) number of edges m (i.e., density) we aim at; then, every experiment consists essentially in two phases:

- generating a node-feature matrix Z for n nodes (with certain values for the parameters α and β and with r_i uniformly distributed on the interval $[\varepsilon, 2 - \varepsilon]$ for a certain ε);
- building the graph according to one of the models described in Sections 3.2 and 3.3.

The second phase needs to fix some further parameters: $\mathbf{W}$ (the feature-feature influence matrix), the function ϕ and, for the mixed models (FFBA and FFJR, section 3.3), the parameter δ.

As said above, throughout this section we will assume that $\mathbf{W} = I$ for the sake of simplicity. Regarding the activation function, we take ϕ as a sigmoid function given by:

$$\phi(x) = \left(e^{K(\vartheta - x)} + 1\right)^{-1}.$$

In other words, the existence of an edge (i, j) depends simply on the number of features that i and j share (this is an effect of choosing $\mathbf{W} = I$). More features in common induce larger probability: the sigmoid function smoothly increases (from 0 to 1) around a threshold ϑ, and $K > 0$ controls its smoothness; when $K \to \infty$ we obtain a step function and edges are chosen deterministically based on whether the two involved nodes share more than ϑ features or not.

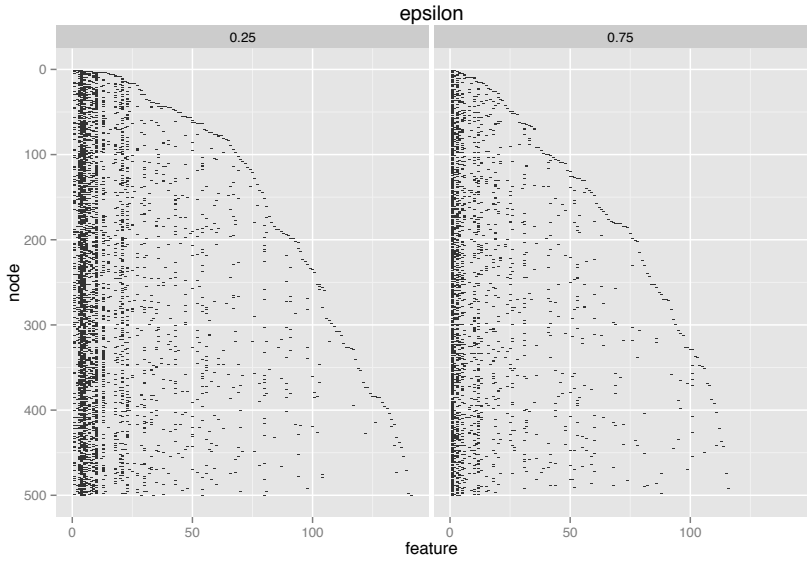


Figure 4.5: Here $n = 500$, $\alpha = 3$, $\beta = 0.5$ and the random variables r_i are uniformly distributed on the interval $[\varepsilon, 2 - \varepsilon]$ (so that $m_r = 1$ and $Var[r_i] = (1 - \varepsilon)^2/3$), for different values of ε ($\varepsilon = 0.25$ and $\varepsilon = 0.75$). The figure shows how ε affects the shape of Z.

In the experiments, we fix K and determine ϑ on the basis of the desired density of the graph (or, equivalently, the desired number of edges m); in practice[3], this is obtained by solving numerically the equation

$$E\left[\sum_{1\leq j<i\leq n} A_{i,j}\right] = \sum_{1\leq j<i\leq n} \phi\left(\sum_{h,k} Z_{i,h} W_{h,k} Z_{j,k}\right) = m$$

for the indeterminate ϑ (using, for example, Newton's method). Since $\mathbf{W} = I$ the equation in fact simplifies into

$$\sum_{1\leq j<i\leq n} \left(e^{K(\vartheta - \sum_h Z_{i,h} Z_{j,h})} + 1\right)^{-1} = m.$$

With these assumptions, every experiment depends on the parameters used for generating Z (i.e., α, β and ε), on K (that controls the smoothness of the sigmoid function), on ϑ (ultimately responsible for the graph density) and on δ (for the mixed models). In the graphs produced by each simulation, we took into consideration the degree distribution, the percentage of reachable pairs (i.e., the fraction of pairs of nodes that are reachable) and the distribution of distances (lengths of shortest paths); the latter data are computed using a probabilistic algorithm [173].

[3]The described method needs some (obvious) adjustments when applied to the mixed models, to take into account the edges generated by preferential attachment.

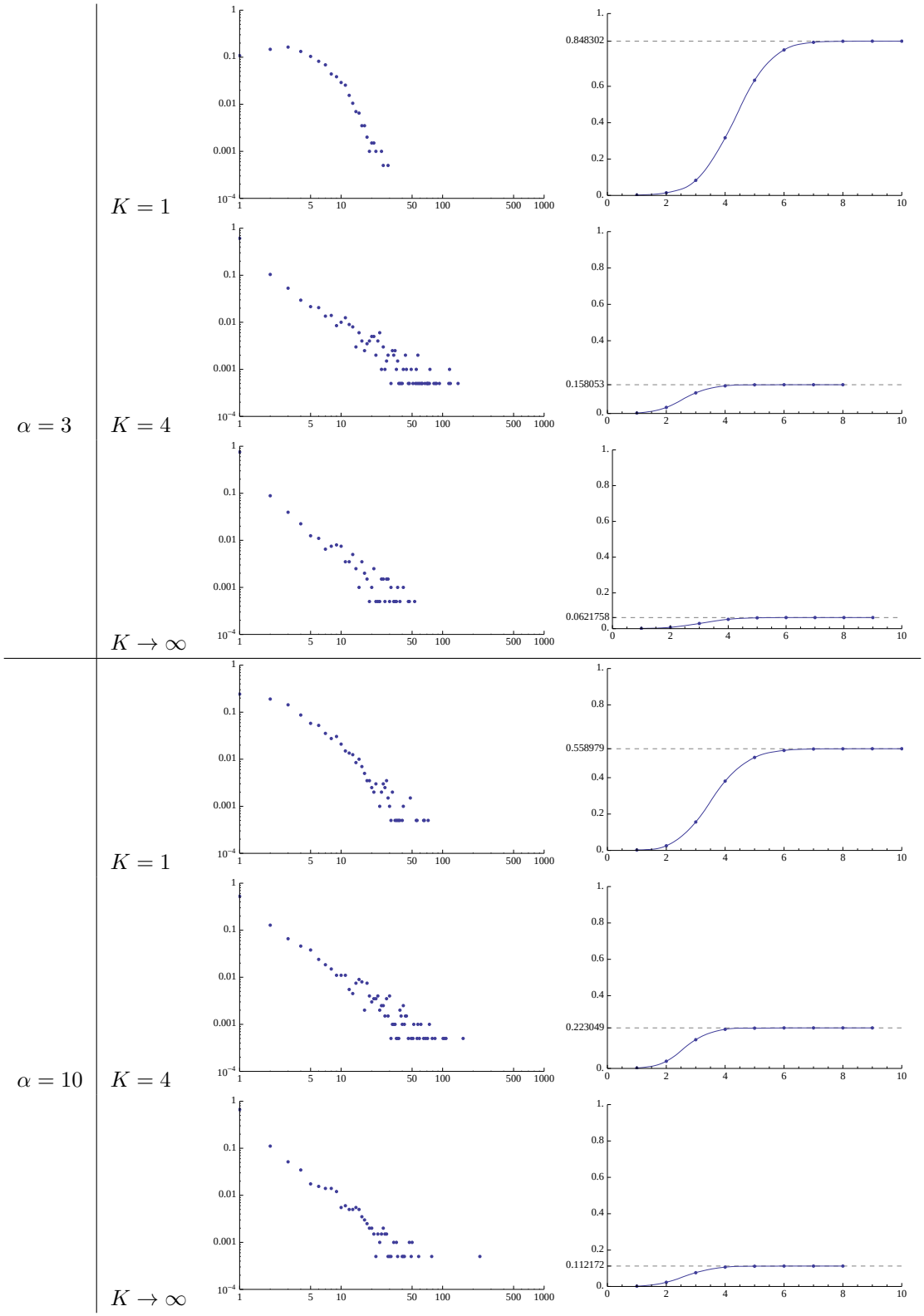


Figure 4.6: Properties of graphs generated by our model. We show the degree distribution in a log-log plot and the fraction of pairs at distance at most k; in the latter, we highlight the peak value (fraction of mutually reachable pairs).

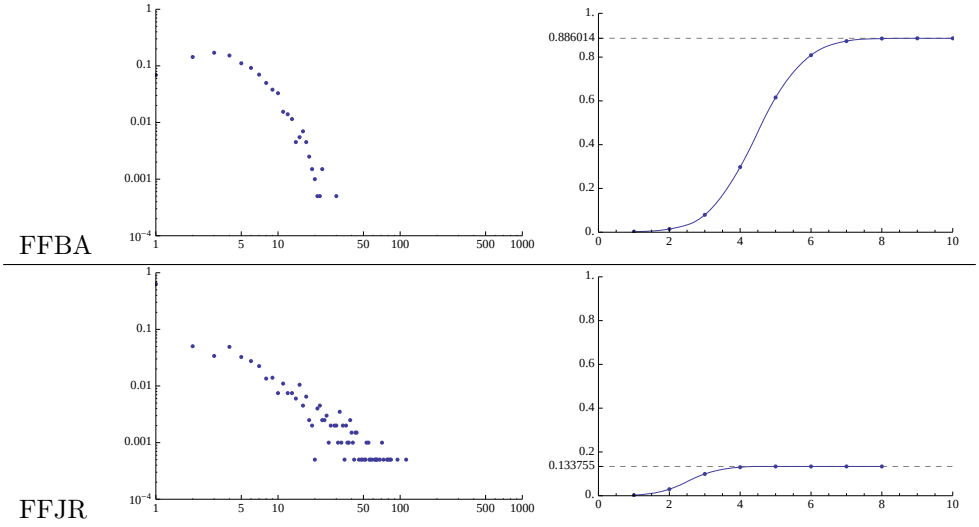


Figure 4.7: Properties of graphs generated by mixed models with $K = 1$ and $\delta = 0.75$. We show the degree distribution in a log-log plot and the fraction of pairs at distance at most k; in the latter, we highlight the peak value, indicating how many pairs of nodes are mutually reachable. The parameters of the underlying node-feature matrix model are $\alpha = 3$ and $\beta = 0.75$ and the r_i's are uniformly distributed on $[0.75, 1.25]$.

Some of the results obtained (for $n = 2\,000$ and[4] $m = 4\,000$) for the FF model are shown in Figure 4.6. For those experiments, the underlying node-feature matrix is generated with $\beta = 0.75$ and r_i uniformly distributed on the interval $[0.75, 1.25]$; we compare $\alpha = 3$ (resulting in ≈ 1200 features) with $\alpha = 10$ (≈ 4000 features). Results regarding mixed models are reported in Figure 4.7.

The properties of the obtained graphs can be summarized as follows:

- the pure FF model exhibits a behavior that strongly depends on the smoothness parameter K (see Fig. 4.6):
  - for $K = 1$, the degree distribution is power-law only when α is large (e.g., $\alpha = 10$), whereas the distribution is often non-monotonic for smaller α's, especially on large graphs; the fraction of reachable pairs is quite large (between 40% and 90%);
  - for $K = 4$, degrees are always distributed as a power-law (with exponents around 3), but the graph becomes largely disconnected (the reachable pairs are never more than 20%): this is because nodes with the same degree tend to stick together (assortativity), forming a highly connected component and leaving the remaining nodes isolated;
  - for $K \to \infty$, the power-law distribution of degrees is even more clear-cut, but the number of reachable pairs becomes smaller (no more than 10%); the exponent of the power-law distribution depends on α, with larger α's yielding larger absolute values of the (negative) exponents;

[4] We observed analogous phenomena also for larger and denser networks; we hereby report only the smaller case for the sake of readability of the pictures.

- the FFBA model (see Fig. 4.7), presented in section 3.3.1, increases slightly the number of reachable pairs in all cases; the shape of the power-law distribution is essentially unchanged with respect to the pure FF model;

- finally, for the FFJR model (see Fig. 4.7), presented in section 3.3.2, we observe a reduced connectivity; this is due to holding the expected number of edges as a constant, while devoting some of them to closing triangles – an operation that cannot increase connectivity. The degree distribution seems closer to a power-law with respect to the pure FF model.

4.6 Ranking from features

One of the most important notions that researchers have been trying to capture in complex networks is "node centrality": ideally, every node in the graph has some degree of importance within the network under consideration, and one expects such importance to surface in the structure of the social network; centrality is a quantitative measure that aims at revealing the importance of a node.

It was noted [32] that the most used centrality measures can be broadly categorized into geometric measures (e.g., closeness centrality [21], Lin's index [125] or harmonic centrality [32]), path-based measures (e.g., betweenness [11]) and spectral measures (e.g., PageRank [148] or Katz's index [110]).

In this section, we will consider only features in nodes (without the links they carry) in order to rank nodes. The purpose of this section is limited to show how the model for the evolution of features presented in section 4.3 can naturally lead to a node ranking.

Indeed, in section 4.3 we introduced a new version of the Indian Buffet process to explain how the node-features matrix $\mathbf{Z}$ can be generated. As in the classical Indian Buffet process, nodes arrive in a sequence $\{0, \dots, n-1\}$, and node i selects a set of new features not selected by nodes $\{0, \dots, i-1\}$ and also a set of features already selected by others; the latter are chosen accordingly to the popularity of those features (in a rich-get-richer way).

Our work there was innovative in two main ways:

- We provided parameters with a clear semantics and their unbiased estimators.

- We introduced *competition*: not all nodes are created equals, but each one is equipped with a *fitness value* R_i that express how influencing is that node in popularizing its features.

This second point is the key of this section: since our model for the evolution of features described formally how the fitness of nodes impact the likelihood of a given $\mathbf{Z}$, we can employ this to extract in a rigorous way a node ranking from a node-feature association. In doing this, we will in fact first maximize the likelihood of the node-feature matrix we are seeing (employing the estimators for α and β we provided in section 4.3), and then we will provide a naive algorithm, which work surprisingly good for this task.

4.6.1 Likelihood of fitness values of nodes

Now our purpose is to find, under the assumption of our model, a procedure to get information on the random variables r_i from the data, that typically are the values of $\mathbf{z}_1, \ldots, \mathbf{z}_n$, i.e., n rows of the matrix $\mathbf{Z}$, where n is the number of the observed nodes.

We are going to develop a maximum log-likelihood procedure to find a plausible realization $\widehat{r}_1, \ldots, \widehat{r}_{k_n}$ of $r_1, \ldots, r_{k_n}$, for a suitable $k_n \leq n$. Note that we ideally would like to find a probable realization for all the fitness parameters of the observed nodes (not only for the first k_n nodes), but we do not possess the same amount of information about all r_i: in particular, while r_1 influences all the subsequent observed rows of the matrix $\mathbf{Z}$, r_{n-1} has only influence over $\mathbf{z}_n$, and the value r_n has no influence at all. So we cannot expect to find good values for all the random variables.

With the above purpose in mind, we now give a general expression for the conditional probability of observing $\mathbf{z}_1 = \widehat{\mathbf{z}}_1, \ldots, \mathbf{z}_n = \widehat{\mathbf{z}}_n$ given $r_1, \ldots, r_{n-1}$.

The first row $\mathbf{z}_1$ is simply identified by $L_1 = N_1$ and so

$$\begin{aligned} P(\mathbf{z}_1 = \widehat{\mathbf{z}}_1) &= P(N_1 = n_1 = \left|\{k : Z_{1,k} = 1\}\right| \\ &= Poi(\alpha)\{n_1\} = e^{-\alpha}\frac{\alpha^{n_1}}{n_1!}. \end{aligned}$$

Then the second row is identified by the values $Z_{2,k}$ with $k = 1, \ldots, L_1 = N_1$ and by N_2 and so

$$\begin{aligned} &P(\mathbf{z}_2 = \widehat{\mathbf{z}}_2 | \mathbf{z}_1, r_1) = \\ &P(Z_{2,k} = \widehat{Z}_{2,k} \text{ for } k = 1, \ldots, L_1,\ N_2 = n_2 = \left|\{k > L_1 : \widehat{Z}_{2,k} = 1\}\right| | \mathbf{z}_1, r_1) = \\ &\prod_{k=1}^{L_1} P_1(k)^{Z_{2,k}}(1 - P_1(k))^{1 - Z_{2,k}} \times Poi(\Lambda_1)\{n_2\}, \end{aligned}$$

where $P_1(k)$ is defined in (4.1) and Λ_1 is defined in (4.2).

The general formula is

$$\begin{aligned} &P(\mathbf{Z}_{j+1} = \widehat{\mathbf{z}}_{j+1} | \mathbf{z}_1, r_1, \ldots, \mathbf{z}_j, r_j) = \\ &P\Big(Z_{j+1,k} = \widehat{Z}_{j+1,k} \text{ for } k = 1, \ldots, L_j, \\ &\qquad N_{j+1} = n_{j+1} = \left|\{k > L_j : Z_{j+1,k} = 1\}\right| | \mathbf{z}_1, r_1, \ldots, \mathbf{z}_j, r_j\Big) = \\ &\prod_{k=1}^{L_j} P_j(k)^{Z_{j+1,k}}(1 - P_j(k))^{1 - Z_{j+1,k}} \times Poi(\Lambda_j)\{n_{j+1}\}, \end{aligned}$$

where $P_j(k)$ is defined in (4.1) and Λ_j is defined in (4.2).

Hence, for n nodes, we can write a formula for the conditional probability of observing $\mathbf{z}_1 = \widehat{\mathbf{z}}_1, \ldots, \mathbf{z}_n = \widehat{\mathbf{z}}_n$ given $r_1, \ldots, r_{n-1}$:

$$P(\mathbf{z}_1 = \widehat{\mathbf{z}}_1, \ldots, \mathbf{z}_n = \widehat{\mathbf{z}}_n | r_1, \ldots, r_{n-1}) = \\ P(\mathbf{z}_1 = \widehat{\mathbf{z}}_1) \prod_{j=1}^{n-1} P(\mathbf{z}_{j+1} = \widehat{\mathbf{z}}_{j+1} | \mathbf{z}_1, r_1, \ldots, \mathbf{z}_j, r_j). \tag{4.8}$$

4.6.2 Ranking through Gibbs sampling

The algorithm we applied is essentially a MCMC (Markov Chain Monte Carlo) method [79], which uses the basic principle of Gibbs sampling [43]: fix all components of a vector except one and compare the different values of the likelihood obtained for various values of the non-fixed component.

The method employs the aforementioned formula (4.8) for the conditional probability of observing $\mathbf{z}_1 = \widehat{\mathbf{z}}_1, \ldots, \mathbf{z}_n = \widehat{\mathbf{z}}_n$ given the values of $r_1, \ldots, r_{n-1}$. Precisely, using the symbol $\widehat{\mathbf{Z}}$ in order to denote the matrix with rows $\widehat{\mathbf{z}}_1, \ldots, \widehat{\mathbf{z}}_n$ and the symbol $\widehat{\mathbf{r}}$ in order to denote a vector of component $\widehat{r}_1, \ldots, \widehat{r}_n$, set

$$P(\mathbf{Z} = \widehat{\mathbf{Z}} | \mathbf{r} = \widehat{\mathbf{r}}) = P(\mathbf{z}_1 = \widehat{\mathbf{z}}_1, \ldots, \mathbf{z}_n = \widehat{\mathbf{z}}_n | r_1 = \widehat{r}_1, \ldots, r_n = \widehat{r}_n). \tag{4.9}$$

We want to find a vector $\widehat{\mathbf{r}}$ that is a point maximizing the likelihood function (4.9) corresponding to the observed $\widehat{\mathbf{Z}}$.

The basic algorithm[5] is described in Alg. 1. It is regulated by these parameters:

- $\mathbf{r}^0 \in \mathbb{R}^n$ is the initial guess for $\widehat{\mathbf{r}}$;
- $J \in \mathbb{N}^+$ is the number of *jumps*, i.e., the number of the new values analyzed for a certain component r_i at each step;
- $\sigma \in \mathbb{R}^+$ is the standard deviation of each "jump".

To check for convergence, we keep track of all the increments of the log-likelihood obtained in each one of the last n steps; as a stopping criterion, we check when the maximum of these increments is under a certain threshold t: if it is, we stop the algorithm.

It is worth to note that, given $\mathcal{L}(r_i)$, it is possible to find $\mathcal{L}(h)$ without re-doing the whole computation. In fact, let us consider the product in (4.8): a change from r_i to h must be taken into account only from the i-th factor onward – that is, for the factors that come after $P(\mathbf{z}_i = \widehat{\mathbf{z}}_i | \mathbf{z}_1, \mathbf{r}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{r}_{i-1})$. In particular, let $\delta = h - \widehat{r}_i$ – i.e. the update to add to the estimated fitness $\widehat{r}_i$ of node i. Then, for each j-th factor with $j \geq i$ in (4.8) – i.e., how likely are the features chosen by node j – we have to:

- add δ to the term $\sum_{i=1}^{j} r_i$, inside Λ_j and $P_j(k)$; that is, as defined in (4.1) and (4.2), the denominator normalizing each r_i to their total sum.

[5]We point out that our algorithm can not be considered a proper statistical estimation procedure for the fitness parameters. It resembles the Bayesian *Maximum a posteriori probability* (MAP) estimation when the a priori distribution is a uniform distribution, but the number of parameters in our case increases with the number of observations.

Algorithm 1 Basic Monte Carlo algorithm to find $\widehat{\mathbf{r}}$.

INPUT:

$\widehat{\mathbf{z}}_1, \ldots, \widehat{\mathbf{z}}_n$, the observed rows of the node-feature matrix $\widehat{\mathbf{Z}}$

OUTPUT:

$\widehat{\mathbf{r}}$, a maximum point for the likelihood function associated to the input data

DESCRIPTION:

1. $\widehat{\mathbf{r}} \leftarrow \mathbf{r}^0$
2. Repeat the following loop until convergence:
   (a) Choose a random node $i \in \{1, \ldots, n\}$
   (b) Extract J values $h_1, \ldots, h_J$ from the normal distribution $\mathcal{N}(r_i, \sigma^2)$; re-sample each h_j until $h_j > 0$.
   (c) For each value h_j, compute
   $$\mathcal{L}(h_j) := P(\mathbf{Z} = \widehat{\mathbf{Z}} | r_1 = \widehat{r}_1, \ldots, r_i = h_j, \ldots, r_n = \widehat{r}_n)$$
   (d) $\widehat{r}_i \leftarrow \underset{h \in \{r_i, h_1, \ldots, h_J\}}{\arg\max} \mathcal{L}(h)$

- add δ to the numerator of $P_j(k)$ when k is s.t. $Z_{i,k} = 1$; that is to say, change the global weight of a feature only if the node we changed has that feature.

Every other term in the equation remains unchanged and does not need to be computed again. This remark allows us to speed up the implementation considerably.

Figure 4.8 confirms that the algorithm moves toward a vector $\widehat{\mathbf{z}}$ maximizing $P(\mathbf{Z} = \widehat{\mathbf{Z}} | \mathbf{r} = \widehat{\mathbf{r}})$ and shows that the algorithm effectively converges. The obtained outputs will be discussed in details in Section 4.6.3.

As already said, one point that we need to keep in mind is that we do not possess the same amount of information about all the random variables r_i: in particular, while r_1 influences all the subsequent rows of the matrix $\mathbf{Z}$, r_{n-1} has only influence over the last one. So we cannot expect the output values to be very accurate for the last segment. For this reason, we also implemented a variant of the algorithm that considers only the first k_n nodes. Thus, we have another algorithm parameter k_n so that the choice of the jumping node at step 2(a) is restricted to $i \in \{1, \ldots, k_n\}$ and, finally, the output will be the corresponding segment of $\widehat{\mathbf{r}}$, i.e., $\widehat{r}_1, \ldots, \widehat{r}_{k_n}$. This variant converges faster and moreover it allows to use larger values of the algorithm parameter J.

Another relevant point is that the parameters α and β enter the expression (4.8). Therefore, in practice, before applying the algorithm, we need to estimate them. As shown in Remark 4.3.1, we are able to estimate β starting from the observed values of the matrix Z. On the other hand, as shown in Remark 4.3.2, the estimation of α presupposes the knowledge of the mean value m_r of the fitness parameters r_i (except for the special case $\beta = 1$). Hence, we are in the situation in which, in order to get information on the fitness parameters by the proposed algorithm, we need to estimate α and β, but, in order to estimate α, we need to know the mean value m_r.

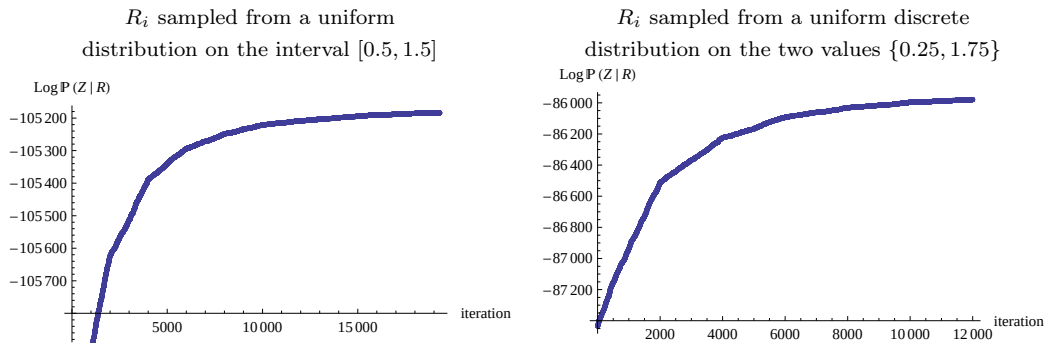


Figure 4.8: Value of the log-likelihood during the execution of the algorithm, for different distributions of r_i. The chosen algorithm parameters are $\sigma^2 = 1$, $J = 4$ and $\mathbf{r}^0 = \mathbf{1}$ (the vector with all 1's). The matrix Z has 2000 rows (nodes) and it was generated with $\alpha = 3$ and $\beta = 0.9$.

This problem can be partially solved as follows. Since the term $P(Z_1 = z_1)$ does not contain the r_i's, the research of a vector $\widehat{\mathbf{r}}$ that maximizes (4.9) is equivalent to the research of a vector $\widehat{\mathbf{r}}$ maximizing the product

$$\prod_{j=1}^{n-1} P(\mathbf{z}_{j+1} = \widehat{\mathbf{z}}_{j+1} | \widehat{\mathbf{z}}_1, r_1, \ldots, \widehat{\mathbf{z}}_j, r_j)$$

in formula (4.8). On the other hand, each term of the above product contains the inclusion probabilities $P_j(k)$, that are invariant with respect to the normalization of the r_i's by their mean value m_R, and the Λ_j's that have the property

$$\Lambda_j = f(\alpha, \beta, \mathbf{r}) = f(\alpha/(m_R)^{1-\beta}, \beta, \mathbf{r}/m_r)$$

(where $\mathbf{r}/m_r$ denotes the vector with components r_i/m_r). Consequently, starting from the observed values of the matrix $\mathbf{Z}$, we can

- first, estimate β by Remark 4.3.1;
- then estimate $\alpha' = \alpha/(m_R)^{1-\beta}$ by Remark 4.3.2 (i.e., $\widehat{\alpha}'_n$ equal to $\widehat{\gamma}_n$ or $\beta\,\widehat{\gamma}_n$ according to the estimated value of β);
- finally, extract a plausible realization $\widehat{\mathbf{r}}' = \widehat{\mathbf{r}}/m_r$ (of the random variables $r'_i = r_i/m_r$) as a maximum point of the corresponding expression of the likelihood with the estimated value of β and α'.

  Therefore, the output of the algorithm will be α', β and a plausible realization $\widehat{\mathbf{r}}'$ of the random variables $r'_i = r_i/m_r$.

Finally, we highlight that it is possible to experiment other variants of the algorithm, for example, by using a distribution different from the normal for the jumps, or changing σ during the execution (e.g., reducing it according to some "cooling schedule", as it happens in simulated annealing [76]). Additionally, instead of looking for the values on the whole positive real line, we could restrict the research on a suitable interval (guessed for the particular real case).

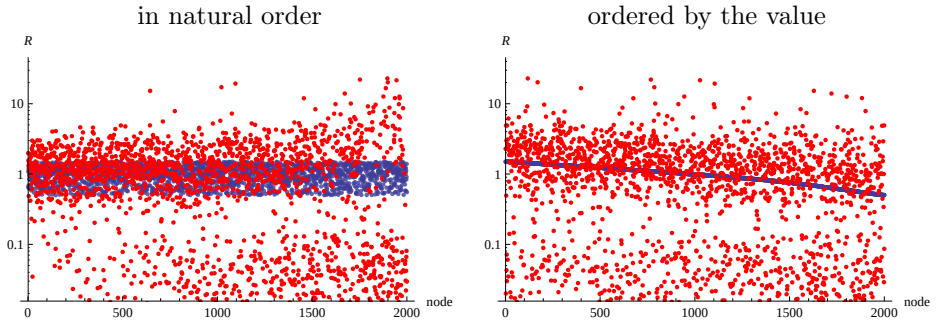


Figure 4.9: The extracted realization $\widehat{\mathbf{r}}$ (in red) versus the true realization $\mathbf{r}$ (in blue), with two different orderings, in the case of uniform distribution on the interval $[0.5, 1.5]$. The empirical mean of the first $\frac{n}{2}$ extracted values is 1.18.

4.6.3 Experiments

We proceeded to test empirically how the Monte Carlo method performs in recovering the information on the fitness parameters r_i. We tested its behavior against various distributions of r_i; specifically, a uniform distribution on an interval, a two-class uniform distribution, and finally a discrete power-law distribution with 10 classes. In the following of this section we illustrate the details of such experiments, while, in the next section, we will try to measure the performance of the proposed technique.

In every experiment, the matrix Z has $n = 2\,000$ nodes and it was generated with $\alpha = 3$ and $\beta = 0.9$. The Monte Carlo algorithm parameters were set as follows: $\sigma^2 = 1$, $J = 4$ and $\mathbf{r}^0 = \mathbf{1}$ (the vector with all 1's).

For the first experiment, each r_i is sampled from the uniform distribution on the interval $[0.5, 1.5]$. We used the previously discussed techniques to find the estimates of α and β: the estimated values are $\widehat{\alpha} = 3.095$ and $\widehat{\beta} = 0.893$ (note that we have $m_r = 1$ and so $\alpha = \alpha'$ and $\widehat{\mathbf{r}} = \widehat{\mathbf{r}}'$). Then, we tried the proposed Monte Carlo algorithm with the stopping threshold $t = 1/4$. Results are visualized in Figure 4.9, according to two different orderings of the nodes:

i) in the natural order, so that we confirm that our predictions are better for the first (i.e., the oldest) nodes than for the last (i.e., the youngest) ones;

ii) ordered by their true fitness values, so that we can show that we are, more or less, able to reconstruct the relative order of the fitness parameters (this fact will be made clearer in Section 4.6.4).

In the second experiment, we applied our algorithm to a discrete case: we sampled the fitness parameters r_i from a set of only two values, $\{0.25, 1.75\}$, each with probability $\frac{1}{2}$. We left the parameters of the model and the ones of the algorithm unaltered, except for moving the stopping threshold t from $1/4$ to 1. The estimated values for $\alpha = 3$ and $\beta = 0.9$ are, respectively, $\widehat{\alpha} = 2.922$ (again $m_r = 1$ and so $\alpha = \alpha'$ and $\widehat{\mathbf{r}} = \widehat{\mathbf{r}}'$) and $\widehat{\beta} = 0.903$. The results of this second experiment are more encouraging (we will see precise measurements in Section 4.6.4). In this case, the

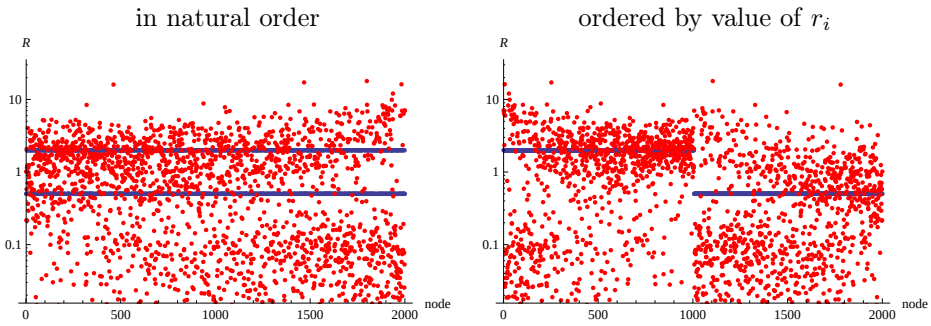


Figure 4.10: The extracted realization $\widehat{\mathbf{r}}$ (in red) versus the true realization $\mathbf{r}$ (in blue), with two different orderings, in the case of the uniform discrete distribution on the two values $\{0.25, 1.75\}$. The empirical mean of the the first $\frac{n}{2}$ extracted values is 1.33.

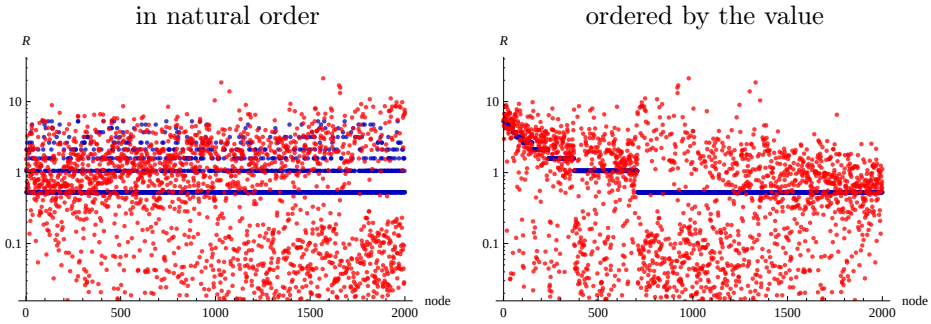


Figure 4.11: The extracted realization $\widehat{\mathbf{r}}$ (in red) versus the true realization $\mathbf{r}$ (in blue), with two different orderings, for the normalized discrete Zipf's distribution with exponent 2 and 10 values. The empirical mean of the the first $\frac{n}{2}$ extracted values is 1.25.

output values of the algorithm are closer to the true ones (see Figure 4.10). Moreover, we can still observe the same phenomena, i) and ii), described above.

Finally, we applied the algorithm to a third case: we sampled r_i from a normalized power-law discrete distribution, with 10 possible values – specifically, a normalized discrete Zipf's law with exponent 2 and number of values 10. We left both algorithm and model parameters unaltered and we used 1 as the stopping threshold t.

The estimated values for $\alpha = 3$ and $\beta = 0.9$ are, respectively, $\widehat{\alpha} = 3.595$ (again $m_r = 1$ and so $\alpha = \alpha'$ and $\widehat{\mathbf{r}} = \widehat{\mathbf{r}}'$) and $\widehat{\beta} = 0.868$.

Results for this case show that – despite the fact that we have now a discrete distribution with more than two values – our approach can recover information (especially for larger fitness values), as can be seen in Figure 4.11 and in Section 4.6.4.

We conclude this section noting that, for each of the experiments, the Monte Carlo algorithm looks for the values of the fitness parameters on the whole positive real line – not on a suitable interval for each case.

Algorithm 2 NAIVE heuristic to find $\widehat{\mathbf{r}}$.

INPUT: the node-feature matrix $\widehat{\mathbf{Z}}$
OUTPUT: $\widehat{\mathbf{r}}$
DESCRIPTION:
For each node i, from 1 to n, do the following:

1. Let $F_i = \{k \big| Z_{i,k} = 1\}$
2. Let $\mu = 0$
3. For each $k \in F_i$:
   (a) Let $S_{\text{before}} = 0$
   (b) Let $j_{\min} = i$
   (c) For each node j s.t. $1 < j < i$ and $Z_{j,k} = 1$:
      i. $S_{\text{before}} \leftarrow S_{\text{before}} + 1$
      ii. $j_{\min} \leftarrow \min(j_{\min}, j)$
   (d) Let $S_{\text{after}} = 0$
   (e) For each node j s.t. $i < j < n$ and $Z_{j,k} = 1$:
      i. $S_{\text{after}} \leftarrow S_{\text{after}} + 1$
   (f) $\mu \leftarrow \mu + \frac{S_{\text{before}}/(i - j_{\min})}{S_{\text{after}}/(n-i+1)}$
4. $\widehat{r_i} \leftarrow \frac{\mu}{|F_i|}$

4.6.4 Results in ranking

In a real application, we may content ourselves in finding not the realized fitness parameters themselves but rather their ordering, that is, the ordering of the nodes from larger to smaller values of the fitness parameter. As mentioned before, the ordering induced by the fitness values can be thought, in fact, as a ranking, ordering nodes from the "most influential" to the least.

To resolve this task, we can easily add some baselines, that could be able to obtain similar results. The algorithm we introduced is in fact able to properly maximize the likelihood defined by our model; however, if we content ourselves in finding the ordering, some easier alternatives could be both faster and easier to implement. Therefore, we will now propose two different naive baselines for finding the ordering induced by the fitness values, that we will compare experimentally to the Monte Carlo algorithm.

Baseline. Since the final effect of a fitness value r_i on the development of the matrix $\mathbf{Z}$ is that the features displayed by a node i with a large r_i should be more popular than those displayed by a node j with a low r_j, this suggests a very simple baseline to compare our algorithm to: to estimate r_i, we just measure how popular the features k s.t. $Z_i, k = 1$ are. Formally, if we indicate with $F_i = \{k \big| Z_{i,k} = 1\}$ the features displayed by node i, then:

$$r_{\text{BASELINE}}(i) = \frac{1}{|F_i|} \sum_{k \in F_i} \frac{\sum_{j=i+1}^{n} Z_{i,k}}{n - i} \tag{4.10}$$

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor \sqrt{n} \rfloor = 44$	0.281	0.206	0.463
$k_n = \frac{n}{2} = 1000$	0.229	0.188	0.337
$k_n = n = 2000$	0.150	0.139	0.155

Table 4.1: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by MONTE CARLO in the case of the uniform distribution on the interval $[0.5, 1.5]$.

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor \sqrt{n} \rfloor = 44$	0.676	0.593	0.713
$k_n = \frac{n}{2} = 1000$	0.586	0.585	0.625
$k_n = n = 2000$	0.438	0.477	0.434

Table 4.2: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by MONTE CARLO the case of the uniform discrete distribution on the two values $\{0.25, 1.75\}$.

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor \sqrt{n} \rfloor = 44$	0.735	0.762	0.772
$k_n = \frac{n}{2} = 1000$	0.453	0.516	0.803
$k_n = n = 2000$	0.313	0.402	0.543

Table 4.3: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by MONTE CARLO in the case of the normalized discrete Zipf's distribution with exponent 2 and 10 values.

This formula gives us a rough measure of how popular the features displayed by i turns out to be. We will refer to it as BASELINE.

Naive. Furthermore, we will employ a slightly more sophisticated heuristics. In fact, the effect of a large r_i is not to make features popular in general, but to make them *more* popular after node i acquire them. Therefore, we need to compare how the popularity of a feature k adopted by i changes before and after it has been adopted by i. To do that, we also need to take into consideration the fact that such feature may be born with an arbitrary node j s.t. $1 \leq j < i$. We took care of these considerations, and use them to present a naive algorithm, for which we provide the pseudocode in Algorithm 2. We also note that this algorithm can also be efficiently parallelized. We will refer to this heuristic as NAIVE.

Experimental results. To evaluate if we can extract values $\widehat{r}_i$ that respect this ordering, we decided to compare the drawn vector $\widehat{\mathbf{r}}$ with the true realization $\mathbf{r}$ by the use of Kendall's τ and some variants of it.

To keep track of the fact that, as said before, the first nodes contain more information than the last ones, we evaluated Kendall's τ not only on the whole vector but also on a short initial segment of size $k_n = n/2$ or $k_n = \sqrt{n}$. Besides this, we tried to use a variant of Kendall's τ [186], that we apply in two separate and different ways:

1. inducing a hyperbolic decay based on the position of the nodes – that is, weighting more

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor\sqrt{n}\rfloor = 44$	-0.068	-0.020	-0.212
$k_n = \frac{n}{2} = 1000$	-0.009	0.011	-0.016
$k_n = n = 2000$	0.009	0.021	-0.095

Table 4.4: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by BASELINE in the case of the uniform distribution on the interval $[0.5, 1.5]$.

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor\sqrt{n}\rfloor = 44$	-0.180	-0.196	0.132
$k_n = \frac{n}{2} = 1000$	-0.018	-0.068	0.237
$k_n = n = 2000$	-0.030	-0.068	0.247

Table 4.5: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by BASELINE the case of the uniform discrete distribution on the two values $\{0.25, 1.75\}$.

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor\sqrt{n}\rfloor = 44$	-0.019	-0.028	0.074
$k_n = \frac{n}{2} = 1000$	0.027	0.010	0.217
$k_n = n = 2000$	0.026	0.011	0.215

Table 4.6: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by BASELINE in the case of the normalized discrete Zipf's distribution with exponent 2 and 10 values.

the first (the oldest) nodes, and less the last (the youngest) ones;

2. inducing a hyperbolic decay based on the true realized values r_i – that is, assigning a higher weight to the nodes with a greater fitness parameter r_i.

The results of these measures are reported in tables 4.1, 4.4 and 4.7 for the experiment with the uniform distribution on an interval, in tables 4.2, 4.5 and 4.8 for the experiment with the uniform discrete distribution on the two values $\{0.25, 1.75\}$, and in tables 4.3, 4.6 and 4.9 for the discrete Zipf's distribution with 10 values and exponent 2. The tables show that, although we are unable to reconstruct the actual realized values of the fitness parameters, MONTE CARLO actually recovers some information about node ranking (and that, as already seen before, the output of the Monte Carlo algorithm is better for the discrete cases), while BASELINE is not able to recover any information about node ranking (very often it is anti-correlated with the actual ordering).

Most surprisingly, however, is that the results obtained by NAIVE are instead strikingly good, and it often outperforms MONTE CARLO in experiments. As testified by the summarizing plot in Figure 4.12, MONTE CARLO obtains much worst performances in the uniform-distributed case, and similar performances in the discrete-distributed cases. This insight can be helpful in some settings: the fitness vector obtained by NAIVE can in fact be thought as a good (order-wise) approximation of our fitness values.

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor \sqrt{n} \rfloor = 44$	0.852	0.877	0.978
$k_n = \frac{n}{2} = 1000$	0.619	0.677	0.953
$k_n = n = 2000$	0.603	0.660	0.953

Table 4.7: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by NAIVE in the case of the uniform distribution on the interval $[0.5, 1.5]$.

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor \sqrt{n} \rfloor = 44$	0.438	0.321	0.694
$k_n = \frac{n}{2} = 1000$	0.434	0.382	0.680
$k_n = n = 2000$	0.428	0.384	0.680

Table 4.8: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by NAIVE the case of the uniform discrete distribution on the two values $\{0.25, 1.75\}$.

Considered nodes	Kendall's τ	τ weighted by position	τ weighted by value
$k_n = \lfloor \sqrt{n} \rfloor = 44$	0.629	0.572	0.883
$k_n = \frac{n}{2} = 1000$	0.470	0.489	0.907
$k_n = n = 2000$	0.427	0.464	0.910

Table 4.9: Comparing orderings induced by the true realization $\mathbf{r}$ versus the $\widehat{\mathbf{r}}$ found by NAIVE in the case of the normalized discrete Zipf's distribution with exponent 2 and 10 values.

4.7 Discussion

In this chapter we proposed and analyzed a model that combines two aspects characterizing the evolution of a network:

1. Behind the adjacency matrix of a network there is a *node-feature association*, in the sense that each node is characterized by a number of features and the probability of the existence of an edge between two nodes depends on the features they have. Such a scheme can be seen in many real cases (e.g., interests in a network of friends, or topics in a citation network).

2. Not all nodes are equally successful in transmitting their own features to the new nodes. Each node n is characterized by a *random fitness parameter* r_n describing its ability to transmit the node's features: the greater the value of the random variable r_n, the greater the probability that a feature of node n will also be a feature of a new node; this fact typically translates into a greater probability of the creation of an edge between n and the new node. Consequently, a node's connectivity does not depend on its age alone (so also "young" nodes are able to compete and success in acquiring links). We called this aspect *competition*.

   This assumption in many cases offers not only realism ("not all web pages are equal", e.g.), but also useful insights: as we have seen in Section 4.6, allowing for this degree of freedom

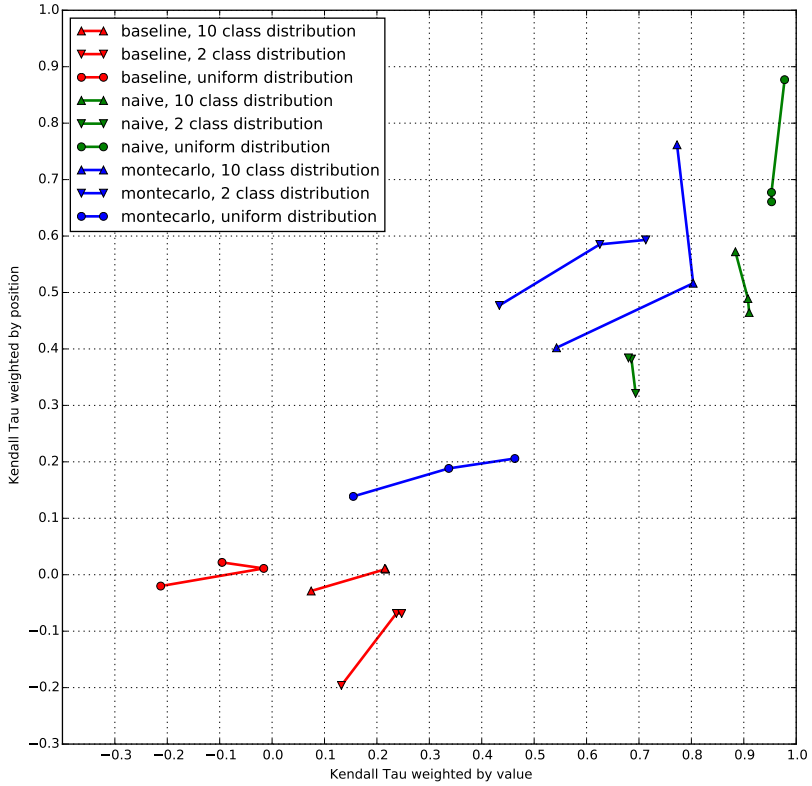


Figure 4.12: A summarizing plot of tables 4.1 to 4.9, plotting the Kendall's τ weighted by value on the x-axis and weighted by position on the y-axis. The different colors represents our different algorithms (BASELINE, NAIVE, and MONTECARLO). For each algorithm, we display a different line plot for the various distributions of the fitness values we tested it against; for each distribution, we plot the three different values we obtained when testing on $\sqrt{n}$, $\frac{n}{2}$ and n nodes.

to the model means gaining information about it from how different nodes behaved, and even identifying "fittest" nodes among the oldest ones.

We shaped the first aspect by the definition of a model which connects the pair of feature-vectors of two nodes, say i and j, to the probability of the existence of an edge between i and j. Other examples, which are related to the Bayesian framework based on the standard Indian Buffet model, can be found in previous literature [136, 139, 149, 162].

We modeled the second aspect by the definition of a stochastic dynamics for a bipartite "node-feature" network, where the probability that a new node exhibits a certain feature depends on the ability (represented by some random fitness parameter) of the previous nodes possessing that feature in transmitting it. It is worthwhile to underline that in our model, as in the standard Indian Buffet process, the collection of features is potentially unbounded. Thus, we do not need

to specify a maximum number of latent features *a priori*.

The proposed approach is a simple and natural integration of various ideas: it is a feature-based model – like the original Miller-Griffiths-Jordan model [136] – and therefore it can represent well any network whose relationships are based on the features of its nodes; the features are ruled by a process of Indian-Buffet type [80] allowing them to grow boundlessly in time and avoiding model selection problems. On top of it, we considered different abilities from each node in propagating its own features and acquiring links, importing the notion of *competition dynamics* into the feature-based realm. It is the first time, to our knowledge, that these aspects are combined in a unique model. Such a model is at the same time very expressive and easy-to-tune, due to the natural interpretation of the involved parameters. Finally, we have provided some theoretical, as well experimental, results regarding the power-law behavior of the model and the estimation of the parameters, showing how the proposed model for the feature generation naturally leads to a complex network model.

As we will see in the third, experimental, part of this thesis (section 7.3), the comparison with real datasets can cast more light on this model: in fact, this model seems to be able to produce quite realistic feature matrices while at the same time capturing most local and global properties (e.g., degree distributions, connectivity and distance distributions) real networks exhibit.

Part II

Mining

Contents

Chapter 5

Inferring feature-feature interaction

5.1 Introduction

In our model, a special role is played by the feature-feature matrix $\mathbf{W}$. In Chapter 3, we highlighted how this matrix can express various kinds of interplay between features and links; we mentioned that we usually consider it as *latent*: an unobservable element of our model, that can compactly explain the links we are observing.

The question is therefore the following: assuming to know L and $\mathbf{Z}$, how can we reconstruct a plausible $\mathbf{W}$? That is, knowing the links of our network and the features every node bears, how can we estimate how features interact with each other?

This question has a lot of practical implications. In a semantic graph endowed with categories, $W_{h,k}$ describes how two categories h, k relate to each other: it summarizes if they interact positively, negatively and how much; it can be therefore used for measuring their semantic connection. In a linguistic graph (maybe obtained from a large corpus of text) linking words used as subjects to those used as objects for a certain verb, $\mathbf{W}$ describes the semantic areas this verb can connect. In a scientific network (e.g., a citation network) where features are academic fields, the set $S_k = \{h|W_{h,k} > 0\}$ contains the fields for which the field k is useful, and so forth.

Furthermore, if we have complete knowledge of $\mathbf{Z}$ and an estimate of $\mathbf{W}$, we can assign to every pair of nodes a score representing how much our model expects that pair to form a link. Since in our model ϕ is monotonically increasing, in fact, if we know $\mathbf{W}$ and $\mathbf{Z}$ we do not need any further assumption on it: the higher the score, the more likely the presence of a link is. Such a score can therefore be used in practice in many ways. For example, in section 8.5 we will show how its opposite can be used to catch serendipity in links among a large corpus. The accuracy of these predictions will give a measure of how well a set of features can explain a graph, as we will show in section 7.4.

Many other examples are possible; it is however important to note how many of these applications require to deal with a huge number of nodes and links. Experiments we will see in the third part of this work will run on graphs with millions (or tens of millions!) of nodes, and hundreds of millions to billions of links. Operating at this scale demands new techniques; we will see in section 5.2 how many of the existing techniques are not able to scale up to this size.

A first idea, that we will describe in section 5.3, is to just estimate the probability of a link from the category pairs we see in the data. We will derive formally this approach, showing that it can be ascribed to the family of Naïve Bayes learning. In particular, we will see that it requires independence assumptions that are particularly unrealistic in practice in many cases. For example, consider the semantic link between the entity *"Ronald Reagan"* and the 1954 Western film *"Cattle Queen of Montana"* ; such an approach will increment, in consequence of this link, the element of $\mathbf{W}$ corresponding to $(films,\ U.S.\ presidents)$, regardless of the fact that this link could be already well explained by $(films,\ actors)$.

For this reason, we will need to simplify our model by setting ϕ as the step function $\phi(x) = \chi_{(\vartheta,\infty)}(x)$, and thus making our model deterministic. As we will describe in section 5.4, this fact allow us to see our model equation as the prediction rule of a perceptron – a very simple shallow neural network – and in the end to develop a more sophisticated approach based on online machine learning. What we will do is to see L (the links in the graph) as partially unknown and to be learned; we will show that in this way the internal state of the perceptron will tend to $\mathbf{W}$. This approach, that we will call *Llama* – Learning LAtent MAtrix – will overcome the naïve assumptions of the previous model; in the third part of this thesis we will see how it can be applied to the very large networks we are interested in.

In section 5.5 we will test this approach to our model, by simulating a graph with the techniques described in the previous chapter and then observing how this way of reconstructing $\mathbf{W}$ behaves.

Finally, we would like to point out how a similar result for the construction of $\mathbf{W}$ could be obtained by assuming a partial knowledge of $\mathbf{Z}$ instead of L, and learning from it. We will explore this possibility in the next chapter.

Section 5.4.2 (and a part of section 5.2) contains contents published in a conference paper at *"Web Science 2016"* [8].

5.2 Related works

The goal of this chapter is to develop methods able to recover the latent feature-feature matrix, knowing links and features of each node, in order to apply them to the large networks available in real data (e.g., the web).

As we mentioned in section 2.3, this task has strong empirical foundations: a feature-feature matrix has been used to model the tracking of diseases, from sexual diseases [12] to respiratory infections [140]. In this area, they involve small datasets; there such matrices are defined as "Who Acquires Infection From Whom" (WAIFW) matrices; they have been empirically assessed in the field in various ways (surveys [93] and wearable sensors [98]).

The task we are defining ultimately falls back into the realm of latent variable models [68], since we are trying to explain a set of manifest variables – links and features – through a set of latent variables – the feature-feature interaction weight, expressed by the elements $W_{h,k}$ of the matrix $\mathbf{W}$. If, like in our case, manifest variables are categorical, they are usually called *latent structure models*, and have been studied as such by statisticians and social scientist since the 1950's. Lazarsfeld started studying the statistics behind these models in an effort to explain peo-

ple answers to psychological questions (in first works [175], answers from World War II soldiers) through quantifiable, latent traits [123]. These techniques were improved by later studies [84, 92]; however, these techniques—since conceived for traditional social studies—were designed for small groups; the use cases described there does not usually involve more than a hundred of nodes. We require our techniques to work with millions of nodes, and hundreds of millions of links. We also point out even if some of the approaches presented in section 2.3 could be able to represent a matrix describing all feature-feature pairs ($\mathbf{W}$), they were not able to handle data this large.

In recent years, computer science modeled relationships between hidden and visible random variables through probabilistic graphical models [26], and especially Bayesian networks [13]. Such models have been applied successfully to a plethora of different areas, since they allow for parameter learning, usually through the application of the Expectation-Maximization algorithm [60]. A very successful case for graphical models has been Latent Dirichlet Allocation (LDA) [28]. The problem we are defining – explaining links through features of each node – can be cast as an instance of it. Previous literature has treated linked document corpora, where features are the words contained in each document [48, 127]. In previous works, authors build a link prediction model obtained from LDA, that considers both links and features of each node. However, the largest graphs considered in these works have about 10^3 nodes (with $\sim 10^4$ possible features), and they do not provide running time. Henderson *et al.* [91] developed an LDA approach explicitly tailored for "large graphs" — but without any external feature information for nodes; the largest graph they considered has about 10^4 nodes and 10^5 links, for which they report a running time of $45 - 60$ minutes. The algorithm we are going to propose in section 5.4.2, although simpler, requires 9 minutes to run on a graph three orders of magnitude larger (about 10^6 nodes and 10^8 links).

5.3 A naïve approach

Let us describe a naive approach to construe the latent feature-feature matrix $\mathbf{W}$. We shall use a naive Bayes technique [26], estimating the probability of existence of a link through maximum likelihood and assuming independence between features; that is, we are going to assume that the events $\{Z_{i,h} = 1\}$ and $\{Z_{i,k} = 1\}$ are independent $\forall h, k$.

Let us introduce the following notations, that we will use throughout this chapter:

- Let $N_k \subseteq N$ be the set of nodes with the feature k, i.e. $N_k = \{i \in N | Z_{i,k} = 1\}$.
- Reversely, we will call F_i the set of features exhibited by a node i, that is $F_i = \{k \in F | Z_{i,k} = 1\}$.
- Let us also use $\mathcal{Z}_{i,k}$ to denote the event $\{Z_{i,k} = 1\}$.

Now, fixing two features h and k, let us consider the probability $p_{h,k}$ that there is a link between two arbitrary nodes with those features, such as $i \in N_h$ and $j \in N_k$:

$$p_{h,k} := \mathbb{P}\Big((i,j) \in L \Big| \mathcal{Z}_{i,h} \cap \mathcal{Z}_{j,k}\Big)$$

This quantity can be estimated as the fraction of pairs (i, j) such that both $\mathcal{Z}_{i,h}$ and $\mathcal{Z}_{j,k}$ are true, that happen to be links. In other words,

$$p_{h,k} = \frac{|\{N_h \times N_k) \cap L\}|}{|N_h| \cdot |N_k|}$$

For a specific pair of nodes (i, j), the probability of the presence of a link under the full knowledge of $\mathbf{Z}$ is given by

$$\mathbb{P}\Big((i,j) \in L \Big| \big(\bigcap_{h \in F_i} \mathcal{Z}_{i,h}\big) \cap \big(\bigcap_{h \in F_j} \mathcal{Z}_{j,h}\big)\Big)$$

Let us naively assume that $\mathcal{Z}_{i,h}$ and $\mathcal{Z}_{j,k}$ are independent for all i, j, h, k with $i \neq j$ and $h \neq k$; we also assume the they are independent even under the knowledge that $(i, j) \in L$. Then, under these naïve independence assumptions, the last probability can be expressed as

$$\prod_{h \in F_i} \prod_{k \in F_j} \mathbb{P}\Big((i,j) \in L \Big| \mathcal{Z}_{i,h} \cap \mathcal{Z}_{j,k}\Big) = \prod_{h \in F_i} \prod_{k \in F_j} p_{h,k}$$

Next, we will estimate our matrix $\mathbf{W}$ as:

$$W_{h,k} = \log \frac{|\{(N_h \times N_k) \cap L\}|}{|N_h| \cdot |N_k|} \tag{5.1}$$

Finally, let us check that such a matrix is correct. Taking back the definition of our model (3.1) and inserting such a $\mathbf{W}$, we obtain, in virtue of the previous definitions:

$$\begin{aligned}
\mathbb{P}\Big((i,j) \in L\Big) &= \phi\Big(\sum_{h \in F_i} \sum_{k \in F_j} W_{h,k}\Big) = \phi\Big(\log \prod_{h \in F_i} \prod_{k \in F_j} \frac{|\{(N_h \times N_k) \cap L\}|}{|N_h| \cdot |N_k|}\Big) = \\
&= \phi\Big(\log \prod_{h \in F_i} \prod_{k \in F_j} p_{h,k}\Big) = \phi\Big(\log \mathbb{P}\big((i,j) \in L \big| \big(\bigcap_{h \in F_i} \mathcal{Z}_{i,h}\big) \cap \big(\bigcap_{h \in F_j} \mathcal{Z}_{j,h}\big)\big)\Big) = \\
&= \phi\Big(\log \mathbb{P}\big((i,j) \in L \big| \mathbf{Z}\big)\Big)
\end{aligned} \tag{5.2}$$

Confirming that, for a certain choice of ϕ (namely[1], $\phi(x) = \min(1, e^x)$) and under the previously mentioned independence assumptions, this estimate of $\mathbf{W}$ is correct for our model.

5.4 A learning approach

As we said before, the naïve approach has its drawbacks. Its independence assumptions are not tenable in practice. Because of them, it sees a link as explained in equal parts among all the involved features. As we said in the introduction, consider a semantic link between two entities: the entity for "Ronald Reagan", and the one for *"Cattle Queen of Montana"*, a 1954 Western film starring Ronald Reagan. The naïve approach will count the link between the two as a member

[1] We need the min in this formula to respect our assumption that ϕ only has values in $[0, 1]$. However, it does not change anything in practice, since in (5.2) the argument of the logarithm is a probability, and therefore it is forced to be in $[0, 1]$.

of the set $\{(N_{presidents} \times N_{movies}) \cap L\}$, and thus incrementing the corresponding value in the feature-feature matrix, namely $W_{presidents,\ movies}$. We would rather have an algorithm that is able to recognize the fact that this link is probably already well explained by the (supposedly already high-valued) matrix element $W_{actors,\ movies}$; and, since it is already sufficiently explained by that, not enforcing a false association between politicians and western movies. In other words, we want our algorithm to perceive if some feature is already explaining a link, or – if it does not – to update its estimate of $\mathbf{W}$. In this perspective, how can we properly cast our problem to learning theory?

A deterministic model. In order to obtain this result, let us simplify our framework by fixing a simple ϕ, the step function $\chi_{(0,\infty)}$, that is:

$$\phi(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases}$$

This can be also seen as a sigmoid (see (3.2)) whose parameters are $\vartheta = 0$ and $K \to \infty$. As such, we have studied its effects in our model through simulations in section 4.5 (see for example fig. 4.6); there we found that, even if it produces a more disconnected network, its degree distribution tends even more to the classic power law of complex networks. Such experiments suggest that this choice of ϕ is reasonable for our model.

It is important to note that this choice will make our model fully deterministic. In other words, given the complete knowledge of $\mathbf{Z}$ and $\mathbf{W}$, the model *per se* does not allow for any missing or wrong links. For this reason, with this model we can not measure the *likelihood* of a real network; instead, we will just separate its links in *explained* and *unexplained* by the model with respect to a certain set of feature F.

A decision rule. Now, by using this deterministic activation function, the equation of our model (3.1) becomes:

$$(i,j) \in L \iff \sum_h \sum_k Z_{i,h} Z_{j,k} W_{h,k} > 0 \tag{5.3}$$

Let us indicate the i-th row of $\mathbf{Z}$ with $\mathbf{z}_i$, the outer product with $\otimes$ and with $\circ$ the Hadamard product. Then, we can alternatively write the above rule in one of these two equivalent forms:

$$(i,j) \in L \iff \mathbf{z}_i^T \mathbf{W} \mathbf{z}_j > 0$$

or

$$(i,j) \in L \iff \sum_{h,k} \Big[\big(\mathbf{z}_i \otimes \mathbf{z}_j \big) \circ \mathbf{W} \Big]_{h,k} > 0 \tag{5.4}$$

5.4.1 A perceptron

This equation – shown pictorially in Figure 5.1 – reveals itself to be a special case of the *decision rule* of a simple neural network classifier, the perceptron [158]. The idea here is that by learning

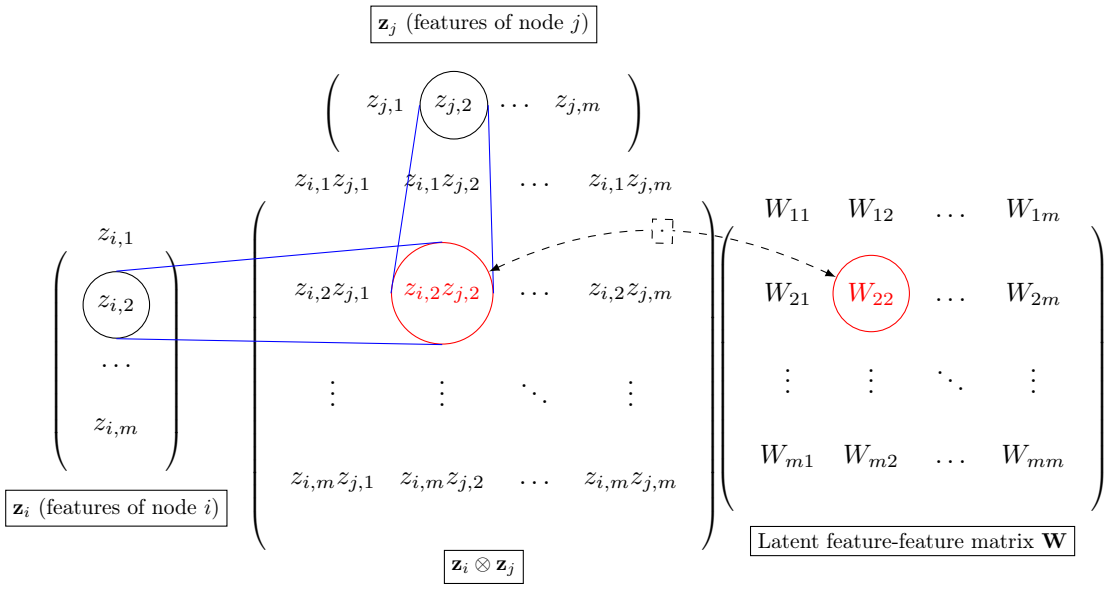


Figure 5.1: Relationship between the binary vectors $\mathbf{z}_i$ and $\mathbf{z}_j$, representing the features in node i and j, their outer product $\mathbf{z}_i \otimes \mathbf{z}_j$, and the corresponding feature-feature latent matrix $\mathbf{W}$. The prediction of the perceptron – and of the deterministic version of our model – will be given by $\operatorname{sgn}\big((\mathbf{z}_i \otimes \mathbf{z}_j) \circ \mathbf{W}\big)$.

how to separate links from non-links (in fact a form of link prediction, see page 64), the classifier must infer $\mathbf{W}$ as its internal state.

Let us define the standard perceptron formally; its internal state $\mathbf{w} \in \mathbb{R}^n$ is initialized randomly, and then it operates like this on a sequence of pairs numbered by $i \in T$ with $T = \{0, 1, \ldots, t-1\}$:

1. observes an *example* such as $\mathbf{x}_i \in \mathbb{R}^n$;
2. emits a *prediction* $\hat{y}_i = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x}_i)$;
3. receives the *true label* $y_i \in \{-1, 1\}$;
4. if $y_i \neq \hat{y}_i$, it updates its internal state with $\mathbf{w} = \mathbf{w} + y_i \lambda \mathbf{x}_i$, where $\lambda \in (0, 1]$ is a parameter called *learning rate*.

The key point here is that the decision rule for emitting a prediction can be cast to be fundamentally the same as in our model. Specifically, if we view the latent feature-feature matrix $\mathbf{W}$ as a m^2-length vector, and we do the same for $\mathbf{z}_i \otimes \mathbf{z}_j$, then we can see that the decision rule $\operatorname{sgn}(\mathbf{w} \cdot \mathbf{x}_i) = 1$ corresponds to (5.4), if we set $\mathbf{W}$ as the vector $\mathbf{w}$ and $\mathbf{z}_i \otimes \mathbf{z}_j$ as the example $\mathbf{x}$.

Namely, in our case, an *example* for the perceptron will be a pair of nodes (i, j), represented not by a vector but by the $m \times m$ matrix $\mathbf{z}_i \otimes \mathbf{z}_j$: this is a matrix where the element $[\mathbf{z}_i \otimes \mathbf{z}_j]_{h,k}$ is 1 iff the first node has feature h and the second k. This trick is sometimes called the *outer product kernel*: we are transforming a pair of vectors of size $2m$ into a high-dimensional representation of size m^2. This $m \times m$-matrix in fact can be alternatively thought of as a vector of size m^2,

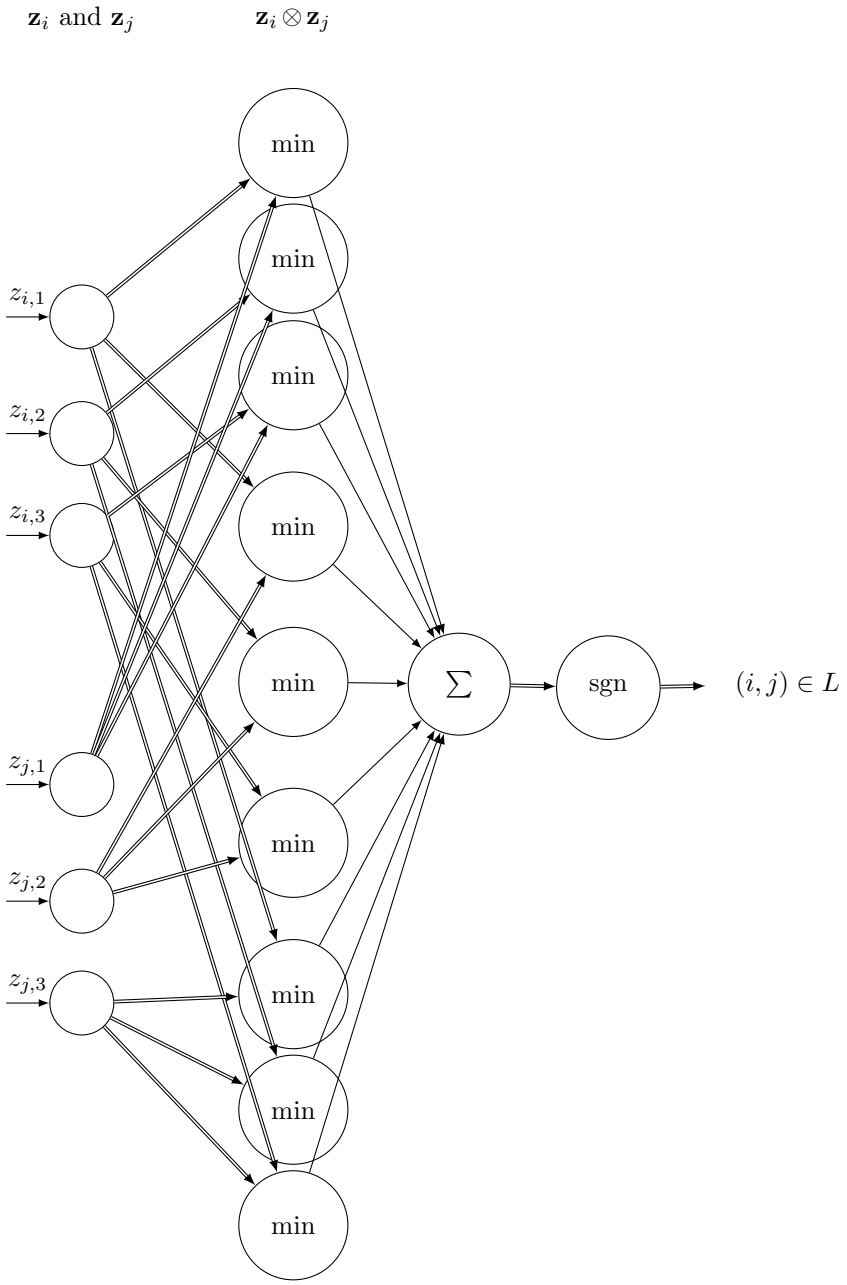


Figure 5.2: A neural-network view of our perceptron, for the case of $m = 3$ features. We indicate fixed weights with double lines, with min those nodes activating only iff both input nodes are active (that is, the min of their inputs), and with sgn the sign function. The only non-fixed weights (learned by the perceptron update rule) are those from the $\mathbf{z}_i \otimes \mathbf{z}_j$ layer to the $\sum$ neuron: they correspond to the matrix $\mathbf{W}$ appearing in our model.

allowing us to use them as training examples for the perceptron, where the label is $y = 1$ iff $(i, j) \in L$, and $y = -1$ otherwise. The learned vector $\mathbf{w}$ will be, if seen as a matrix, the desired $\mathbf{W}$ appearing in (5.3), as we are going to analyze next.

To recap, the perceptron we are going to use operates like this, on a sequence $T \subset N \times N$:

1. observes a pair of nodes $(i, j) \in T$, through their binary feature vectors $(\mathbf{z}_i,\ \mathbf{z}_j)$;
2. computes a prediction on whether they form a link, according to (5.4); that is, $\hat{y}_{i,j} = \text{sgn}(\mathbf{z}_i^T \mathbf{W} \mathbf{z}_j)$
3. receives the ground-truth: $y_{i,j} = 1$ iff $(i, j) \in L$, or $y_{i,j} = -1$ otherwise;
4. if the prediction was wrong, it updates its internal state by adding (if the pair was a link) or subtracting (if it was not) to $\mathbf{W}$ the quantity $\lambda(\mathbf{z}_i \otimes \mathbf{z}_j)$.

In doing this, we are using m^2 features, in fact a kernel projection of a space of dimension $2m$ onto the larger space of size m^2. Similarly the weight vector to be learned has size m^2. Positive examples are those that correspond to existing links. We can view this as a shallow, simple neural network, as pictured in Figure 5.2.

Connection with link prediction. As we anticipated at the beginning of this section, the technique we are going to employ is in fact a form of *link prediction*. In fact, we wish to highlight that, in the light of (3.1), whenever our model fits a graph, then the problem of estimating its latent feature-feature matrix is fundamentally equivalent to the problem of assigning a probability to every link knowing each node features. As we saw in this section, however, by fixing a parameter of our model – specifically, by setting the function ϕ to a step function – we can trace a formal link between predicting links with the perceptron algorithm and estimating $\mathbf{W}$ within our model.

This connection with link prediction grant us three gifts:

1. It give us an algorithm – actually, a family of algorithms, as we will see in section 5.4.2 – able to estimate $\mathbf{W}$ by training them to predict links.
2. It allow us to employ previous analysis done for the perceptron in order to have formal guarantees about the ability of such algorithms to estimate the feature-feature matrix, as we will show in the remaining part of this section.
3. It permits us to adopt a link prediction test setting to evaluate our algorithms: we can evaluate not only how close the estimated matrix is to the original one (a difficult evaluation problem *per se*, e.g., since $\mathbf{W}$ could be scaled), but instead apply the standard machinery developed to correctly evaluate link prediction algorithms, as we will do in section 5.5.

Because of these considerations, we believe that correctly connecting the concepts of feature-rich graph models and link prediction through online machine learning algorithms was instrumental in our understanding of the former.

Interpretation of the error bound. The perceptron algorithm, since its conception in 1957 by Frank Rosenblatt (who built it as an *ad-hoc* machine, where weights were potentiometers, updated by electric motors!) has been subject to many analysis. In particular, some bounds on its number of errors were found. Let us take into consideration the bound discussed by Cesa-Bianchi and Lugosi [46, p. 337], in its standard case.[2] Considering our definitions, it is just a matter of rewriting it to state the following bound for the number of errors $M = |\{(i,j) \in T \; s.t. \; \hat{y}_{i,j} \neq y_{i,j}\}|$:

$$M \leq \inf_{\mathbf{U} \in \mathbb{R}^{m \times m}} \left(H(\mathbf{U}) + \left(R\|\mathbf{U}\| \right)^2 + R\|\mathbf{U}\| \sqrt{H(\mathbf{U})} \right) \tag{5.5}$$

Where

- $H(\mathbf{U}) = \sum_{(i,j) \in T} \max\left(0, 1 - \mathbf{z}_i^T \mathbf{U} \mathbf{z}_j\right)$ is the sum of the *hinge losses*.
- $R = \max_{(i,j) \in T} \|\mathbf{z}_i \otimes \mathbf{z}_j\|$ is the *radius* of the examples.

Let us interpret this bound for our case.

First, the matrix $\mathbf{U}$ can be thought as an *ideal* latent feature-feature matrix $\mathbf{W}$. In particular, we would like it (in order for this bound to be low) to be a good $\mathbf{W}$ with the lowest possible norm (considering that the predictions of the model according to (5.4) remain the same if we multiply $\mathbf{W}$ by a positive constant). The appearance of the norm $\|\mathbf{U}\|$ in this bound is explained by the fact that a model with a large norm is – apart from scaling – more complex: e.g., a very sparse $\mathbf{W}$ – meaning that only a few pairs of features interact – will have a very low norm.

$H(\mathbf{U})$ is a measure of how good is the model on these instances. For example, if the sequence of input pairs T is set to be equal to the links L, then let us consider the set:

$$E_{\mathbf{U}} = \left\{ (i,j) \in L \;\middle|\; \mathbf{z}_i^T \mathbf{U} \mathbf{z}_j < 1 \right\}.$$

Those are the links for which the model, with this ideal $\mathbf{U}$, cannot obtain a value in (5.4) (that should be positive for links) that is at least 1. Considering again that we can multiply $\mathbf{U}$ by a positive constant, those are the links that cannot be *fully expected* by our model. $H(\mathbf{U})$ becomes then:

$$H(\mathbf{U}) = \left| E_{\mathbf{U}} \right| - \sum_{(i,j) \in E_{\mathbf{U}}} \mathbf{z}_i^T \mathbf{U} \mathbf{z}_j$$

That is, we are counting how many links are in $E_{\mathbf{U}}$ and then adding *how far* was our model from expecting these links. Specifically, $\mathbf{z}_i^T \mathbf{U} \mathbf{z}_j$ will represent a small "discount" for links that were predicted correctly but with a small (i.e., < 1) value; instead, for the links that were *not* predicted as such, it will represent a cost, depending on how far from 0 they were. We can therefore see how $H(\mathbf{U})$ is a measure of how well our model could fit (in the best case) this particular feature-rich graph.

Finally, R^2 can be rewritten as

$$R^2 = \max_{(i,j) \in T} \sum_h \sum_k z_{i,h} z_{j,k} = \max_{(i,j) \in T} |F_i| \cdot |F_j|$$

[2] In fact, we are considering only Euclidean norm and standard hinge loss for simplicity.

In other words, it measures *how many pairs of features* we need to consider in our set of examples. Again, if $T = L$, this is the number of possible pairs among the features of each predecessor and each successor. This number is limited by the number of total features m – i.e., $R \leq m$: this means that the bound is smaller if we need less features to explain the graph. Moreover, it is smaller if there is little overlapping of features (i.e., if $\max_{i \in N} |F_i|$ is low). This bound tells us that they are all factors affecting the number of errors of this algorithm.

Finally, in case the feature-rich graph can be perfectly explained by a latent feature-feature matrix $\mathbf{W}$, according to our deterministic model, we have $H(\mathbf{W}) = 0$ and so the bound becomes $M < \big(R\|\mathbf{W}\|\big)^2$. As noted by Cesa-Bianchi and Lugosi [46] in the general case, this bound is the same result proved in 1961 for the perceptron [7] (the convergence theorem); in our case, it tells us that if such a perfect $\mathbf{W}$ exists, the perceptron will converge to it.

5.4.2 A passive-aggressive algorithm

Online learning. In general, what we did was to recast our goal in the framework of online binary classification. Binary classification, in fact, is a well-known problem in supervised machine learning. *Online* classification simplifies this problem by assuming each example is presented in a sequential fashion; the classifier operates by repeating this cycle:

1. observes an example;
2. tries to predict its label;
3. receives the true label
4. updates its internal state consequentially, and moves on to the next example.

An online learning algorithm, generally, needs a constant amount of memory with respect to the number of examples, which allows to employ these algorithms in a situation where a very large set of voluminous input data is available. Many surveys are available in literature [44].

A well-known type of online learning algorithms are the so-called perceptron-like algorithms. They all share the same traits of the perceptron: each example must be a vector $\mathbf{x}_i \in \mathbb{R}^n$; the internal state of the classifier is also represented by a vector $\mathbf{w} \in \mathbb{R}^n$; the predicted label is $y_i = sign(\mathbf{w} \cdot \mathbf{x}_i)$. The algorithms differ on how $\mathbf{w}$ is built. However, since their decision rule stays the same, they all lead back to the decision rule of our model (5.4). Therefore, it allow us to employ *any* perceptron-like algorithm for our purpose.

Perceptron-like algorithms (for example, ALMA [77] and Passive-Aggressive [57]) are usually simple to implement, provide tight theoretical bounds, and have been proved to be fast and accurate in practice.

A Passive-Aggressive algorithm. Among the existing perceptron-like online classification frameworks, we will heavily employ the well-known Passive-Aggressive classifier, characterized by being extremely fast, simple to implement, and shown by many experiments [42, 138] to perform well on real data.

Algorithm 3 LLAMA, the passive-aggressive algorithm to build the latent feature-feature matrix $\mathbf{W}$.

INPUT:
The graph $G = (N, L)$, with $L \subseteq N \times N$
Features $F_i \subseteq F$ for each node $i \in N$
A parameter $K > 0$
OUTPUT:
The feature-feature latent matrix $\mathbf{W}$

DESCRIPTION:

1. $\mathbf{W} \leftarrow \mathbf{0}$
2. Let $(i_1, j_1), \ldots, (i_T, j_T)$ be a sequence of elements of $N \times N$.
3. For $t = 1, \ldots, T$
   (a) $\rho \leftarrow |F_{i_t}| \cdot |F_{j_i}|$
   (b) $\mu \leftarrow \sum_{h \in F_{i_t}} \sum_{k \in F_{j_t}} W_{h,k}$
   (c) **If** $(i_t, j_t) \in L$
   $\delta \leftarrow \min(K, \max(0, \rho(1 - \mu)))$
   else
   $\delta \leftarrow -\min(K, \max(0, \rho(1 + \mu)))$
   (d) For each $h \in F_{i_t}$, $k \in F_{j_t}$:
   $W_{h,k} \leftarrow W_{h,k} + \delta$

Le us now describe the well-known Passive-Aggressive algorithm [57], while showing how to cast this algorithm for our case. To do this let us consider a sequence of pairs of nodes

$$(i_1, j_1), \ldots, (i_T, j_T) \in N \times N$$

(to be defined later). Define a sequence of matrices $\mathbf{W}^0, \ldots, \mathbf{W}^T$ and of slack variables $\xi_1, \ldots, \xi_T \geq 0$ as follows:

- $\mathbf{W}^0 = \mathbf{0}$
- $\mathbf{W}^{t+1}$ is a matrix minimizing $\|\mathbf{W}^{t+1} - \mathbf{W}^t\| + K\xi_{t+1}$ subject to the constraint that

$$y_{i_t, j_t} \cdot \sum_{h \in F_{i_t}} \sum_{k \in F_{j_t}} W^{t+1}_{h,k} \geq 1 - \xi_{t+1}, \tag{5.6}$$

where, as before

$$y_{i_t, j_t} = \begin{cases} -1 & \text{if } (i, j) \notin L \\ 1 & \text{if } (i, j) \in L \end{cases},$$

$\| - \|$ denotes the Frobenius norm and K is an optimization parameter determining the

amount of aggressiveness.

The intuition behind the above-described optimization problem, as described in the original work where the Passive-Aggressive algorithm [57] was presented, is the following:

- the left-hand-side of the inequality (5.6) is positive iff $\mathbf{W}^{t+1}$ correctly predicts the presence/absence of the link (i_t, j_t); its absolute value can be thought of as the confidence of the prediction;
- we would like the confidence to be at least 1, but allow for some error (embodied in the slack variable ξ_{t+1});
- the cost function of the optimization problem tries to keep as much memory of the previous optimization steps as possible (minimizing the difference with the previous iterate), and at the same time to minimize the error contained in the slack variable.

By merging the Passive-Aggressive solution to this problem with our aforementioned framework, we obtain the algorithm described in Alg. 3. We will refer to this algorithm as *Llama*: that is, *Learning LAtent MAtrix*.

We wish to remark how the obtained algorithm, despite being non-naïve, is amazingly fast and scale up very well with the size of data; e.g., as we will see in section 8.4, to run on $2.2 \cdot 10^8$ node pairs, considering a set of $2 \cdot 10^4$ features, it took 9 minutes on an Intel Xeon CPU with 2.40GHz.

Normalization. For perceptron-like algorithms, normalizing example vectors (in our case, the matrix $\mathbf{z}_i \otimes \mathbf{z}_j$) often gives better results in practice [58]. This is equivalent to using the ℓ^2-row-normalized version of our model, presented in section 3.2.4. The assumption behind that model is in fact that nodes with less features provide a stronger signal for the small set of features they have; nodes with many features bear less information about those features. To use this normalization in the algorithm, we just need to change the point (c) in Alg. 3 to:

(c) **If** $(i_t, j_t) \in L$: (5.7)
$\delta \leftarrow \frac{1}{\sqrt{\rho}} \min(K, \max(0, 1 - \frac{\mu}{\sqrt{\rho}}))$
else:
$\delta \leftarrow -\frac{1}{\sqrt{\rho}} \min(K, \max(0, 1 + \frac{\mu}{\sqrt{\rho}}))$

Sequence of pairs. Finally, let us discuss how to build the sequence of examples. We want $\mathbf{W}$ to be built through a single-pass online learning process, where we have all positive examples at our disposal (and they are in fact all included in the training sequence), but where negative examples cannot be all included, because they are too many and they would produce overfitting.

Both the Passive-Aggressive construction described above and the Perceptron algorithm depend crucially on the sequence of positive and negative examples $(i_1, j_1), \ldots, (i_T, j_T)$ that is taken as input. In particular, as discussed by Japkowicz *et al.* [104], it is critical that the number of

Avg. features per node			
	S	χ	exp
$\mathcal{B}$	3.82	5.51	5.66
$\mathcal{N}$	4.21	4.57	5.94

Avg. degree			
	S	χ	exp
$\mathcal{B}$	447.96	51.90	17.90
$\mathcal{N}$	91.05	27.56	15.53

Mean harmonic distance			
	S	χ	exp
$\mathcal{B}$	1.68	2.80	1.89
$\mathcal{N}$	1.95	4.28	1.82

Table 5.1: Properties of the synthetic feature-rich graphs. The 6 generated graphs are indicated according to the ϕ function used (S is the sigmoid, χ is the step function, and exp is the exponential) and to the distribution of the values of $\mathbf{W}$ (Bernoullian or normal). The listed properties are: the average size of the set of features owned by each node, the average degree, and the mean distance (computed with the harmonic mean to keep track of infinite distances).

negative and positive examples in the sequence is balanced. Taking this suggestion into account, we build the sequence as follows: nodes are enumerated (in arbitrary order), and for each node $i \in N$, all arcs of the form $(i, \bullet) \in L$ are put in the sequence, followed by an equal number of pairs of the form $(i, \bullet) \notin L$ (for those pairs, the destination nodes are chosen uniformly at random). Of course, if $t = |L|$ is the number of links, then $T = 2t$ and the sequence contains all the t links along with t non-links.

Obviously, there are other possible ways to define the sequence of examples and to select the subset of negative examples. However, we chose to adopt this technique – single pass on a balanced random sub-sample of pairs – in order to define and test our methodology with a single, natural and computationally efficient approach.

5.5 Simulations

In this section, we will test how these methods will perform on graphs generated by our model. In the third, experimental part of this work, we will see instead how the same algorithms behave on real-world data. However, we have seen in section 4.5 how our model is able to synthesize feature-rich networks with the same traits of typical real complex network (i.e., a power-law degree distribution). In section 7.3 we will see in particular how we can carefully generate a given real feature-rich graph; in fact, estimating the latent parameters of a real node-feature matrix $\mathbf{Z}$ and of the graph G, our model can produce feature-rich networks with similar properties to the given one. These experiments allow us to use these synthetic graphs as a test bed for the algorithms presented in this section.

5.5.1 Synthetic networks

To generate each network, we first produced its node-feature association $\mathbf{Z}$ as explained in section 4.3, using the same parameter values adopted there: $\alpha = 3$, $\beta = .5$, $c = 0$. Then, we fed these matrices $\mathbf{Z}$ to our model equation (3.1) to generate a number of graphs. For the graph model, we employed the following parameters:

- We used $n = 10\,000$ nodes.

- We applied three different types of activation function ϕ, to compare their results:
  1. The classic sigmoid function $S(x) = \left(e^{K(\vartheta - x)} + 1\right)^{-1}$, cited in section 3.2.1 as well as in [30] as the standard approach; we set $\vartheta = 0$ and $K = 5$. Please note that this function does *not* respect the assumptions for which we derived LLAMA, nor those of NAIVE.
  2. The step function $\chi_{(0,\infty)}$, characterizing the model behind LLAMA.
  3. The exp function, which characterizes the model behind NAIVE.
- The latent matrix $\mathbf{W}$ was generated assuming that its entries are i.i.d., with the following two value distributions:
  1. A generalized Bernoulli distribution $W_{h,k} \sim \mathcal{B}(p)$ that assumes the value 10 with probability $p = \frac{10}{m}$ and -1 with probability $1 - p$. This choice was determined through experiments, with the purpose of obtaining graphs with a realistic density independently from the number of features m.
  2. A normal distribution $W_{h,k} \sim \mathcal{N}(\mu, \sigma)$ with mean and variance identical to the previous Bernoulli distribution.
  3. We had to slightly modify these distributions for the case $\phi = \exp$, in order to obtain realistic graphs also in that case: in particular, when $\phi = \exp$ we used a Bernoulli distribution with value 1 with probability $p = \frac{1}{m}$ and -1 with probability $1 - p$, and a normal distribution that had the same mean and variance as the just-described Bernoulli distribution. In the following, when we say that $\phi = \exp$ we imply that we used one of these two modified distributions to generate $\mathbf{W}$.

With these 3 choices for ϕ and 2 choices for $\mathbf{W}$, we generated 6 different feature-rich graphs. The properties of these networks are summed up in Table 5.1. They represent a wide range of realistic traits we could actually observe in complex networks.

5.5.2 Evaluation of estimates

We will first direct our attention to measuring how the estimated $\mathbf{W}$ matrix is similar to the original $\mathbf{W}$ matrix used to produce the graph. In this section, we are going to generate 100 different graphs for each of the 6 graph families we described above. Each graph was generated with a different matrix $\mathbf{W}$. For each of these graphs, we run the Llama and the Naive algorithm, obtaining from each algorithm a different estimate $\widehat{\mathbf{W}}$ of the original $\mathbf{W}$, employing only the knowledge of the graph G and the binary node-feature matrix Z.

A first, unsophisticated method to perform this comparison would be just to measure the Mean Squared Error [29], i.e. $\text{MSE} = \frac{1}{n}\sum_{i,j}(\widehat{W_{i,j}} - W_{i,j})^2$. However, as mentioned in section 5.4.1, there is a whole family of matrices $\mathbf{W}$ that would be isomorphic for our model point of view; that is, two matrices that would generate, given two nodes i and j and their features $\mathbf{z}_i$ and $\mathbf{z}_j$, the same probabilities for a link $\mathbb{P}(i, j)$. An obvious example is that, when ϕ is the step function, $\mathbf{W}$ and $\lambda\mathbf{W}$ would be isomorphic for any $\lambda > 0$. To overcome this obstacle, a first idea

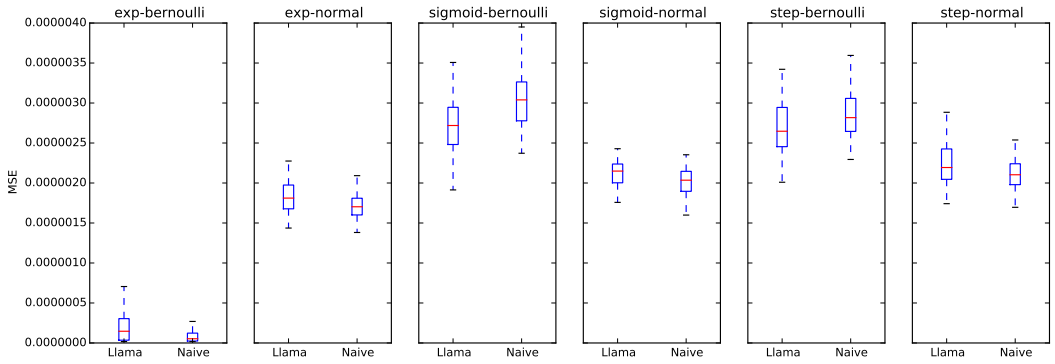


Figure 5.3: Mean squared error of the estimated matrix, with respect to the original, over 100 experiments for each graph type, with $\alpha = 3$.

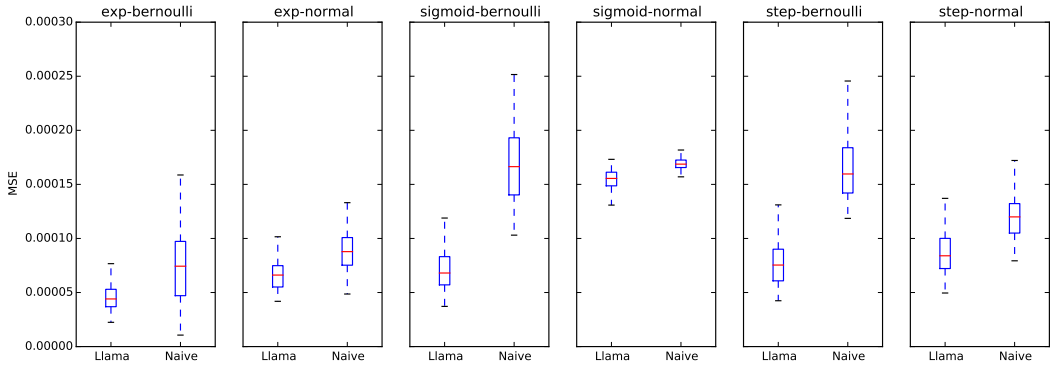


Figure 5.4: Mean squared error of the estimated matrix, with respect to the original, over 100 experiments for each graph type, with $\alpha = \frac{1}{2}$.

is to normalize the two matrices $\mathbf{W}$ and $\widehat{\mathbf{W}}$, and then computing the MSE. In other words, our measure will be $\text{NMSE} = \text{MSE}(\frac{\mathbf{W}}{\|\mathbf{W}\|}, \frac{\widehat{\mathbf{W}}}{\|\widehat{\mathbf{W}}\|})$.

Results with this measure are reported in Figure 5.3. Average results for both the Naive and the Llama algorithm are around $1.9 \cdot 10^{-6}$; for comparison, a random matrix would result in a mean error of $1 \cdot 10^{-5}$ (we do not report that in the plot since it would be out of scale). We can say, therefore, that the algorithms seem to extract some information about $\mathbf{W}$, but we do not get a clue of which one does a better job, nor a sense of how much these estimates could be useful in practice.

Some answers come out of a different question. We then asked ourselves, in fact, how the specifics of the node-feature association influence these results. That is: what would happen with a lower number of features? In order to obtain that, and recalling the role of the parameters delineated in section 4.4, we lowered the parameter α of the model from 3 to 0.5. In this way, the number of generated features goes from an average of 594 features (and a matrix of 353 597 elements on average) to an average of 99 features (and a matrix of 9 851 elements on average). The number of nodes is still 10 000. We report results in fig. 5.4. Here, the average result of a random estimate is $2.7 \cdot 10^{-4}$; the average result of Naive is $1.3 \cdot 10^{-4}$; the average result of Llama is $8.3 \cdot 10^{-5}$. The difference in absolute values from the previous case (e.g., the random

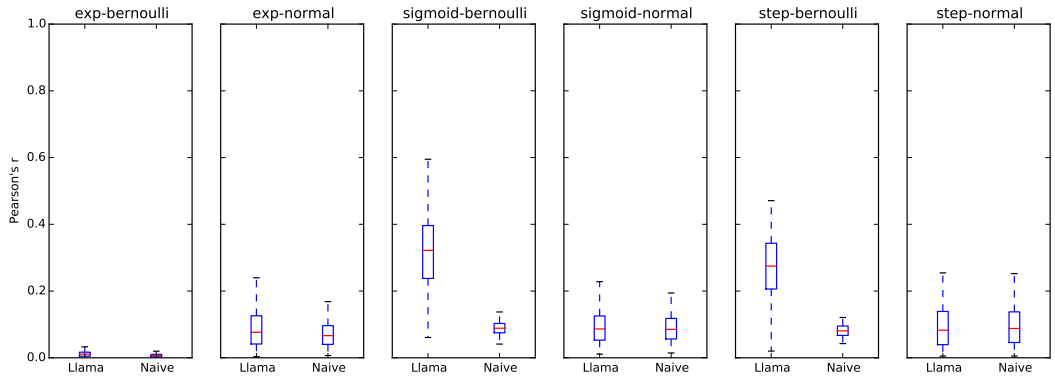


Figure 5.5: Pearson correlation between the original and the estimated matrix, over 100 experiments for each graph type, with $\alpha = 3$.

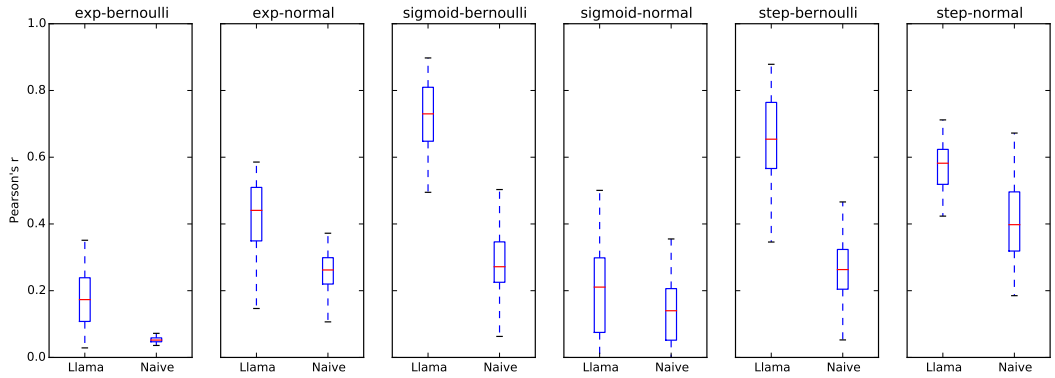


Figure 5.6: Pearson correlation between the original and the estimated matrix, over 100 experiments for each graph type, with $\alpha = \frac{1}{2}$.

estimate going from an error around 10^{-4} to 10^{-5}) is caused by the fact that we are dealing with matrices of different size, and this is affecting our measure.[3] However, the improvement of Llama over Naive is evident in the plots for the Bernoulli-generated $\mathbf{W}$, and especially for the sigmoid and step-like activation function. However, it is still difficult to compare these results with the previous case of $\alpha = 3$.

To overcome this problem, we turn to the usage of the Pearson's r as a measure of comparison between the two matrices. Pearson's r measures the linear dependence between two series of values (in our case, the sequences $W_{0,0}, \ldots, W_{m,m}$ and $\widehat{W}_{0,0}, \ldots, \widehat{W}_{m,m}$); in this way, the measure is indifferent to scaling, and in general to any linear transformation, solving in a better way the problem of isomorphic matrices. Pearson's r ranges from -1 (perfect inverse linear correlation) to 1 (perfect linear correlation).

We report results with Pearson's r in Figure 5.5 for the case of $\alpha = 3$ and in Figure 5.6 for the case of $\alpha = 0.5$. We remind that, while the previous measure is a measure of error, this is a measure of correlation (so the higher the values, the better). A random estimate would obtain,

[3] In particular, because we are dividing each matrix by its norm; if the matrix size is $m \times m$, then the norm of a random matrix with the same size will grow with m [47]; for this reason, our measure NMSE on a random matrix of size m will grow like m^{-2}.

by definition, a value of 0.

We can see that results are generally positive (the estimates correlates linearly with the original values), and that results for Llama are generally better than those of Naive. The worst case is the exponential activation function, that is very unnatural for the Llama algorithm. We can see that a lower number of features results again in generally better results. The best results appear to be in the sigmoid activation function and Bernoulli-generated $\mathbf{W}$, where the average Pearson's r for the case of $\alpha = 0.5$ is 0.72.

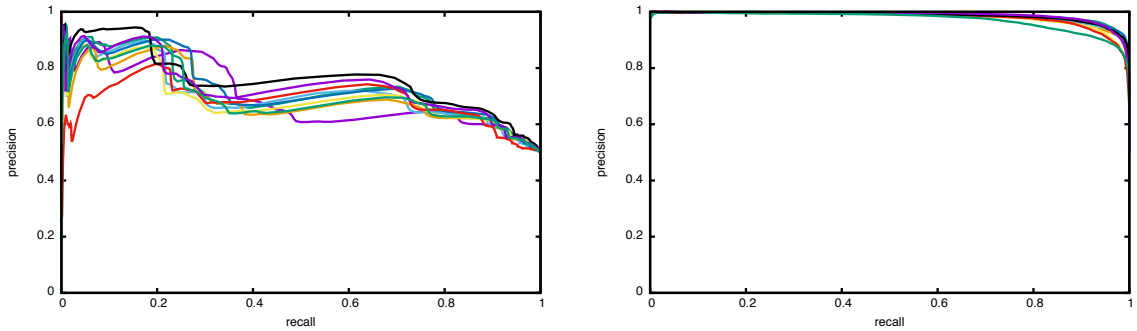
5.5.3 Evaluation of predictions

The previous measures of how good the estimates are, however, still do not tell us much about how useful is our model in practice. Our goal is in fact to find a feature-feature matrix for which our model works. For this reason, we will now devote our attention to measuring directly how accurate our methods are in terms of predicting if a node pair (i, j) forms a link, given their features. To keep this evaluation meaningful, our algorithms will not be allowed to see the whole graph: we will use the standard approach of 10-fold cross-validation; we will divide the set of nodes N in ten folds of equal size, and for each we used the other nine to train the algorithm and that fold only for testing the results.

Our evaluation closely resembles *link prediction*. There are of course some differences: first of all, we are using an external source of information (node features); second, our aim is to evaluate our model and our algorithms to find $\mathbf{W}$ *through* link prediction. That is, we are not interested in finding the best existing link predictor, but in measuring if our algorithms can correctly fit our model on a specific instance of feature-rich graph $G, \mathbf{X}$. However, we kept in consideration the guidelines for link prediction recently stated by Yang *et al.* [193]:

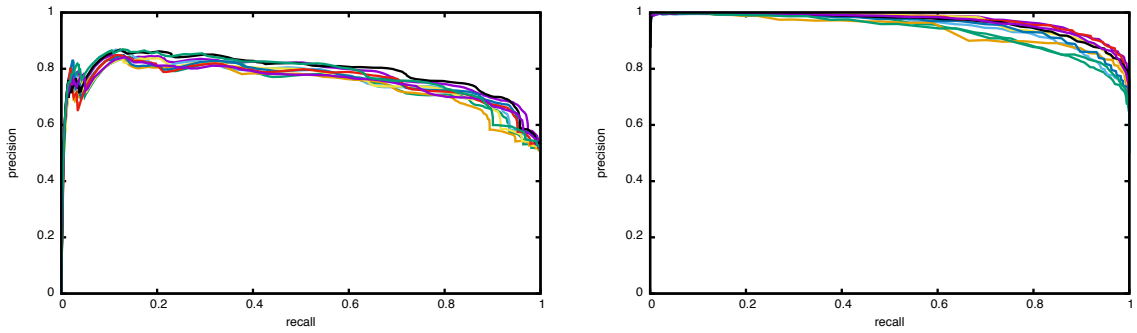
- We will evaluate how accurate are our algorithms in prediction by showing precision/recall curves, as they suggest; in fact, they find that other measures, such as the ROC curve, are heavily biased after undersampling negatives and can represent misleading results.
- To do this, we avoided fixed threshold between "link"/"not link". It is important, in fact, to evaluate the *scores* themselves; on the contrary, by choosing a threshold ϑ and then converting each score x to a binary event $x > \vartheta$ would make the comparison unfair. Instead, we used directly the score computed by our model (the argument of ϕ in (3.1), since the higher this score, the most probable that link should be). The use of precision/recall curves allow us to do this.
- We used the same testing set instances for all the tested algorithms.
- Although in our case it was necessary to undersample negatives (the total number of node pairs would be unmanageable), we took care of sampling uniformly the edges missing from the test network: for each node i, we chose other nodes j such that $i \neq j$ and $(i, j) \notin L$ until we had a number of non-successors equal to the number of successors, if possible.

Since our methods are not influenced by the distances of the considered node pairs (contrarily to standard link prediction approaches), we avoided to divide our results by geodesic distance.



Naïve Llama

Figure 5.7: Precision-recall curves for each fold in the network $\chi, \mathcal{B}$.



Naïve Llama

Figure 5.8: Precision-recall curves for each fold in the network $\chi, \mathcal{N}$.

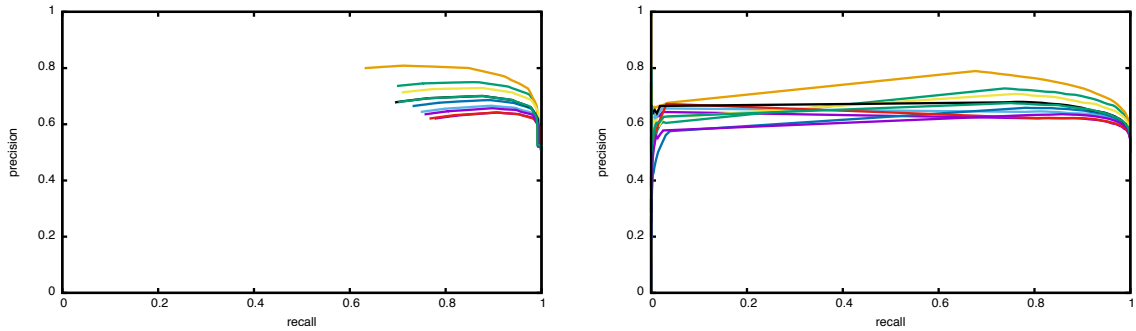
With these considerations in mind, we proceeded to evaluate through precision-recall curves our approaches. For each simulated network and for each fold, we gave the training graph to our algorithm and obtained an estimated matrix $\mathbf{W}$. This matrix is defined by (5.1) for the Naïve method and by Alg. 3 for *Llama*.[4] Each method then assigned its score to the testing set, according to our model (i.e., the argument of ϕ in (3.1)).

5.5.4 Experimental results

Results of these experiments are reported in figs. 5.7 to 5.12. Let us discuss these results, dividing them by the activation function employed to generate the graph.

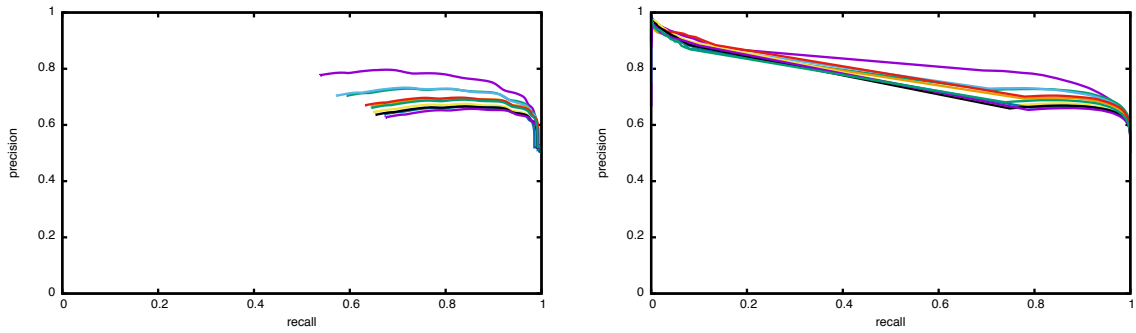
Step activation. First, let us consider the case of the network generated with a step activation function. Note that by using $\chi_{(0,\infty)}$ as the activation function we are making our model

[4]We also considered two variations: first, since the Naïve approach need a lower bound on $W_{h,k}$ if $L_{h,k} = \emptyset$ in the training graph, we also tried to adopt add-one smoothing [160], i.e. using $\log(x+1)$ in place of $\log(x)$. Second, we tried out also the normalized version of *Llama* expressed in (5.7), leaving the model unchanged. However, these approaches consistently got the same results of their respective original algorithms (despite not having the same formal guarantees), and thus we are not depicting their precision-recall curves.



Naïve Llama

Figure 5.9: Precision-recall curves for each fold in the network exp, $\mathcal{B}$.



Naïve Llama

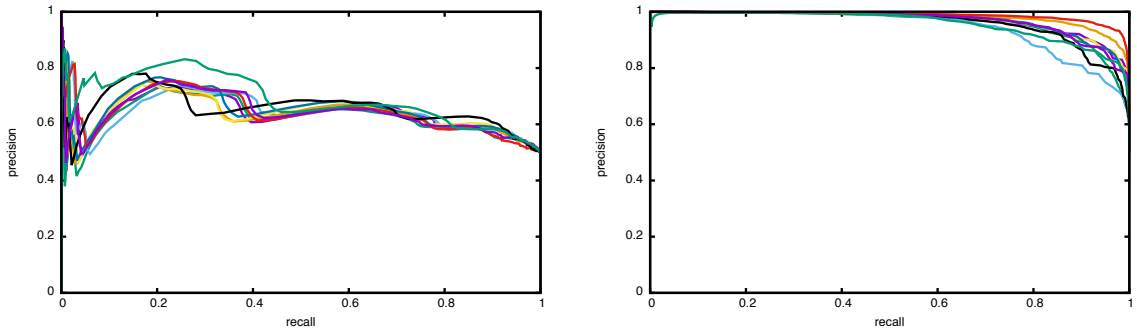
Figure 5.10: Precision-recall curves for each fold in the network exp, $\mathcal{N}$.

deterministic (a pair forms a link if and only if their score is positive) and thus simpler. Furthermore, it is precisely the model for which we have formal guarantees on the *Llama* algorithm performance. It is therefore natural that its results are remarkably good. For the Bernoullian distribution, *Llama* performance peaks at a 94% precision with a 94% recall; also, the precision can be 99.97% with a recall as high as 70%.

The Naïve algorithm, despite taking advantage of this clean activation function, behaves much worse than Llama; when recall is higher than 60% its performances quickly degrade toward a random classifier.

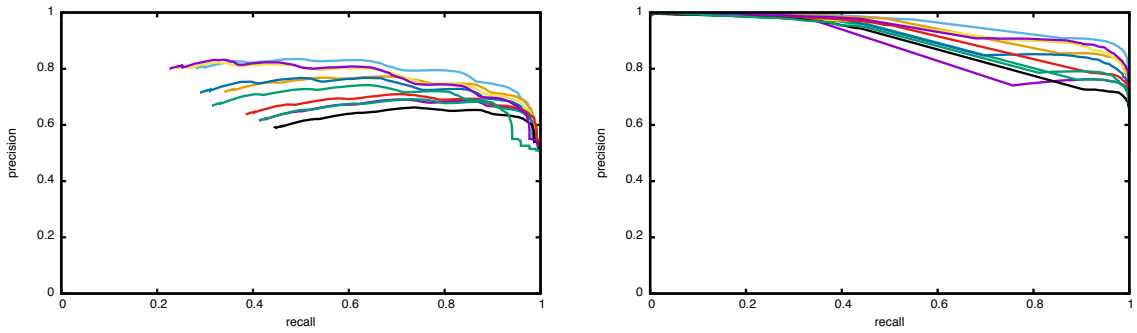
The same results can be also obtained with a normal distribution on the original $\mathbf{W}$. Here *Llama* reaches a 91%-91% precision-recall level, while the Naïve algorithm obtains results similar to before, although more stable.

Exponential activation. Then, let us look at the exponential activation function, for which we have formally derived the Naïve algorithm. Despite this, its results are still not satisfying: even its highest scores do not have a precision consistently higher than 65%, for both the Bernoullian and the Gaussian cases. Llama get its worst performance on this simulation, due to the furthest activation function from its natural one, and in the Bernoullian case its performance are around



Naïve Llama

Figure 5.11: Precision-recall curves for each fold in the network $S, \mathcal{B}$.



Naïve Llama

Figure 5.12: Precision-recall curves for each fold in the network $S, \mathcal{N}$.

a 65% precision for all recall levels. In the normally-distributed case, instead, with low levels of recall (10%) it is able to reach a precision of 90%, despite this unnatural activation function.

Sigmoid activation. Finally, let us look to the results obtained when the activation function is a sigmoid (see (3.2)) with $K = 5$. We highlight how this activation function is not the same of neither *Llama* (which assumes a step function) nor the Naïve algorithm (which assumes an exponential).

We report in fig. 5.11 the precision-recall curves for the network obtained with the standard sigmoid as ϕ and $W_{h,k}$ distributed on the discrete values $\{-1, 10\}$. We can see how the Naïve approach is able to get a reasonable precision only in the range of recall between 20% and 40%; instead, the non-naïve approach of Llama behaves extremely better, and it is in fact able to learn a model whose precision and recall lie both around 90%.

In fig. 5.12 we show the precision-recall curves when $W_{h,k}$ is distributed on all the real numbers, with a normal distribution. This $\mathbf{W}$ obviously introduces more noise and it is harder to retrace for our methods. The Naïve approach, in particular, has a high variance, and it is not able to give scores with a high precision at all (that is why we have not precision-recall values in the left corner). *Llama* instead is able to do that and it can obtain a precision of 99% at the

	$S, \mathcal{B}$	$S, \mathcal{N}$	$\chi, \mathcal{B}$	$\chi, \mathcal{N}$	$\exp, \mathcal{B}$	$\exp, \mathcal{N}$
Naïve	0.64 ± 0.01	0.70 ± 0.06	0.71 ± 0.02	0.76 ± 0.01	0.68 ± 0.05	0.67 ± 0.04
Llama	0.97 ± 0.01	0.92 ± 0.02	0.98 ± 0.00	0.96 ± 0.01	0.65 ± 0.03	0.78 ± 0.01

Table 5.2: Area under the precision-recall curve of the Naïve baseline and of the Llama algorithm. For each of the simulated graph, we report the mean and the standard deviation across the ten folds.

price of a low (20%) recall, or it can balance the two at around 80%.

Area under curve. To recap all the above results, we report in Table 5.2 the area under the precision-recall curve for each one of the curves pictured above. Following Yang *et al.* [193], we use this area as an overall measure of the goodness of our approach. We can see how the results of the Llama algorithm are consistently above 0.9 for both the step function and the sigmoid. For the exp function its results are less favorable; they are however comparable to the Naïve baseline in the Bernoulli case, and definitely above it for the Gaussian case.

5.6 Discussion

In this chapter, we put ourselves in the following scenario: assume to have complete knowledge of a node-feature association matrix – i.e., to know for every node, the features it exhibits; also, assume to have an (at least partial) knowledge of the links between these nodes. These objects corresponds to $\mathbf{Z}$ and G in our model (3.1). Our goal in this chapter was, given these elements, to find the latent interaction between features that governs link formation in the graph G; i.e., to discover the latent matrix $\mathbf{W}$ of our model. This estimate alone allow us to use our model as a possible way to predict which pair of nodes form a link.

We then presented some methods able to learn this latent feature-feature matrix $\mathbf{W}$. To develop these methods, we restricted ourselves to techniques able to consider a (positive or negative) interaction between all possible pairs of feature, discarding methods considering only homophily. At the same time though, we considered only methods able to operate on large-scale graphs (millions of nodes).

To be able to accomplish both of these goals, we tried to adopt different simplifying assumptions. First, we developed a Naïve technique with the usual assumptions of Naïve Bayes approaches: the independence of probabilities of features, even under the knowledge that there is a link. This assumption gave us a very simple algorithm (described by (5.1)) which allow us to estimate $\mathbf{W}$; this algorithm also needed to assume an exponential activation function ϕ. However, we pointed out how its naïve assumptions can cause problems in practical applications, and for this reason we developed a different, more sophisticated, approach.

To do this, we streamlined the model by assuming it to be deterministic: all links should have a positive score. This corresponds to set the activation function ϕ to be the step function $\chi_{(0,\infty)}$. This simplification allow us to align our model equation to a perceptron decision rule; it is enough to apply an outer product kernel to the binary vectors $\mathbf{z}_i$ and $\mathbf{z}_j$ representing the

features in nodes i and j, and to make the perceptron predict whether they form a link or not. In this way, the internal state of the perceptron converges to the latent feature-feature matrix $\mathbf{W}$. We described this learning approach, and analyze what a classical bound on the number of errors of a perceptron means in our case. Then, since *any* perceptron-like algorithm can be adapted for our purposes we chose the simple and fast Passive-Aggressive algorithm [57] to concretely implement this approach (Alg. 3).

Finally, we tested how this algorithm behaves on simulations. We generated graphs and node-feature associations according to the model presented in section 4.3, under different assumptions: in particular, we employed the ϕ assumed by the Naïve algorithm (that is, the exponential function), the one assumed by the perceptron-like algorithm (i.e, $\chi_{(0,\infty)}$), and a third one, a classic sigmoid. To evaluate our methods, we showed to each algorithm just a part of the graph nodes, and we tested its scores on the remaining part; in other words, we employed a full knowledge of $\mathbf{Z}$ and a partial knowledge of G to estimate $\mathbf{W}$, which in turn was used to make predictions on the unseen part of G. We did this through a 10-fold cross-validation. In measuring the outcomes, we took care of the caveats recently stated for link prediction [193]. Results of these experiments showed how our learning approach outperforms the Naïve baseline in all the analyzed cases. Moreover, our algorithm turns out to be able to carefully make predictions on these simulated graphs, both in the case the activation function is the deterministic step function and in the case the graph was generated with a non-deterministic sigmoid.

Chapter 6

Discovering features

6.1 Introduction

As we have seen, a feature-rich graph can be represented by two matrices: one, that we call $\mathbf{Z}$, represent the associations between nodes and features; the other, $\mathbf{L}$, is the proper graph: associations among nodes. In the previous chapter, we showed how by predicting parts of $\mathbf{L}$ from $\mathbf{Z}$ – that is, predicting links in the network knowing each node features – we can assess the feature-feature matrix $\mathbf{W}$, the main latent factor of our model.

In this chapter, we will focus on the dual problem: how can we predict features, knowing the links? More precisely: given a full knowledge of the links, and a partial knowledge of the features – i.e., knowing the features possessed by a subset of the nodes – can we predict the features of unseen nodes? The method we have in mind is the following: after we trained our model on features of known nodes, it will be shown a new node and be informed of the features of its neighborhood; from that, it must be able to have a sense of what features that node should display.

We would like to see this task as an information retrieval problem: we want to assign a rank to all the existing features, so that relevant features – that is, the ones actually belonging to the node – are ranked higher than irrelevant features. So formally, our model will learn to rank features by taking the following inputs:

- the full graph $G = (N, L)$;
- a subset of nodes $T \subset N$, i.e. the training set
- for those nodes, their features: $\{\mathbf{z}_i \mid i \in T\}$.

Then, the output of this training phase is a function that for a certain node $i \in N$ can provide a ranking function $x_i : F \to \mathbb{R}$ able to identify its possible features.

Possible applications for this problem are copious. In the context of a linked corpus of documents, each tagged with tags defining its content, this problem translates to assigning tags to an unknown document just by knowing its links. For a semantic network which connects

related concepts, each tagged with a set of types, it is equivalent to identifying the types of a new concept, basing on its semantic connections; e.g., it means being able to understand that a concept linked from a director, and linking to a bunch of actors, could very well be a movie (or a TV series, and so on).

Our weapon of choice in this chapter will be neural networks. In the last years, neural networks proved to be very accurate in many different pattern recognition tasks, outperforming previous literature and in some cases outperforming human performance [52]. Similar problems have in fact been proposed in the literature – we will review some examples in section 6.2; but, to the best of our knowledge, they have never been tackled through neural networks. The versatility of these techniques—whose applications range from image classification to reinforcement learning—has in fact proven beneficial in our problem as well. In fact, the problem can indeed be cast naturally and painlessly into the neural network framework. As we will detail in section 6.3, the features appearing in the neighborhood of a node can be easily encoded as the input layer of the network; the desired output—a vector of real numbers, one per each feature—is the typical output layer of a multiclass classification problem. By describing our algorithm as a neural network we do not lose—and in fact we strengthen—the relationship to $\mathbf{W}$, which as we will see will be the backbone of the network. We will show how—similarly to the case of the previous chapter, where the unknowns were some of the links—the task of forecasting features naturally leads to an estimate of the latent feature-feature matrix $\mathbf{W}$. However, as we will see in our experiments with synthetic data in section 6.4, the reconstruction of $\mathbf{W}$ will be less effective, since the features are a noisier element; we will however be able to give an effective ranking of the features, able to discern relevant from irrelevant ones. Finally, we will sum up our findings in section 6.5.

6.2 Related works

In the last years, label prediction on graphs has received a great deal of attention. Many techniques, within various communities, have emerged. In the machine learning community, however almost all of them focused on a *single* (binary) label: it is usually assumed a partial labeling on nodes, with just positive or negative labels. Furthermore, they considered only the case where pure *homophily* regulates the behavior of the labels. The amount of homophily in the graph is referred to as the regularity of the labels in graph, and it is usually measured through the cut-size: the fraction of arcs between labeled nodes where the labels are the same.

Most algorithms for label prediction assume to deal with a regularly labeled network. Examples include the Label Propagation Algorithm [196]: it works by propagating labels in the graph in order to minimize the cut-size – i.e., obtaining the most regular graph possible from the known labels. Another well-studied algorithm is the Laplacian Kernel for the perceptron [94], for which formal bounds on the number of errors are available. It embeds each node from a graph with n nodes in a n-dimensional space, through of the Laplacian of the graph. However, this technique can not scale with reasonably sized graphs, and thus the graph needs to be transformed in a spanning tree. A similar strategy is used by the Weighted Tree Algorithm [45], which transforms the graph into a weighted tree, and then predicts labels basing on Nearest Neighbors.

Ali, Zappella, De Bie, and Cristianini [6] proposed an empirical comparison among these techniques. They generate networks by using news articles as nodes, their main topic as labels, and co-occurrence of names (precisely, of named entities) as links. Their goal is to generate feature-rich networks characterized by a high regularity (i.e., where labels display homophily). The resulting networks have around 3 000 nodes, and they show a cut-size which ranges from 4.4% and 16.8% (i.e., the fraction of links which connects nodes with a positive label to nodes with a negative one). Their most surprising finding is that a simple baseline – majority voting, where labels are predicted as equal to the majority of the labeled neighbors – performs better in every network. Furthermore, they find that Label Propagation obtains slightly worst results; since the main difference between the two methods is that the latter consider non-local effects, they conjecture that considering non-neighbors might be a source of noise, due to the small-world effect. However, we remark how all the examined algorithms only considered one label at a time, without dealing with inter-label interaction, nor multi-label prediction.

The focus on homophily and single-labels has been overcome in different areas and communities. In the specific field of functional genomics, label prediction on networks is needed to assign biological functions to (DNA and RNA) reading frames, a key problem in modern biology. Since this task can be framed as a multi-label prediction on graphs, it has sparkled interest in the problem, seeing (biological) functions as labels. In particular, Barutcuoglu, Schapire, and Troyanskaya [18] in 2006 wrote that "existing prediction approaches typically formulate the problem on a per-function basis", since it is "a convenient form for common machine learning algorithms". To solve this problem, authors proposed to consider existing ontologies for biological functions, viewing them as a tree-based relationship among the labels. This hierarchical structure forms the basis for an ensemble of Support Vector Machines [178]. They show that this enhances predictions for their task. Among the works that followed this line, we mention Schietgat *et al.* [163], where they highlight how a tree-based classifier can offer accurate and readable results, since it permits to visualize the learned relationship among labels.

Finally, we wish to highlight how the problem we are proposing presents strong similarity to those treated in the last fifteen years in the area of Statistical Relational Learning (SRL): works from this community try to model logic relationship between data, for example by considering directly entities as they are represented as tables in a relational model database, or as objects in an Object-Oriented paradigm. For an introduction, we refer to the one authored by Getoor *et al.* [78]. Typical statistical tools used in this area are Markov Random Fields [180] and Bayesian Networks [90], or graphical models such as LDA [27]. Kemp *et al.* [111] developed a powerful model able to make inference on arbitrarily complex systems of attributes, entities and relations. They obtain striking results on sets of related entities, whose size is at most in the hundreds of nodes; for example, their model inferred geo-political blocs from a database of 14 countries, 54 interactions between countries, and 90 features.

We wish to propose a model able to obtain results on much larger networks. The synthetic data we will use in this chapter will be of 10 000 nodes, on which our method will work in seconds. Nonetheless, we wish to deploy fully the modeling capabilities of our framework; without restricting ourselves to homophily, our approach could work in those scenarios (like ontology construction, or bioinformatics) where assuming a complex interaction among labels can be

highly valuable.

6.3 A neural network approach

Our goal is to train a model able to see the features appearing in the neighborhood of a given node in a large network, and from them infer its own possible features. We will try to investigate this problem from a purely neural network perspective. First, because in recent years they have been shown to be incredibly versatile and powerful at the same time (see for example the survey by Schmidhuber *et al.* [164], or other recent works [137, 168] for popular and interesting applications). Mikolov [135] proved how shallow neural networks can be of extreme interest in learning for knowledge representation. Secondly, contrarily to Bayesian models, they can scale up easily to very large datasets. Finally, a good reason is that, to the best of our knowledge, their use for the prediction of overlapping features in networks is novel.

Let us now describe the architecture of the network we used for this task. In Fig. 6.1, we report a visual depiction of the network.

Input layer. Our neural network essentially mimics the inference that is predicted by our model. For this reason, we split the neighborhood of a node i in its two directions: the in-neighborhood of i (i.e., $N^-(i) = \{j \mid (j,i) \in L\}$), and its out-neighborhood ($N^+(i) = \{j \mid (i,j) \in L\}$). For each, we considered two alternatives for the representation of feature k in the neighborhood. One way is the most natural vector representation, the *count*:

$$\mathbf{c}^+(i) := \sum_{j \in N^+(i)} \mathbf{z}_j, \quad \mathbf{c}^-(i) := \sum_{j \in N^-(i)} \mathbf{z}_j \tag{6.1}$$

That is, a pair of vectors where the k-th element is the number of occurrences of the feature k in the in- (or out-) neighborhood of the node i:

$$\mathbf{c}^+(i)_k = \Big|\{j \mid (i,j) \in L \wedge (j,k) \in Z\}\Big| \tag{6.2}$$

Continuing the previous example of a semantic network, if one of the unknown categories of i is a movie, one could expect the vector $\mathbf{c}^+(i)$ to have a high value corresponding to the category "Actor", since i will probably link to many nodes tagged as actors.

Another choice is the *ratio* between nodes with feature k which appears in the neighborhood and those that are outside of it. Reminding the notation introduced in section 5.3 we will call N_k the set of nodes displaying feature k; then, the ratio we will use[1] is:

[1] In practice we employ a small $\varepsilon > 0$ in the denominator in order to avoid numerical errors:

$$r^+(i)_k := \frac{\Big|N_k \cap N^+(i)\Big|}{\Big|N_k \setminus N^+(i)\Big| + \varepsilon}$$

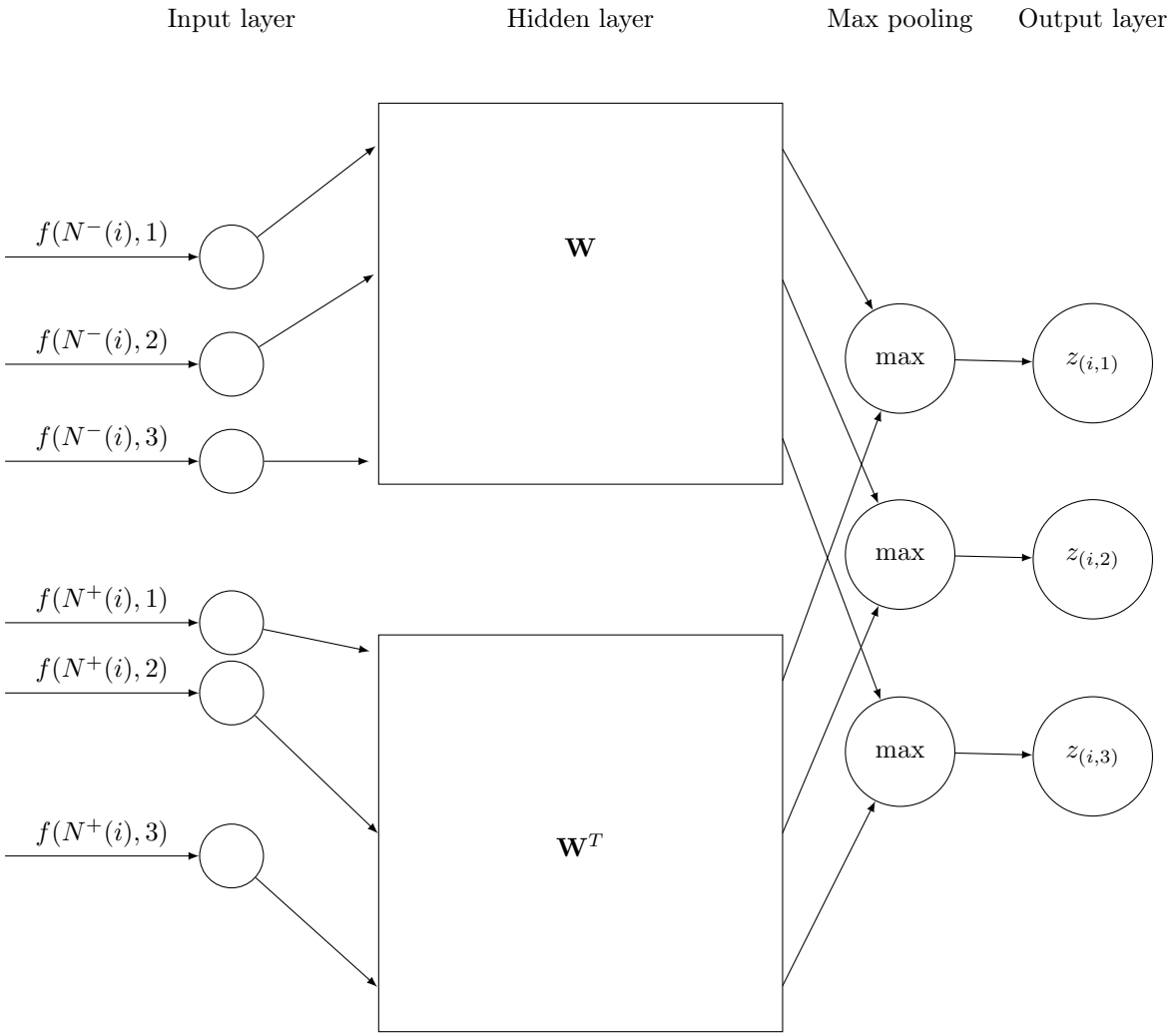


Figure 6.1: The main architecture of our neural network, for the case of $m = 3$ features. The input function f could be either the count or the ratio vectors, described respectively by equations 6.1 and 6.3. We indicate with max the node resulting in the max of its inputs, while $\mathbf{W}$ and $\mathbf{W}^T$ are matrices that multiply their input vector. See section 6.3 for details.

$$r^+(i)_k := \frac{\left|N_k \cap N^+(i)\right|}{\left|N_k \setminus N^+(i)\right|}, \qquad r^-(i)_k := \frac{\left|N_k \cap N^-(i)\right|}{\left|N_k \setminus N^-(i)\right|} \tag{6.3}$$

These two representations of the neighborhood, $\mathbf{c}$ and $\mathbf{r}$, differ in the information they capture. While $\mathbf{c}$ considers only information locally available in the neighborhood, $\mathbf{r}$ captures also some *global* information about features: a feature k which has a couple of nodes in the neighborhood of i will be highly represented by r_k if it is a rare feature, and lowly represented if it is a very common one. This behavior is natural considering how our model works. Instead, any node without feature k will not influence neither $\mathbf{c}$ nor $\mathbf{r}$.

This raw representation (that could be either $\mathbf{c}$ or $\mathbf{r}$) is directly fed into the neural network. The input layer is divided in two parts, which correspond to the two "local fields" of the considered node i: the features of the in-neighborhood and those of the out-neighborhood. We use the term "local field" in analogy with the local receptive fields of convolutional neural network in computer vision, which considers only a subregion of the input image. Exactly as in that case, also our neural network behavior is linked across these two fields.

Hidden layer. In fact, the main component of this network is, once again, the feature-feature latent matrix $\mathbf{W}$. For the in-neighborhood vector $\mathbf{c}^-(i)$, the network will compute $\mathbf{c}^-(i)^T \cdot \mathbf{W}$; for the out-neighborhood vector $\mathbf{c}^+(i)$, the computation will be $\mathbf{W} \cdot \mathbf{c}^+(i)$ or equivalently $\mathbf{c}^+(i)^T \cdot \mathbf{W}^T$ (so that the form stay the same of the in-neighborhood, but using $\mathbf{W}^T$ instead of $\mathbf{W}$). The sense of this transpose-based symmetry follows from the fact that the probability of a link (i, j) is higher when the value $\mathbf{z}_i^T \mathbf{W} \mathbf{z}_j$ is higher. Therefore, a node in the input layer representing feature k in the *in*-neighborhood will be multiplied by the k-th *column* of $\mathbf{W}$; the node for feature k in the *out*-neighborhood is multiplied by the k-th *row*. In this way, the parameters (i.e., the elements of the multiplying matrix) of the two different local fields are the same.

After this layer, the outputs of these two halves are merged through a max pooling layer.[2] The advantage of using a max-pool operation, instead of a sum-pool, is that the output gives more importance to the appearance of features and therefore to the activation of an output cell, than to a missing feature (since an appearing feature should be treated as an event more rare). However, experimentally we did not see any change in using a sum-pooling instead.

The output of this layer is the output vector $\mathbf{x}$, i.e. a ranking of the features of the node i such that a feature h that is likely to belong to i will correspond to a larger value in r_h.

Loss function. The estimate of the parameters is carried out through the classic backpropagation algorithm [159]. The backpropagation algorithm updates all the weights in the neural network (in our case, $\mathbf{W}$), following the gradient descent of a global error function. As a global error function, we employed the multi-label sigmoid cross entropy; Nam *et al.* [144] recently

[2] A max pooling layer is an operation that, given n vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ (in our case, $n = 2$), returns a vector $\mathbf{b}$ such that $b_i = \max_{i=0}^{n} a_i$. A *sum* pooling layer, instead, returns $\mathbf{b}$ such that $b_i = \sum_{i=0}^{n} a_i$.

showed how such a loss function can replace more complex measures previously adopted, gaining in efficiency and retaining or improving performances. Let us consider an input node i, and the output of our neural network $\mathbf{x}^i \in \mathbb{R}^{|F|}$ (i.e., a vector ranking each feature); let us refer to the set of actual features of i as $Y_i := \{k \parallel (i,k) \in Z\}$ and to its complementary set $\overline{Y_i} := F \setminus Y$; finally, let us recall the standard sigmoid function $\phi(x) = (e^{-x} + 1)^{-1}$. Then, the cross entropy loss function is:

$$H(\mathbf{x}^i, Y_i) = \sum_{k \in Y_i} \log\left(\phi(x^i_k)\right) + \sum_{k \in \overline{Y_i}} -\log\left(1 - \phi(x^i_k)\right) \tag{6.4}$$

According to Zhang *et al.* [195], this is classified as a first-order error function, since it only considers output features (*labels*, according to the usual terminology) separately, instead of considering them pair-wisely as other learning-to-rank error functions do. However, cross-entropy presents several advantages. First, it is obviously highly more efficient. Second, Dembczynski [61] proved that minimizing cross-entropy is consisent with minimizing rank loss (i.e., the number of misordering between a pair of correct and incorrect labels).[3] Finally, recent experiments from Nam *et al.* [144] showed how it works very well in practice on multi-label classification through neural networks.

To this loss function, we also add the 2-norm of the $\mathbf{W}$ matrix in the hidden layer, as a regularization factor. Therefore, the loss function we minimize is

$$\ell(\mathbf{x}^i, Y_i) = H(\mathbf{x}^i, Y_i) + \lambda \|\mathbf{W}\| \tag{6.5}$$

for a small $\lambda \in (0, 1)$. Other regularization methods, such as the widely employed dropout [174], turned out to be less effective in our experiments.

To minimize this function ℓ we used the recently described AdaGrad [64], which significantly simplifies setting the learning rate parameter, resulting in a more robust algorithm. Duchi *et al.* [64] showed the improvement both theoretically and experimentally; AdaGrad is now widely used in neural network training.

6.4 Results

In this section we are going to describe the results we obtained with this learning approach on feature-rich graphs synthesized by our model. The method we used is described in section 4.5; we are going to use, in fact, the same network we employed in the experiments described in section 5.5. As we summed up there, traits of these network closely resemble typical real complex networks (i.e., a power-law degree distribution). Details of the parameters used for the process were described in section 5.5.1. There, the reader can also find the main properties of these networks (average number of features per node, average degree, and mean harmonic distance). We will not consider the networks generated by the exp function, since we employed them only to test the Naïve algorithm described in section 5.3 in its most favorable setting.

[3] Directly minimizing rank loss is practically unfeasible, for computational reasons and since it is a non-convex function.

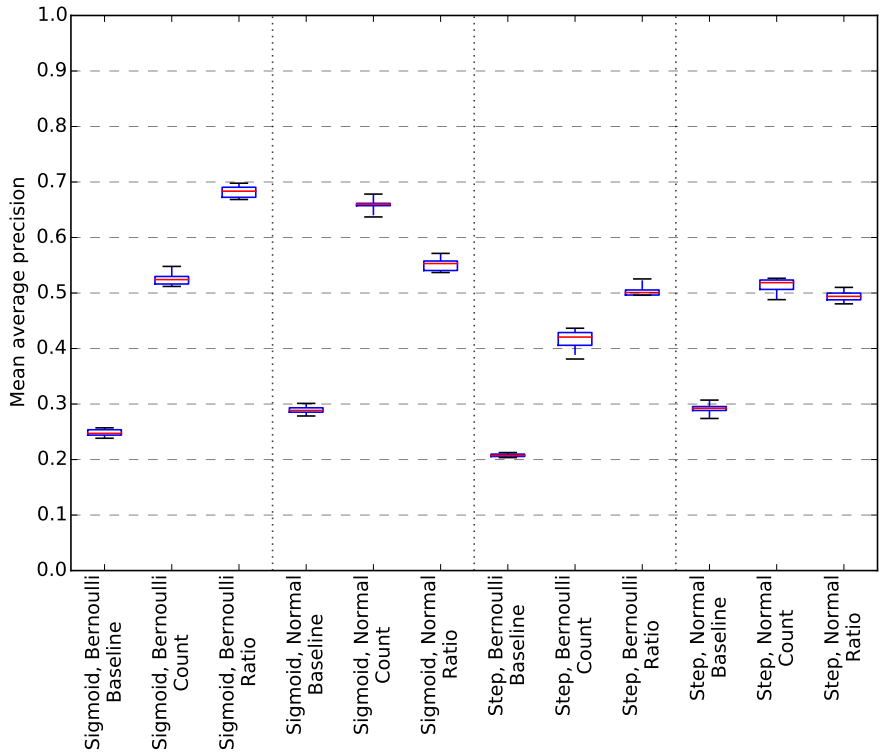


Figure 6.2: Mean average precision on 10-fold cross-validation. Red line is the median over the 10 folds, the edges of the box are the lower hinge (defined as the 25th percentile) and the upper hinge (the 75th percentile), the whiskers extend to minimum and maximum. Each box represent one of the graphs described in section 5.5.1, vectorialized through the count-based strategy (equation 6.1) or the ratio-based one (equation 6.3).

For each of these networks (of 10 000 nodes) we computed the input vectors representing each node, according to our two schemas: count (equation 6.1) and ratio (6.3). For each dataset we used 10-fold cross-validation, so that the test set nodes were unknown to the trained neural network. For each fold, we computed the Mean Average Precision (MAP).

We will also show results for a trivial baseline, feature-wise majority voting: according to this schema, the score attributed to feature k in node i is the fraction of the neighbors[4] of i (considering only those nodes included in the training set) which display feature k. Ali *et al.* [6] showed how this simple baseline outperforms, in their experiments, all the more sophisticated techniques for label prediction. As mentioned in section 6.2, however, this baseline and the other techniques they tested considered only *one* label at a time: our graph model centers instead on how *pairs* of label could foster links. Our technique, in fact, is focused precisely on understanding how different labels interact between them. The aim of this comparison is therefore to show how considering this point – inter-label interaction – could well be crucial in some data.

Seeing our goal as an information retrieval task, MAP is one standard measure. Consider to have a node for which no features are known, but its links to other feature-rich nodes are; our

[4]For simplicity, we will consider only the symmetrized graph.

goal is to provide a list of features so that the most likely features are on top. Take the following scenario as an example: we may have a linked document corpora of online newspaper articles, linking each other and displaying tags describing their content; the writer may want to have the system suggesting possible tags for a new article basing only on the related ones she provided. Average precision (AP) is the average of precision values at all ranks where relevant items are found; MAP is the mean of AP for all queries (that is, for all nodes in the test set).

First, we are going to use the 10-fold MAP to evaluate results for the ratio and count strategies. Results are reported in Table 6.1 and Fig. 6.2. We remind that these synthetic feature-rich networks have hundreds of features (they range from 545 to 658 features), and that each node has, on average, between 3.8 and 5.94 features, a very small part of F; therefore, an AP around 0.5 means that, on average, one can find all the correct features of the node in the top 10 features suggested by the neural network. In some cases – the Bernoullian $\mathbf{W}$, characterized by a simpler latent matrix, and the step activation function, which provide more information in this task – the ratio-based vectors are able to reach a MAP just below 70%. In other graphs – more noisy, since generated with a Gaussian $\mathbf{W}$ – the count vectors behave better, but with a much smaller advantage.

Furthermore, we employed a typical measure for evaluating classifier scores, the Receiver Operating Characteristic (ROC) curve[5]. Since this is a multilabel classification – moreover, with labels possibly overlapping – we have to choose how to aggregate curves; we chose to use the weighted macro-average, a common measure when labels are not uniformly distributed in the dataset [145, 193]. In weighted macro-average, results are aggregated by label, weighting each label accordingly to how popular it is in the dataset. In Table 6.2 we provide the Area Under Curve values for these averaged ROC with this weighted macro-average. We also provide, in Table 6.3, the unweighted macro-average, where features appearing in hundreds of nodes and features appearing in a couple nodes are weighted equally. Both these methods provide a different perspective than the MAP, which aggregates results by item, and not by label.

These measures confirm that the Ratio vectors perform much better. To give a more complete overview of its results, we also plot the ROC of each label individually. The reader can find the plots for the different networks in figs. 6.3 to 6.6.

[5]The considerations made in section 5.5.3, developed by Yang *et al.* [193], do not apply here, since this is a classification task and not a link-prediction one; for example, there is no undersampling in our test set.

	$S, \mathcal{B}$	$S, \mathcal{N}$	$\chi, \mathcal{B}$	$\chi, \mathcal{N}$
Ratio	$\mathbf{0.682 \pm 0.010}$	0.552 ± 0.012	$\mathbf{0.503 \pm 0.009}$	0.494 ± 0.009
Count	0.524 ± 0.010	$\mathbf{0.659 \pm 0.010}$	0.417 ± 0.016	$\mathbf{0.514 \pm 0.012}$
Baseline	0.248 ± 0.006	0.290 ± 0.007	0.208 ± 0.003	0.291 ± 0.010

Table 6.1: Mean average precision on 10-fold cross-validation. Each column represent one of the graphs described in section 5.5.1; the first two rows represent our different vectorialization methods: the count-based strategy (equation 6.1), or the ratio-based one (equation 6.3); the third row represents the baseline, feature-wise majority voting. For each MAP value we report the mean and the standard deviation across the 10 folds.

	$S, \mathcal{B}$	$S, \mathcal{N}$	$\chi, \mathcal{B}$	$\chi, \mathcal{N}$
Ratio	$\mathbf{0.839 \pm 0.005}$	$\mathbf{0.754 \pm 0.009}$	$\mathbf{0.759 \pm 0.005}$	$\mathbf{0.678 \pm 0.009}$
Count	0.678 ± 0.008	0.747 ± 0.009	0.623 ± 0.004	0.652 ± 0.012
Baseline	0.374 ± 0.007	0.423 ± 0.006	0.420 ± 0.004	0.467 ± 0.007

Table 6.2: Area under ROC curve, aggregated by weighted macro-average, on 10-fold cross-validation. Each column represent one of the graphs described in section 5.5.1; the first two rows represent our different vectorialization methods: the count-based strategy (equation 6.1), or the ratio-based one (equation 6.3); the third row represents the baseline, feature-wise majority voting. For each AUC-ROC we report the mean and the standard deviation across the 10 folds.

	$S, \mathcal{B}$	$S, \mathcal{N}$	$\chi, \mathcal{B}$	$\chi, \mathcal{N}$
Ratio	$\mathbf{0.678 \pm 0.011}$	$\mathbf{0.687 \pm 0.014}$	$\mathbf{0.609 \pm 0.012}$	$\mathbf{0.636 \pm 0.013}$
Count	0.611 ± 0.016	0.673 ± 0.013	0.589 ± 0.012	0.631 ± 0.010
Baseline	0.447 ± 0.006	0.481 ± 0.003	0.481 ± 0.004	0.488 ± 0.002

Table 6.3: Area under ROC curve, aggregated by unweighted macro-average, on 10-fold cross-validation. Each column represent one of the graphs described in section 5.5.1; the first two rows represent our different vectorialization methods: the count-based strategy (equation 6.1), or the ratio-based one (equation 6.3); the third row represents the baseline, feature-wise majority voting. For each AUC-ROC we report the mean and the standard deviation across the 10 folds.

By looking at the individual ROC plots, we can get more insights on the performance of our classifier. Darker curves – representing more popular features – constantly outperform the lighter, less popular, features. In other words, the classifier is much better at correctly predicting features that are more common in the network, while it makes many errors for quasi-unseen features. This behavior is quite obvious, considering that the rarest features in each dataset appear in 1 or 2 nodes, making it very hard for the classifier to generalize their effects.

From the ROC curve for the different networks, it also possible to notice different behaviors depending from the parameters of the network, confirming what we observed with the MAP. First of all, when $\mathbf{W}$ is Bernoulli-distributed on two possible values it is easier to fit it than when it is normally distributed. Second, the networks generated by a sigmoid activation function obtained much better results than those obtained with step-function-generated network. For the LLAMA algorithm (section 5.5.3), the sigmoid function represented an obstacle, since it meant that for the same inputs the outcome (i.e., the presence of a link) could be different; instead, for this task the non-determinism means that the classifier can distinguish when a feature pair cause a high score from when it causes a low score. In other words, the uncertainty leads to a more rich representation of possible outcomes.

6.5 Discussion

In this chapter, we confronted ourselves with a somehow dual problem with respect to link prediction (covered in chapter 5): given the links of a network where most of the features are

known, can we infer the remaining features? We saw this task as an Information Retrieval one, considering possible use cases (e.g., automatic suggestion of tags in a corpus of linked documents); for this reason, and since we expect this target to be affected by noise, we focused on obtaining a good *ranking* of the features of un-labeled nodes, thus using Mean Average Precision (the most common IR measure) as our main evaluation method.

To solve this task, we decided to design an ad-hoc neural network. Their use is novel in this area; they have proved to be powerful instruments when predicting labels in large datasets; also, they are a good way to infer knowledge from raw data. In our case, the network has as input a simple representation of the in-neighborhood and the out-neighborhood of a node; as output, a vector of real numbers, one for each existing feature. The hidden layer contains, once again, a latent feature-feature matrix $\mathbf{W}$ (for the in-neighborhood) and $\mathbf{W}^T$ (for the out-neighborhood).

We experimented this layout on the same synthetic data we used in chapter 5. Results are very encouraging: the obtained MAP ranges from 0.49 to 0.68, meaning that the top $2k$ features according to the output ranking are enough to find all the relevant results (where in our data $2k$ is orders of magnitude less than the total number of features). We also used the ROC and AUC-ROC measure, commonly employed in pattern recognition; the average of the AUCROC of every class, weighting each according to its importance, is around 0.75.

However, a great amount of future research is still needed: first of all, slight changes in our implementation choices – e.g., by introducing non-linearities in the network – could further improve our results. Secondly, a proper theoretical analysis of why this technique works would be beneficial to the understanding of both neural network and feature-rich graphs. Furthermore, it would be intriguing to mix this approach with auto-encoders for de-noising and dimensionality reduction [187]: by inserting another hidden layer between the input and $\mathbf{W}$ / $\mathbf{W}^T$, and another one, reversed, between the max-pooling and the output, one could easily train a function to reduce the dimensionality of the features of the nodes, making the actually employed feature-feature matrix much smaller, and thus allowing to treat larger feature spaces in data.

Finally, a forthcoming goal of our endeavors will be to employ this technique on real networks: possible fields of application include both ontology construction and bioinformatics, where graphs equipped with many, intertwined, features are the primary source of information.

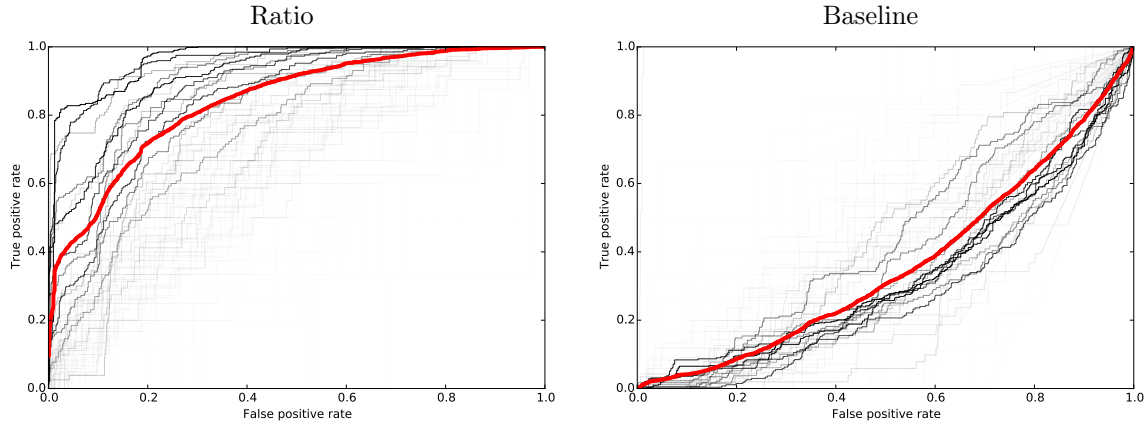


Figure 6.3: ROC curves for the first fold, in the network $S, \mathcal{B}$, using the Ratio vectors and the baseline, feature-wise majority voting. Black lines represent the ROC of each single feature in the feature-rich network; darker lines represent most popular features, lighter the less popular ones. The red line represent the weighted macro-average ROC among all features.

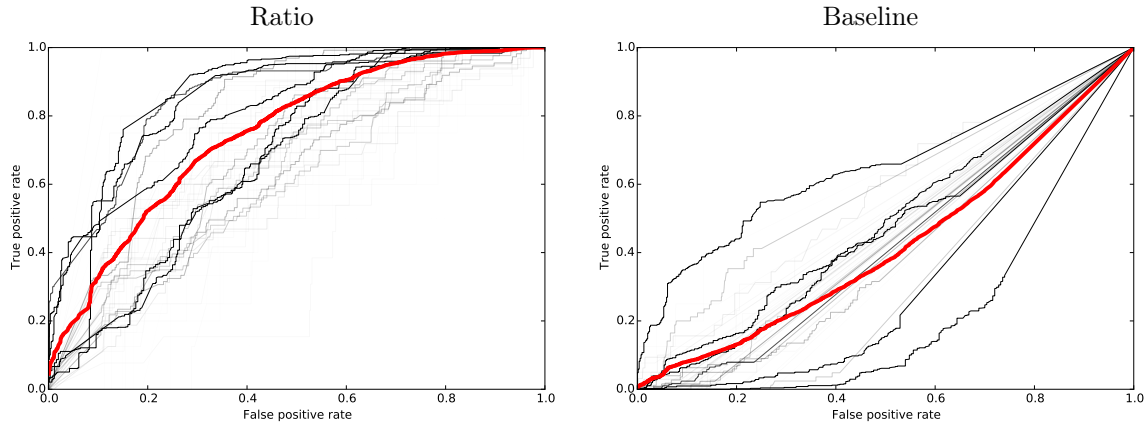


Figure 6.4: ROC curves for the first fold, in the network $S, \mathcal{N}$, using the Ratio vectors and the baseline, feature-wise majority voting. Black lines represent the ROC of each single feature in the feature-rich network; darker lines represent most popular features, lighter the less popular ones. The red line represent the weighted macro-average ROC among all features.

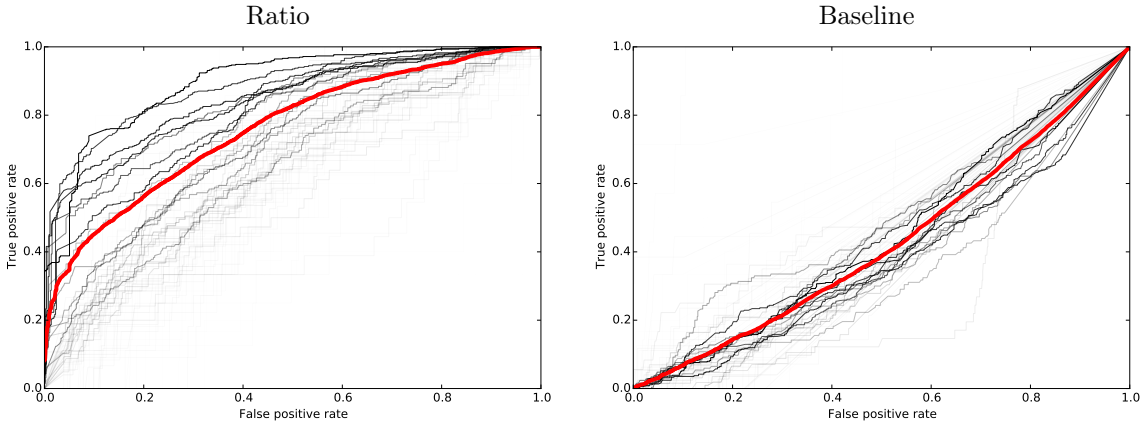


Figure 6.5: ROC curves for the first fold, in the network $\chi, \mathcal{B}$, using the Ratio vectors and the baseline, feature-wise majority voting. Black lines represent the ROC of each single feature in the feature-rich network; darker lines represent most popular features, lighter the less popular ones. The red line represent the weighted macro-average ROC among all features.

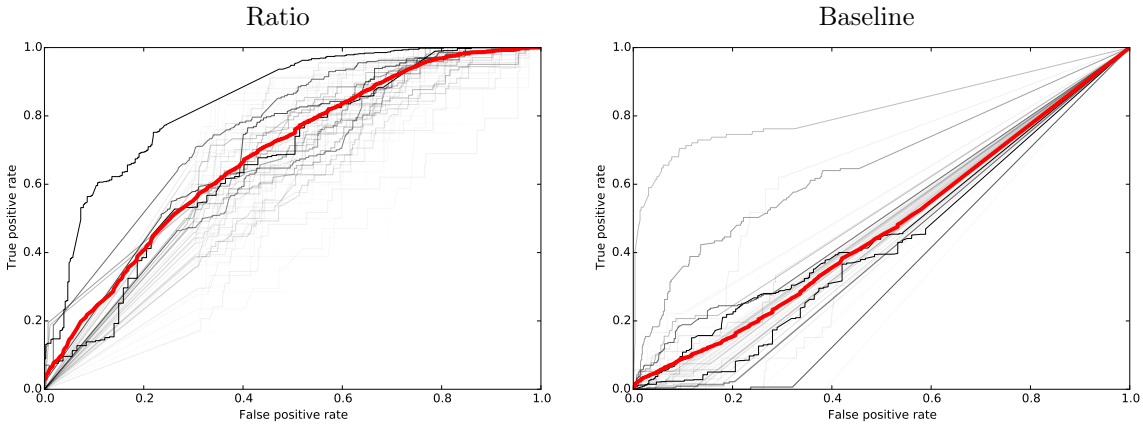


Figure 6.6: ROC curves for the first fold, in the network $\chi, \mathcal{N}$, using the Ratio vectors and the baseline, feature-wise majority voting. Black lines represent the ROC of each single feature in the feature-rich network; darker lines represent most popular features, lighter the less popular ones. The red line represent the weighted macro-average ROC among all features.

Part III

Experiments

Contents

Chapter 7

Citation networks

7.1 Introduction

In this chapter, we will use *citation networks* as test beds for our model. Citation networks are a kind of *information networks* [146]. Specifically, they involve relations between scientific papers. There is, in fact, a tradition dating back at least to the nineteenth century of *citing*, inside one's work, previous works that treated similar subjects; referencing other scientific works is supposed to *"identify those earlier researchers whose concepts, methods, equipment, etc. inspired or were used by the authors in developing his or her own article"* [65].

Thanks to those references, it is possible to build a citation network: each node represents a scientific work, and a directed edge (i, j) represents the fact that the article i contains a citation to the article j. The importance of such networks in the field of bibliometry is well-known, and we will see how our model can give valuable insights.

In section 7.3, we will see first how our node-feature model (section 4.3) can be used to estimate parameters of a real node-feature association, where nodes are scientific papers and features are the (uncommon) words they use. We will also see how these parameters can then be used to simulate a different but realistic (i.e., globally similar to the given one) node-feature matrix. Then, we will show how this matrix can be used to produce a network, according to our model. Such a network – we will illustrate – has global topological properties that closely resemble those of a citation network. Specifically, we will look at their degree distribution – a classic fingerprint of social and information networks – but also at their distance distribution – a more complex, geometric attribute of a graph. These observations will qualify our model as a plausible way to simulate a realistic complex network. Then, in section 7.4 we will define the *explainability* of a specific feature set for a network as a measure of how good our model is, using it to predict the links by looking only at the features. We will see how the Llama approach presented in section 5.4.2 is able to grasp such a relationship better than a Naïve approach on a scientific network with tens of millions of nodes; with these tools, we will investigate which features can adequately explain a citation network. In doing so, we will also replicate with real, citation network data what we experimented on synthetic data in section 5.5.

Section 7.3 contains content previously published in *"Information Sciences"* [30].

7.2 Related works

The first author to study citation networks was Derek Price in 1965 [59]. He studied and defined a citation network in the modern sense; among its results, there is one of the first empirical observations of the appearance of a power law in the degree distribution of a complex network.

These first studies initially employed hand-curated sources of data to build citation networks. In this category, the first seminal example is the Science Citation Index [72]; other examples of hand-curated sources include Scopus, still widely employed in bibliometry. In the last decades, however, a new dimension of such analysis was made possible by the automatic crawling of scientific works, which lead to larger data sources, such as Citeseer and Google Scholar. In 2015, Microsoft released a very large dataset for academical purposes, the Microsoft Academic Graph [170]—that we are going to employ and describe later in this chapter.

It was noted already in the 1970's that the citation network can be used to assess the importance of journals [73]. Different metrics were developed basing on this graph, including the simple number of citation (i.e., the in-degree) and the well-known H-index. Although self-citations and different motives for citing a paper makes it difficult to assign a precise meaning to citations, nonetheless citation analysis has become a standard tool in bibliometry [65].

Beside citation networks, it is worth mentioning that other networks can be defined regarding citations of scientific works: Newman [146] describes two network types that can be derived from a citation network, that are co-citation networks, where i and j are connected if there is a paper k that cites both i and j, and its dual, the bibliographic coupling network, where i and j are linked if there is a paper k that *is cited by* both i and j.

Citation networks, since they describe connections between papers, are often associated with features characterizing each paper; those features interconnect in singular ways with their links. They will be our main focus in this work.

Kiss *et al.* [116] described how an epidemiology model can be used in tracking information spreading on a citation network; they found that this approach can achieve better agreement with empirical data (they employed a dataset of 4 021 biology papers) when each node is characterized with the categories it belongs to.

Jo *et al.* [106] found that citation links and text from papers can produce, when their use is combined, a more powerful way to perform topic modeling on research papers. The same authors [105] also provided a way to model the latent interaction between topics, from the citation network—a technique in fact related to the Naive Bayes baseline we derived for our model in chapter 5. Then, they observed how this topic-topic interaction network evolves in time, using as a dataset the ACM corpus, comprising 129 544 papers.

In this chapter, we will perform an analysis of how features of scientific works interplay with the links, by using the techniques we derived for our model in the previous chapters. In particular, we will analyze how such techniques work when features are uncommon words appearing in the paper, affiliations of the authors in the paper, and topics categorizing each paper.

We will first show how our model can properly simulate a node-feature matrix in this case. Then, for the last two set of features, we will estimate how much they are interconnected with

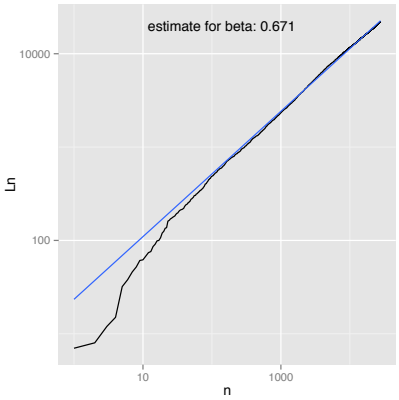


Figure 7.1: Correspondence between the parameter β and the power-law exponent of L_n, as a function of n. The estimate of β is the slope of the regression line.

the citation networks, by measuring the accuracy of the algorithms we presented in chapter 5. We will do so on a network of 18 939 155 papers in section 7.4.

7.3 Simulating a feature-rich network

To simulate our model on a real dataset, we considered a sample of scientific papers[1] (originally released as part of the 2003 KDD Cup) consisting of 27 770 papers from the "High energy physics (theory) arXiv" database. For each paper (node), we considered as features the words appearing in its title and abstract, excluding those that are dictionary words.[2] The papers were organized in order of publication date. These papers form a citation network, where papers (i, j) form a link if one cites the other.

What we want to do with such a dataset is to validate both our models: the one for Z and the one for G. Specifically, we will do the following:

1. We will try to generate a node-feature association that resemble the original one; that is, we will try to estimate the parameters of $\mathbf{Z}$ through the estimators presented in sec. 4.3, in order to generate another simulated node-feature matrix. This matrix will be supposedly drawn from the same distribution (the one defined by our model); then, we will compare the simulated one with the original.

2. We will then use the generated node-feature matrix to simulate a graph with a density similar to that of our network. In doing this, we want to test empirically the simulations we have done in sec. 4.5; for this reason, we will consider $\mathbf{W} = I$ as we did there. We will then compare the global properties of this simulated network with those of the citation graph we are trying to emulate.

[1] The dataset is available within the SNAP (Stanford Large Network Dataset Collection) at `http://snap.stanford.edu/data/cit-HepTh.html`.

[2] According to the Unix `words` dictionary.

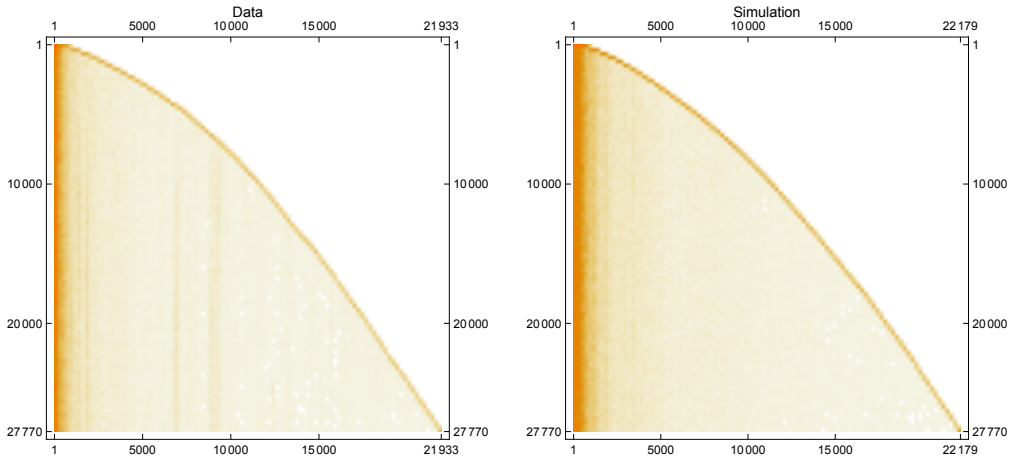


Figure 7.2: Comparison of the real matrix **Z** extracted from the data (on the left) with the one simulated by our model (on the right).

The model we have in mind is the following: when a paper is published, it can be characterized by a set of scientific terms; of these terms, some are introduced *ex novo* by that paper; others are acquired from previous papers. The attribute matrix **Z** for this case has $Z_{i,j} = 1$ iff paper i contains the term j. As we explained, our model assumes that some papers have a larger capability to transmit their terms to future papers, making them popular. In this real matrix, the overall number of features is 21 933, with a matrix density of $0.35 \cdot 10^{-3}$ (there are 214 510 ones in the matrix).

Estimating the parameters. First, we estimate α' and β with the tools presented in section 4.3.1, in particular by using Remarks 4.3.2 and 4.3.1; we recall that β is the power-law exponent of the asymptotic behavior of L_n, i.e. the overall number of distinct attributes.

The estimated values of α' and β are 15.038 and 0.671, respectively. We show the estimate for this real case in Figure 7.1. This regression plot confirms how our estimators fit the real data; it is also very close to the analogous, simulated plots of Figure 4.2.

Then, we proceed to find an approximate value for c via Simulated Annealing [115], obtaining $c \approx 175$. Finally, we use the Gibbs sampling algorithm presented in section 4.6 for the fitness values of the nodes, finding a plausible realization of $\mathbf{r} = r_1, \cdots, r_n$.

7.3.1 Simulating the node-feature association

We proceed to use the estimated parameters to generate a different node-feature association matrix **Z**. The point here is to *simulate* a different set of feature-rich nodes, with similar properties to the real ones. Possible applications include, for example, testing an algorithm for feature-rich graphs on many instances of a problem. We do not expect the simulated **Z** to be close in its *values* (that is the reason why we will not compare them using distance between matrices), but to have similar *global properties*. In fact, what we are going to do is extracting a different particular item from the distribution that – we hypothesize – is behind the real data we are seeing. We chose to

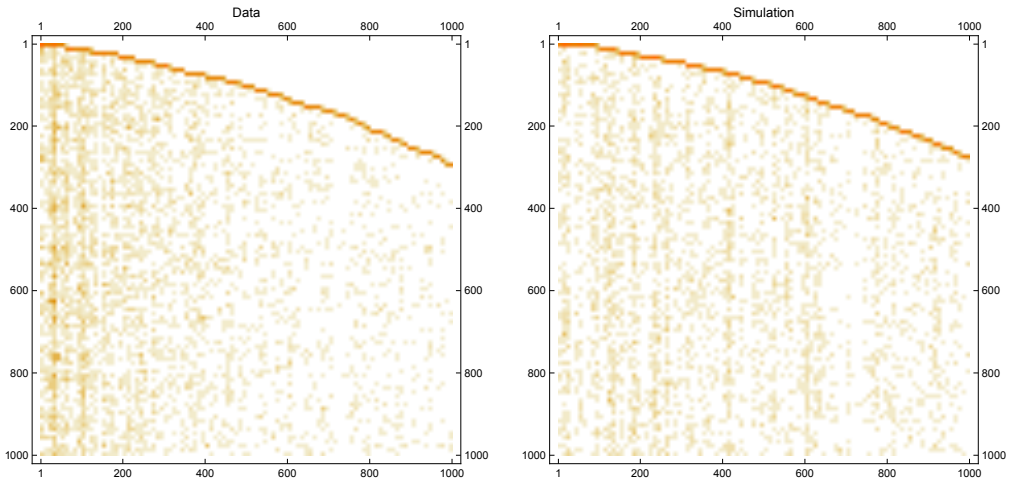


Figure 7.3: Comparison restricted to the first 1 000 nodes and the first 1 000 features of the real matrix $\mathbf{Z}$ (on the left) with the simulated one (on the right).

restrict our analysis to *one* simulated matrix (instead of a sample of simulations) in order to get a deeper analysis, to keep the comparison simple and depictable, and for better computational tractability.

The simulated matrix $\mathbf{Z}$ has 22 179 different features, with a matrix density of $0.37 \cdot 10^{-3}$ (there are 228 328 ones in the matrix). We remind that the real matrix has a density of $0.35 \cdot 10^{-3}$. We show a summary plot comparing the real matrix and the simulated one in Fig. 7.2. We can see how they fundamentally represent the same phenomenon; in particular, also, the progression of L_n (the number of different feature in the first n nodes) is the same. We depict a closer look in Fig. 7.3.

Then, we measure if the number of *nodes per feature* – i.e., how many nodes possess a certain feature – behaves in a similar way among the two matrices. We compare the distribution of it in the real data and in the simulation in Figure 7.4; except for some missing peak in the long tail[3], the simulation behaves once again in a very realistic way.

Finally, we do the opposite analysis, by plotting the number of features per nodes in Fig. 7.5. Here we see a similar distribution but with slightly different parameters; in particular, the average number of feature per node is 9.8 in the real data and 10.3 in the simulation.

7.3.2 Simulating the graph

Finally, let us compare the graph produced by our graph model applied to the simulated $\mathbf{Z}$ matrix we analyzed in the previous section. We are going to show how, topologically, the produced graph is very similar to the citation network we are considering—and, more in general, how its global properties resemble those typical of a social network.

We will make the same assumption we used for simulations in section 4.5:

- We consider *symmetrized* (i.e., undirected) graphs.

[3]Note that the missing peaks correspond to a very small number of features appearing in a large number of nodes in the real matrix.

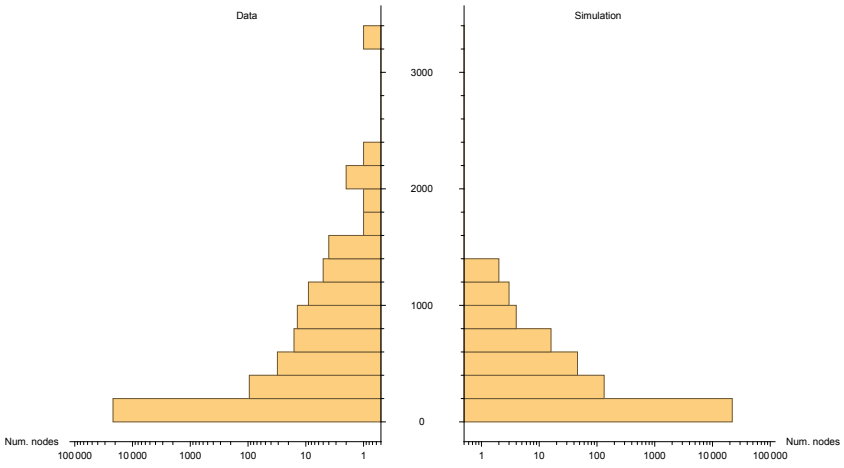


Figure 7.4: Comparison of the distribution of the number of nodes per feature, in the real data (on the left) and in the simulation (on the right).

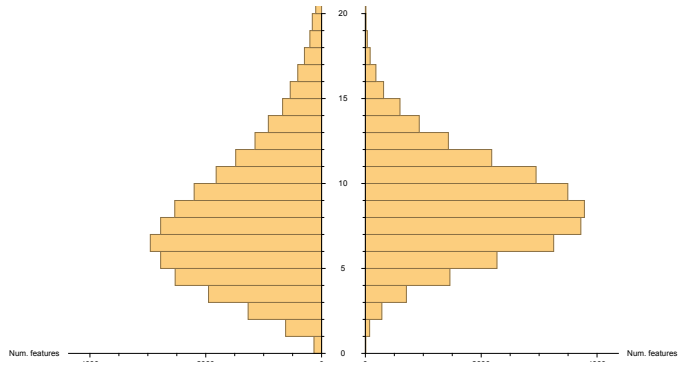


Figure 7.5: Comparison of the distribution of the number of features per node, in the real data (on the left) and in the simulation (on the right).

- We will set $\mathbf{W} = I$.
- We also omit self-loops, i.e., edges of type (i, i).

With these assumptions, we fed the $\mathbf{Z}$ matrix we simulated into our graph model. The Newton method we described in section 4.5 to estimate ϑ found that, in order to obtain a graph with the density of the real citation graph, we have to set $\vartheta \approx 2.105$. Moreover, results depend also on the choice of the parameter K: after some experiments, we observed that we can obtain a good fit with $K = 5$.

This model produces a quite similar degree and distance distribution, as shown in Figure 7.6. From that figure, we can note how our model can be able to replicate not only classical power-law degree distribution—as we saw in section 4.5—but also those closer to a log-normal distribution, a trait that has been recently noted in social networks [83, 99].

As a matter of fact, the two graphs – the one obtained from our model, and the original citation graph – share a number of global topological properties. We report them in table 7.1. There, we can see that the density of the graph was correctly approximated by our method, and

Property	Real data	Simulation
Density	$9.13 \cdot 10^{-4}$	$9.39 \cdot 10^{-4}$
Avg. degree	25.4	26.0
Reachable couples	95.0%	95.2%
Mean harmonic distance	4.12	3.54

Table 7.1: Comparison of the `cit-HepTh` dataset *versus* a graph generated by our model.

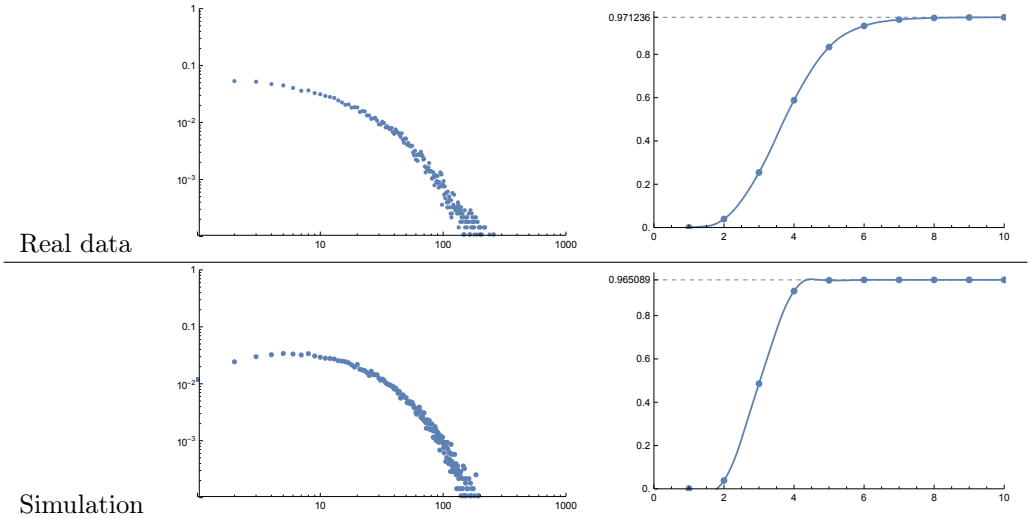


Figure 7.6: Comparison of the `cit-HepTh` dataset *versus* a graph generated by our model. We show the degree distribution in a log-log plot, and the fraction of pairs at distance at most k; in the latter, we highlight the peak value, indicating how many pairs of nodes are mutually reachable.

it is in fact around $9 \cdot 10^{-4}$ for both the real and the simulated network. Therefore, also the average degree results in very similar values (around 26). By looking at the plot, however, it is clear that not only the average degree is close: also the degree distribution is very similar to that of the citation network we are considering.

Also more complex attributes of the network – i.e., distances between nodes – are quite close. In particular, the fraction of reachable pairs – that is, pairs (i, j) for which there is a path between i and j – is around 95% in both cases. From Figure 7.6 we can see how these distributions look quite similar, but – like highlighted in table 7.1 – the resulting mean distance is of 4.12 in the real network and of 3.54 in the simulation. We think this shrinking effect of the simulation could be an artifact of our use of $\mathbf{W} = I$ and thus considering *homophily* as the only way for features to interact. We will explore concretely how to reconstruct a proper $\mathbf{W}$ from citation network data in the next section. However, it is still striking to observe that the two graphs have such a strong similarity in their topology, albeit having positively no direct relation with each other.

7.4 Explaining a network through its features

In this section, we will focus on how our framework – and especially the Llama algorithm (Alg. 3) – can be used to evaluate the relationship between a network and a particular set of features for its nodes. In particular, we will consider the fitness of our model as a measure of how much a certain set of feature can explain the links in such a graph.

Explainability. Given a graph $G = (N, L)$ and a particular set of features $\hat{F}$ that can be associated to its nodes (with $\hat{Z} \subseteq N \times \hat{F}$), we can define the *explainability* of $\hat{F}$ for G to be the area under the precision-recall curve obtained by our model scores; with "score" we mean the argument of ϕ in (3.1), where the matrix $\mathbf{W}$ is the one found by Alg. 3 when inputted with G and $\hat{F}$. The use of the AUPR (Area Under Precision-Recall curve) as a measure of the fitness of our model is due to the theoretical and experimental properties of such a measure analyzed by Yang *et al.* [193], as we described in section 5.5.3.

With this tool in our suitcase, we can take a network, and measure how much a particular property of its nodes can explain their links; this is what we are going to showcase in this section.

7.4.1 Experimental setup

We are going to consider a scientific network recently released by Microsoft Research, and known as the Microsoft Academic Graph [170]. It represents a very large (tens of millions), heterogeneous corpus of scientific works. For every scientific work, some metadata is available in the dataset.

We will consider the citation network formed by these works: that is, a (directed) arc $(i, j) \in L$ will correspond to the paper i citing the paper j. As for the features, we will consider the following sets of features for our nodes/papers:

- Their *affiliations*: for each paper, all the institutions that each author of the paper claims to be associated to. For example, "University of Milan" and "Google" appear in our list of affiliations.
- Their *fields of study*: the field of study associated to the keywords of the paper. For example, "Complex network" and "Vertebrate paleontology" appear in our list of fields a paper can be associated to.

Those features fully respect all the assumptions we made: they are attributes of the nodes, they are binary (a node can have a feature or not, without any middle ground), they are possibly overlapping (a paper can have more than one affiliation/field).

Our goal now is to compare the explainability—as defined above—of these two sets of features for the citation network. Since we want to compare them fairly, we reduced the dataset to those nodes for which the dataset specifies both these features: that is, papers for which both the affiliations and the fields of study are reported. In this way we obtained:

- A graph $G = (N, L)$ where N is a set of 18 939 155 papers, and L is all the 189 465 540 citations between those papers.

		Same affiliation	**Different affiliation**
	Total	48.7%	51.3%
Same field	38.6%	18.7%	19.9%
Different field	61.4%	30.0%	31.3%

Table 7.2: A measure of homophily in this dataset, obtained on a sample of 50 000 links. In this table, we say two nodes have the "same" feature if they share *at least one* feature.

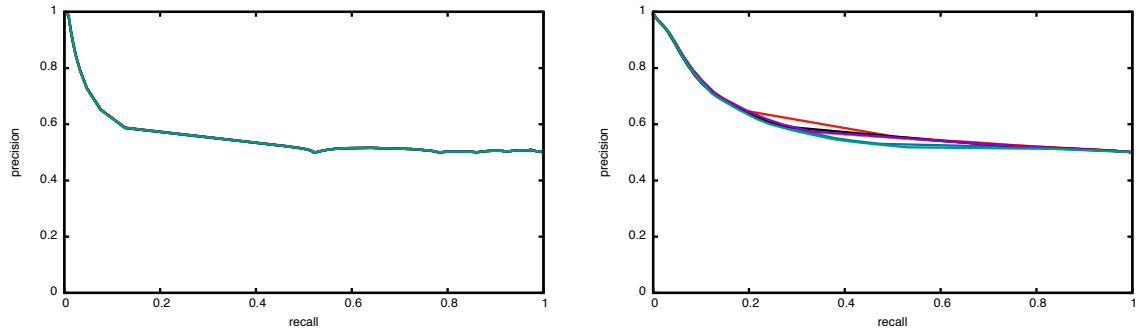
- A set F_a of 19 834 affiliations, and the association Z_a between papers and affiliations. Each paper has between 1 and 182 affiliations; on average, we have 1.36 affiliations per paper.
- A set F_f of 47 269 fields, and the association Z_f between papers and those fields of study. Each paper involves between 1 and 200 fields; on average, we have 3.88 fields per paper.

We proceeded then to evaluate the explainability of F_a and F_f for G with the same approach presented in section 5.5.3:

1. We divide the set N in ten folds $N_0, \ldots, N_9$.
2. For each fold N_i:
   (a) We apply Alg. 3 to the part of L and Z related to the training set $\cup_{j \neq i} N_j$.
   (b) We obtain a matrix $\mathbf{W}$.
   (c) We compute the scores of our model with $\mathbf{W}$ on the test set N_i.
   (d) We measure the precision-recall curve for these scores.

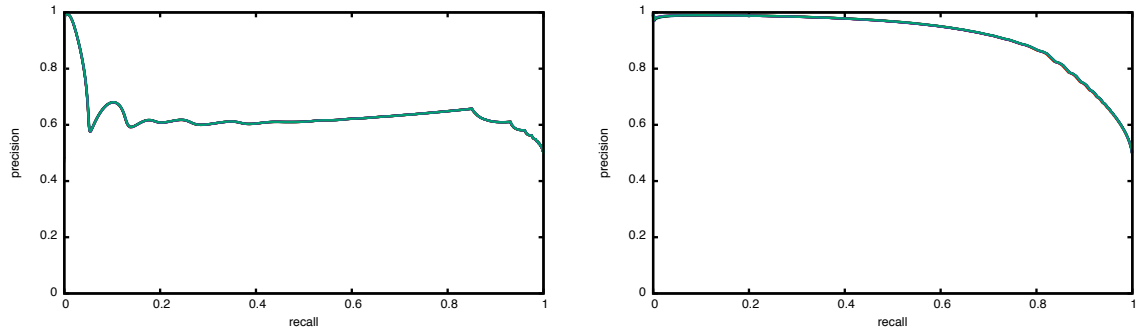
In order to validate on real data the results we obtained in section 5.5.3 for synthetic data, we also carried out the same procedure also with the $\mathbf{W}$ matrix found by the Naïve approach. As a further type of test, we also performed the same experiment on a third set of features: $F_a \cup F_f$, that is the union of affiliations and fields. As a result, we obtained 2 ten-folded precision-recall curves for each of the 3 considered set of features: F_a, F_f and $F_a \cup F_f$.

Measuring homophily. In order to assess how much homophily alone is present in the dataset, we measured the fraction of links (i, j) where i and j share *at least one* feature. We performed this measure for our two feature sets, on a subsample of 50 000 links. We report results in Table 7.2, where we also report how correlated the homophily among fields of research is correlated with homophily among affiliations. These results show (1) that the correlation between these two kinds of homophily seems to be practically non-existent; and (2) that homophily alone is not sufficient to explain links: 31% of citation links happens to be between nodes which do not share any affiliations nor fields of research. The adoption of our model, which consider interactions between *different* features in order to explain links, seems therefore justified.



Naïve Llama

Figure 7.7: Precision-recall curves of the Naïve baseline and of the Llama algorithm, when explaining the citation network using the affiliation of authors as features.



Naïve Llama

Figure 7.8: Precision-recall curves of the Naïve baseline and of the Llama algorithm, when explaining the citation network using the fields of study of each paper as features.

7.4.2 Results

We report in Fig. 7.7 the precision-recall curves for the Naïve and for the Llama algorithms for the affiliations feature set; in Fig. 7.8, for the fields of study feature set; in Fig. 7.9, for the joined feature sets $F_a \cup F_f$. Finally, in Table 7.3 we report the explainability we obtained – that is, the area under the said curves.

In this table, we can see that the explainability of the fields of study for the citation network is much higher than that of the authors' affiliations: the first is above 0.91, while the second is 0.59. In this sense, our model allows us to say that the fields of study of a paper explains very well its citations, while the affiliations of the author do not. This might not come as a surprise—the relationship between the fields a paper belong to and its citations is quite natural—but our contribution here is the formal framework which allow us to back this assertion with solid numbers, through (3.1) and Alg. 3. We can further validate this affirmation by looking at the explainability for both fields and affiliations, joined: its value of 0.921 is just faintly over the value of 0.9175 obtained for fields alone, implying that the gain obtained by including this whole new set of 19 834 features is practically negligible.

Algorithm	Affiliations	Fields of study	Both
Llama	$\mathbf{0.593} \pm 0.005$	$\mathbf{0.918} \pm 0.000$	0.921 ± 0.001
Naïve	0.552 ± 0.000	0.632 ± 0.000	0.635 ± 0.000

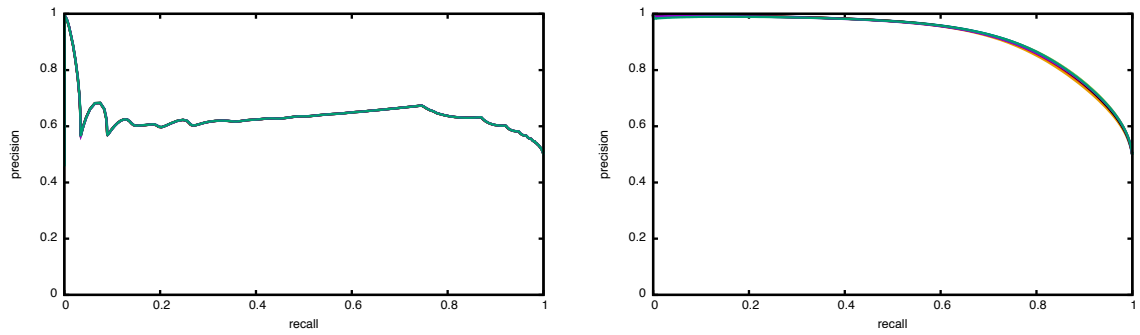
Table 7.3: Area under the precision-recall curve of the Naïve baseline and of the Llama algorithm. For each of the considered feature sets, we report the mean and the standard deviation across the ten folds. We highlight the *explainability* for the citation network of the affiliations and of the fields of study, respectively.

Algorithm	Feature set	Number of random features	AUPR
Naive	Fields of study	100	$0.631 \pm .000$
Naive	Fields of study	1000	$0.628 \pm .000$
Naive	Affiliations	100	$0.551 \pm .000$
Naive	Affiliations	1000	$0.547 \pm .000$
Llama	Fields of study	100	$0.918 \pm .000$
Llama	Fields of study	1000	$0.917 \pm .001$
Llama	Affiliations	100	$0.588 \pm .006$
Llama	Affiliations	1000	$0.592 \pm .003$

Table 7.4: Area under the precision-recall curve of the Naïve baseline and of the Llama algorithm, in the case of an extra number of random features added. For each of the considered feature sets, we report the mean and the standard deviation across the ten folded cross-validation.

We can grasp more details by looking at the specifics of the precision-recall curves. By comparing the Llama curve for affiliations in Fig. 7.7 and the one for fields in Fig. 7.8, we can see immediately that the latter depicts a valid classification instrument; there, precision and recall break-even point is around 83%. Also, we can see some specific aspect of the affiliation feature set: it is in fact able to reach a high precision, but only in the very low range of recall. Here, a precision of 83% is possible only with a recall lower than 7%: the reason behind this is that an author's affiliation is effectful in encouraging a citation link in a very limited set of circumstances; we can conjecture that homophily within small institutions could be an example.

We performed an additional experiment to evaluate how much our algorithms are prone to overfitting. For each of the two considered feature sets, F_a and F_f, we added r random features. Each one of these features had a probability p to be associated with a given node i; in other words, each $z_{i,k}$ is a Bernoulli variable with parameter p. Each of these associations is independent, and it is independent also from any link present in the network, making those features, for all purposes, noise. As a probability p we used the average probability of each feature set (i.e., the density of the matrix $\mathbf{Z}$); this meant a $p = 0.00008$ for fields of studies and $p = 0.00006$ for affiliations; such a probability results in an average of 1515 nodes displaying each added random feature in F_f and of about 1326 nodes for each random feature in F_a. We tested how our algorithms perform in this setting, in the case of $r = 100$ and $r = 1000$ random features added. We report results in Table 7.4. By comparing these results with those in Table 7.3, we can see that—although the added noise, and the amount of added noise, is in general detrimental to the performance of the



Naïve Llama

Figure 7.9: Precision-recall curves of the Naïve baseline and of the Llama algorithm, when explaining the citation network using both the affiliations and the fields, together, as features.

algorithm—the difference is almost negligible: for example, the Llama algorithm on the fields of study has an AUPR of 0.918 without noise, and of 0.917 with $m = 1000$ random features; for the Naive algorithm, a performance of 0.63 degrades only to 0.62. While it is obvious that introducing a much greater amount of noise would start to ruin the performance, we think that this experiment proves that the proposed algorithms are not too sensible to noise.

Finally, let us remark how the results we obtained on synthetic data in section 5.5.3 are fully confirmed by the real data we presented here: the Llama algorithm, in all the three cases, behaves much better than the Naïve baseline. This is especially true for the feature set that actually explains the network: for the fields of study, Llama is able to get a 0.91 area under the precision-recall curve, while the $\mathbf{W}$ matrix found by Naïve approach can get only 0.63. In particular, precision-recall curves look similar to the one pictured in Fig. 5.11, corresponding to the simulation obtained with ϕ set to a sigmoid and $\mathbf{W}$ discretely-distributed; real data is actually less shaky, due to the fact that we have 18 millions nodes instead of the 10 000 we simulated. Besides confirming the validity of the Llama approach, this observation also confirms the goodness of our model for simulations.

7.5 Discussion

In this chapter we tested our techniques, described in the first parts of this work, on a particular kind of complex network: citation networks. Citation networks are representative of how works of different groups of people interconnect and influence each other. Their importance in the field of bibliometry is well-known since the early 1970's.

Firstly, we tested how our model, described in the first part of this thesis, can be used to simulate a citation network and its features. As features, we employed the non-common words that appear in each paper title and abstract. The dataset we employed is the High Energy Physics arXiv dataset, consisting of 27 770 papers.

On this dataset, we made use of the methods described in chapter 4. First, we used the estimators and the heuristics we developed for our model in order to estimate the latent parameters

that lead our model of co-evolution between nodes (papers) and features (words). We estimated the α and β parameters of our model through unbiased statistical estimators; the c parameter through Simulated Annealing; the fitness parameters through our Gibbs-sampling algorithm. After having done so, we used those latent factors to kickstart the model, which produced a new feature-rich complex network, close—in its global properties—to the original one, but with different local values. In other words, we found the distribution governing the matrix $\mathbf{Z}$ that we were looking at, and used this distribution (and its parameters) to extract a different matrix. We showed how the original and the synthetic matrices displayed similar global metrics: they displayed very similar density, progression of the number of features in time, and final number of nodes per feature. We believe this justifies our usage of such a process to synthesize realistic datasets.

Finally, we employed this newly generated $\mathbf{Z}$ to synthesize a graph: the result is a network with degree and distance distribution close to the typical values we find in citation networks.

We considered then the Microsoft Academic Graph, extracting a citation network comprising 18 939 155 papers and 189 465 540 citations, that we used as a test bed for some of the mining techniques we saw in the second part of this work, specifically those we derived for estimating the feature-feature matrix.

The goal was duplex: first, to see how well these algorithms perform on real data, since in previous chapter we focused on synthetic data; second, to define and test a measure of the *explainability* of features with respect to a network: with this term, we indicate how much a certain set of features in the nodes of a network can explain the links we see. We employed the area under precision-recall curves, the usage of which we previously described and motivated, to assess the explainability of the different feature sets we considered. The first feature set was the affiliations of the authors of each paper; the second was composed by the "fields of studies" each paper has been associated to.

Our results first confirmed the results we got on synthetic data: the Llama algorithm works definitely better than the Naïve baseline; this suggests that the sigmoid activation function is probably the one closer to reality, and also indicates that the naïve independence assumptions are not tenable in practice.

Our second empirical result was observing a difference in the explainability of the two feature sets: we showed that the ability of predicting links is definitely higher when considering fields of study, with respect to affiliations: our algorithm in fact could obtain an area under precision-recall curve of 0.92 for the first set of features.

Chapter 8

Semantic networks

8.1 Introduction

In this chapter, we will apply our model (chapter 3) and the data mining tools we described in chapter 5 to a real, large-scale, open-accessed semantic network: the Wikipedia Link Graph. Our aim here is double-sided: first, we want to show if our model can properly fit semantic networks, by studying which features should be used in this regards, and how large this feature set needs to be; second, we want to show how our approach can give useful insights in the analysis of such networks: for this reason we will propose an anomaly detection approach to mine surprising connections between concepts in a semantic network.

The rest of this chapter is organized as follows. First, in section 8.2 we will review previous works on semantic networks, and we will frame the Wikipedia graph in this context. Then, in section 8.3.1, we will focus on how a feature-rich semantic network is extracted from Wikipedia: there, we will describe how previous works defined it and employed it, and then we will propose a novel approach to extract meaningful categorization from Wikipedia, allowing us to use categories as features for Wikipedia entities. Next, in section 8.4 we will lay our model on this dataset, defining our set of nodes, links and features with precision and measuring how good our model is in this case. Finally, in section 8.5 we will describe how our model can be employed to find unexpected relations in a semantic network: we will review related literature, describe our approach, and compare it by experiments. Finally, in section 8.6 we will discuss the results obtained in this chapter.

This chapter (especially section 8.5) is largely based on a work presented as a conference paper in *"Web Science 2016"* [8]. Section 8.3, instead, was published in the Wikipedia Workshop during *WWW 2016* [31].

8.2 Related works

Semantic networks are graphs where nodes are concepts and links are semantic relationship. The term "concept" can in some cases be replaced by "entity"; Richard H. Richens in [157] called them *"naked ideas"*. There, the author gave the first description in computer science of semantic

networks: the author proposed semantic networks as a way to provide a common knowledge for machine translation. The will to represent something as vague as "concepts" came from the need of *"an interlingua in which all the structural peculiarities of the base language are removed"*; the semantic net, however, *"bear some obvious resemblances to the linguistic configurations"*. After this first application in machine translation, semantic networks were heavily employed to build systems able to infer and to draw logic conclusions from facts [130], a field called "automatic reasoning". Shapiro [167] used instead the term *"computational philosophy"* while analyzing SNePS [166], or "Semantic NEtwork Processing System". Semantic networks are now very often used as a common type of Knowledge Representation [49], a sub-field of Artificial Intelligence studying formalisms to aid computational reasoning about facts and reality. They form the basis of Semantic Web [22], where nodes and links are supposed to be manually curated by web masters, but they are also employed to store information derived automatically from human-readable sources [14].

Sowa *et al.* [172] discussed many types of semantic networks. Among the most prominent we find *definitional networks*, where the links between concepts describe sub-type relations, usually called "is-a". This type of network makes it possible for propositions to talk about properties of classes of objects. Another important type of semantic networks are *assertional network*; here, links are propositions, stating a factual relationship that exists in reality. Authors [172] also described *hybrid network*, where links can be of various types.

Nowadays, hybrid semantic networks are widely employed, often as an overlapping of a definitional network with an assertional one. *Knowledge graphs*, introduced by van der Vos [185], often belong to this category. It is worth to mention the Google Knowledge Graph [169], used by Google Search to enrich its results with semantic, contextual information; in a public post, Google stated that in 2012 their Knowledge Graph had 570 million concepts. Other companies, such as Microsoft, Yahoo and Yandex have developed their own knowledge graphs as well, for similar purposes.

The vast majority of these knowledge graphs drain information from Wikipedia, the largest free-access collaborative Internet encyclopedia. Since Wikipedia is the largest open encyclopedia in the world, in fact, it has been helpful in creating countless knowledge bases; among the openly accessible ones, we can cite for example DBPedia [14], YAGO [177], and Freebase [34]. In many of these, Wikipedia articles are used as concepts, and the relationships between them are usually derived from the "infoboxes" they contain (infoboxes are a schematic way to represent information, intended for human readers) or even, sometimes, from the natural language itself.

Many researchers, however, used the link structure of Wikipedia directly as a semantic network [71, 143, 176, 179]. Wikipedia link structure offers in fact a very simple yet effective way to provide meaning and context to entities: to know to which entities *"Barack Obama"* is semantically related, it is sufficient to look at the Wiki-links in his article – e.g., *"African American"*, *"Operation Freedom Falcon"*, and *"Obamacare"*; the original article also provides different names for the same entity, such as *"44th President of the United States"*. From this point of view, the link structure of Wikipedia presents all the properties of a semantic network and can be seen as such [15, 85, 89]. Common application for Wikipedia links as a semantic net are found in named

entity resolution [87] and to measure semantic distance [143].

8.3 Wikipedia categories

Folksonomies are collaborative attempts to categorize items of some type, with the aim of helping users in their searches (e.g., to have information on related items or to cluster items that are similar under some viewpoint). Wikipedia is itself endowed with a folksonomy, that takes the form of a category hierarchy: each Wikipedia article is tagged with one or more categories, that are themselves structured in a collaborative hierarchical framework.

Under this point of view, Wikipedia can be seen as a knowledge graph with an explicit, human-authored form of entities classification[1] (the "is-a" relationships, derived from definitional semantic networks). Users interested in mining data from Wikipedia can naturally rely on categories as a further, precious information source [14, 165, 179]. In our case, we will use the categories of Wikipedia articles as our features, while considering the Wikipedia links as our main graph.

Nonetheless, using Wikipedia categories without filtering is problematic, at best: the category hierarchy is extremely sparse and noisy, it contains duplications, errors and oversights, and it is more often than not too fine-grained to be directly employed.

In this section, we will propose an easy, tunable, endogenous technique to cleanse and prune the category hierarchy. After briefly discussing the problems that the hierarchy exhibits, we focus on the usage of centrality measures to identify important categories and show how harmonic centrality [32] outperforms other alternative measures. The method we propose will define a new, cleaner set of features for the entities in the Wikipedia semantic network, to which we will apply our techniques. However, the method explained in this section is general, and can be used fruitfully as a preprocessing phase in every algorithm that wants to exploit categories in mining Wikipedia data.

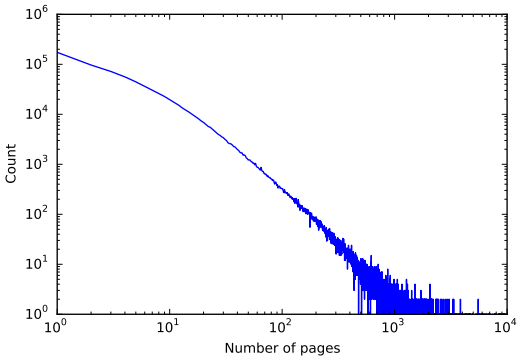
All throughout this chapter, we will employ the `enwiki` snapshot[2] of February 3, 2014 of the English Wikipedia, and in particular:

- the Wikipedia Link graph, composed by 4 514 662 Wikipedia entities, with as many as 110 699 703 arcs[3]; every redirect was merged to its target page;
- the full categorization of pages: a map associating every page to a subset of the 1 134 715 categories;
- the category pseudo-tree: the graph built by Wikipedia editors, with the aim of assigning each category to one or more "parent" categories; it contains 2 215 353 arcs.

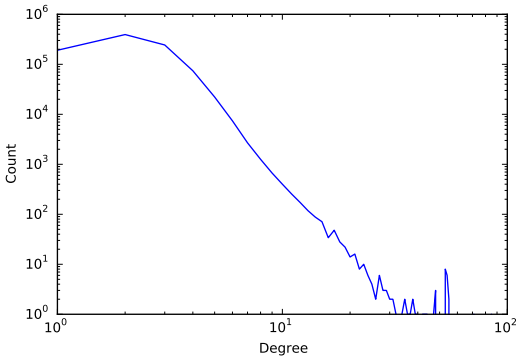
[1] At the time the work described in this chapter was completed, a new project called "Wikidata" [189] was getting a foothold inside the Wikimedia community, in order to build directly a community-curated knowledge graph. In 2016, in particular, Pellissier *et al.* [151] described the development of a tool to import Freebase [34], and other sources, into Wikidata.

[2] This dataset is commonly referred to as `enwiki-20140203-pages-articles` according to Wikipedia naming scheme.

[3] Self-loops were removed.



(a) Distribution of the number of pages per category.



(b) Distribution of the number of sub-categories per category (that is, degree in the category pseudo-tree).

Figure 8.1: Preliminary analysis of Wikipedia categories.

As a preliminary analysis of this dataset, we assessed two global properties of this category structure. In fact, we report in fig. 8.1a the distribution of the number of pages per category, in a log-log scale; in fig. 8.1b, we report the out-degree distribution in the category pseudo-tree; please note that the out-degree of a category in this graph is the number of its sub-categories, so this distribution is equivalent to the distribution of the number of sub-categories per category. Both these properties seem to follow power-law distributions, confirming previous analysis [40, 41].

8.3.1 Related works

Wikipedia articles are endowed with *categories*, intended to group together articles related to similar subjects; categories serve many purposes, like enabling users to browse sets of related articles, or enhancing the automated production of inclusion and navigation boxes.

Many of the works cited in the introduction of this chapter have shown how valuable the category tagging and the Wikipedia Category Graph can be, especially in data mining [70, 102, 107, 165, 176, 179]. The former is the bipartite graph where Wikipedia articles are tagged by ("belong to") one or more categories; if we identify categories as features, it corresponds to the matrix $\mathbf{Z}$ in our model. The latter instead is a new element, and it is the hierarchy specifying how categories are organized in subcategories (i.e., there should be a link between x and y whenever "x is a subcategory of y"). Both of these graphs are completely user-generated and in continuous evolution.

Indeed, the category graph is far from perfect: since the very notion of "subcategory" is fuzzy and no universal policy is strictly enforced, the resulting hierarchy is not a forest, and not even a directed acyclic graph, as pointed out by Kittur *et al.* [117]. The category graph has been described instead as a thesaurus that combines collaborative tagging and hierarchical subject indexing [188] and as an overlay between different trees [161].

In fact, although many works heavily employed categories and the graph they form, all authors had to cope with the extreme level of noise one can find in them. In particular, the fact that the subcategory graph actually contains cycles forces users to take a cleansing step into

Problem solving → Artificial intelligence → Cybernetics → Applied mathematics → Mathematical problem solving → Problem solving

Table 8.1: A cycle in the category pseudo-forest.

careful consideration. Diverse techniques have been tried to do that. Common examples include considering only a tree based in a root category (from a global root [107] or a local one [161, 194]); others have arbitrarily removed cycles [70]. However, to the best of our knowledge, there was no previous work comparing different techniques for this *de-noising* operation.

8.3.2 Noise in Wikipedia categories

The Wikipedia Category Graph expresses a *category hierarchy*, that reflects the notion of "being a subcategory of"; according to the guidelines, the category hierarchy should correspond to a partial order[4], and should therefore be acyclic.

Nonetheless, like the rest of the Wikipedia effort, categories are created and edited collaboratively by users: as a result, the categorization process in Wikipedia is quite noisy and, in fact, a continuous work-in-progress: most importantly, the absence of cycles is neither enforced nor guaranteed. In fact, the *category pseudo-forest* (the directed graph whose nodes are the Wikipedia categories and with an arc from x to y whenever x is a subcategory of y) does contain cycles. Most of them are either consequence of a factual error by the Wikipedia editors or, more commonly, of the fact that the very notion of "being a subcategory" is not precisely defined. An example of a cycle is shown in Table 8.1. An excerpt of the Wikipedia Category Graph, showing 20 categories and the links between them, is pictured in Fig. 8.2.

The presence of cycles is not the only form of noise that we find in the Wikipedia category pseudo-forest. Duplications, misplaced eponymous categories, excessive fragmentation are other problems that make a direct use of the hierarchy difficult at best. This was also highlighted in many previous works [69, 70].

While a very large majority of categories (1 125 823, amounting to 99.22%) lie in a strongly connected component (SCC) of their own, there are some non-trivial SCCs, the largest one composed by 6 833 categories, and the second largest by 105. Let us point out that it means that from each category i belonging to this strongly connected component to any other category j inside it, there is a path of "is a subcategory of" arcs from i to j *and* from j to i!

On the other hand, the graph itself is reasonably connected in the weak sense, with 952 833 (83.97%) nodes belonging to one single weakly connected component (WCC); most of the remaining categories (171 889, amounting to 15.15%) are isolated nodes (the 45% of these nodes are related to specific years, like "1833 births" or "Populated places established in 1864").

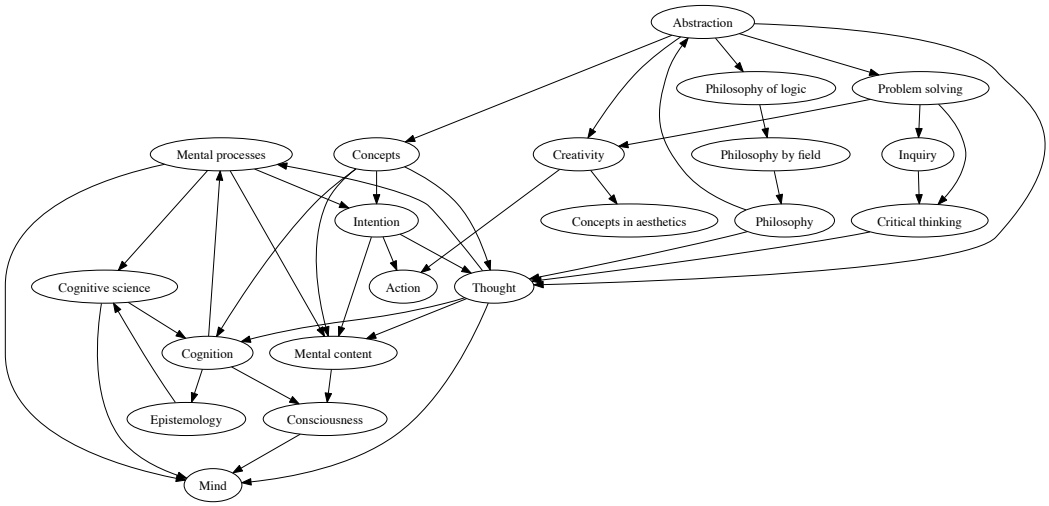


Figure 8.2: Sample from the Wikipedia Category Graph, highlighting the presence of cycles.

8.3.3 Cleansing Wikipedia categories

It is natural to try to employ the Wikipedia articles belonging to each category as a mean to obtain a cleaner category hierarchy; for example, one may forget about the hierarchy structure altogether and try to reconstruct (a less noisy version of) it from the sets of articles belonging to each category. This approach, besides throwing away a large amount of human-cured data, has a quite serious (theoretical and practical) limitation: if one wants to use the category hierarchy to enrich the information on articles, using articles to get the category hierarchy is by all means a catch-22. In our case, specifically, we would like to select the set of features of our graph independently from the nodes they will apply to, in order to cleanly test our model about the node-feature association on a well-defined set of features.

For these reasons, in fact, our cleansing technique is completely *endogenous* (i.e., it uses only the information contained in the category pseudo-forest), and it consists of three phases.

Phase 1: Milestones determination. In the first phase, we select the m topmost categories that we want to preserve (called *milestones*). Since this set of categories will be, in the end, the set of features F of our feature-rich graph, we use the notation m to indicate its cardinality, as we did in the previous chapters. For the moment, here, m is a parameter that determines the granularity of the output hierarchy, and can be set arbitrarily. Categories are then ranked according to their centrality in the category pseudo-forest. The choice of which centrality measure [32] should be adopted will be discussed below. While selecting the m topmost categories, we expunged utility categories such as *"Categories by country"* and *"Main topic classifications"*.

[4]"Categories are organized as overlapping "trees", formed by creating links between inter-related categories (in mathematics or computer science this structure is called a partially ordered set)." [In `https://en.wikipedia.org/wiki/Wikipedia:Categorization`]

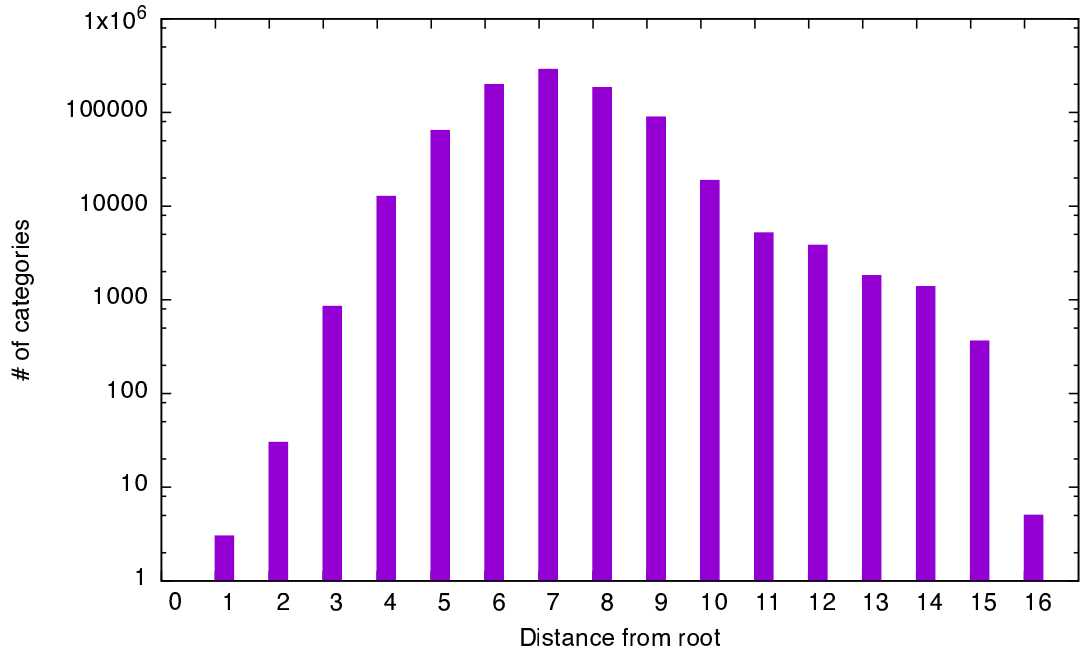


Figure 8.3: The number of categories depending on the distance from the root category "Article". 56% of the categories are at distance 6 or 7.

Note that the milestones determined in this way have a natural hierarchy, that is the one induced from the original pseudo-forest, throwing away the hierarchical arrows that do not match the centrality score chosen (a supercategory cannot be less important than its subcategories). In other words, given two milestone categories x and y, we postulate x is a subcategory of y iff it was marked as a subcategory in the original pseudo-forest and the centrality score of x is smaller than that of y. This reconstruction process guarantees that the resulting hierarchy is acyclic.

Phase 2: Category remapping. Once the milestones categories have been determined, each category is mapped to the closest reachable milestone (i.e., to the milestone category that is the more specific generalization of the category under examination). Categories for which no milestone is reachable are *orphan*, and they in fact disappear. The (partial) map from categories to milestones $\iota(-)$ is called the *category remapping*.

Phase 3: Article categorization. At this point, each Wikipedia article is re-assigned to the remapped categories it belongs; more precisely, if an article was marked as belonging to categories $c_1, \ldots, c_k$ it is now mapped to the milestone categories $\iota(c_i)$ $(i = 1, \ldots, k)$ for which $\iota(-)$ is defined. If all the categories it belonged to are orphan, the article itself remains an orphan. This phase is not strictly part of the category-hierarchy cleansing, but it is necessary to use the cleansed categories as features for Wikipedia articles.

	Lin	Harmonic	PageRank	Katz	Indegree	Root distance	Closeness	Popularity
Lin	1.0000	0.9924	0.9539	0.9653	0.9497	0.6102	0.4010	0.1356
Harmonic	0.9924	1.0000	0.9545	0.9679	0.9490	0.6657	0.3665	0.1127
PageRank	0.9539	0.9545	1.0000	0.9547	0.9392	0.5345	0.4575	0.1683
Katz	0.9653	0.9679	0.9547	1.0000	0.9924	0.6200	0.4222	-0.0786
Indegree	0.9497	0.9490	0.9392	0.9924	1.0000	0.5646	0.4461	-0.0770
Root dist.	0.6102	0.6657	0.5345	0.6200	0.5646	1.0000	0.5810	0.3014
Closeness	0.4010	0.3665	0.4575	0.4222	0.4461	0.5810	1.0000	0.3842
Popularity	0.1356	0.1127	0.1683	-0.0786	-0.0770	0.3014	0.3842	1.0000

Table 8.2: The weighted Kendall's τ [186] between the centrality measures under comparison. Darker shades of gray indicate higher Kendall't τ correlation.

Centrality measures. The most important step in our cleansing procedure, that crucially determine its output, is the selection of the centrality score to be used. Note that we are here sticking to our principle of endogeneity and want therefore to avoid selecting important categories based on notions that are not internal to the category hierarchy itself.

Therefore, we took into consideration the network centrality measures that more widely used and appear to be better-behaved for these kinds of problems [32], namely, indegree (number of incoming arcs), closeness centrality, Lin's index, harmonic centrality, PageRank (with damping factor $\alpha = 0.85$), Katz's index (with parameter $\beta = (2\lambda)^{-1}$ where λ is the spectral radius). To compute geometric centralities in an efficient way, we employed HyperBall [33].

For comparison, we also considered the non-endogenous metric defined by category popularity (i.e., number of articles belonging to that category).

A further, widely used, endogenous measure that was sometimes proposed for Wikipedia categories is their distance from the root category ("Articles") [107]. Albeit simple and natural, this measure has a number of drawbacks: first of all, it has a very limited granularity and a huge number of ties (see Figure 8.3). Moreover, the distance from the root is easy to spam and not very robust: one single misplaced subcategory link is enough to make a category more (or less) important than it should be. While the latter observation is probably irrelevant for the very top levels of the hierarchy (where errors and inconsistencies are easily spotted and corrected), the more crowded lower levels are certainly problematic.

Experimental comparison. The availability of many different ways to rank the categories immediately raises two problems: the first is whether (and how much) those ranking techniques differ in choosing the topmost categories; the second, in case the rankings are significantly different, is to understand which one is more suitable for our needs.

In order to answer the first question, we compared the various rankings using a variant of the classical Kendall's τ [186] that treats differently discordances depending on whether they happen at the top or at the bottom of the rankings, still handling ties in a proper way.

As Table 8.2 shows, there is a group of measures (Lin's index, harmonic centrality, PageRank, indegree and Katz's index) that strongly correlate to one another especially at the top, but appear to be much uncorrelated to the "Distance from the root" and, even more evidently, from "Closeness" and "Popularity".

These differences make it urgent to answer the second question raised above, that is, which

Centrality	AP	NDCG	AUC
Lin's index	**0.14641**	**0.71230**	0.94324
Harmonic centrality	**0.13914**	**0.70149**	**0.94444**
PageRank	0.07411	0.64503	**0.95001**
Distance from the root	0.05339	0.60720	0.92983
Katz's index	0.01606	0.53651	0.92708
Indegree	0.00917	0.50636	0.91532
Category popularity	0.00491	0.47994	0.90082
Closeness centrality	0.00083	0.40392	0.65134

Table 8.3: Comparison of various rankings in retrieving the golden-truth categories from the Library of Congress Classification, highlighting the best two rankings for each metric. We repeated the computation with many different shuffles of the tied scores, to see how ties influenced the result: it turned out that ties do not impact on the relative ordering of the measures; in fact the variance of the accuracy scores computed is quite limited, except for "Distance from root" (where, for example, the average precision ranged from 0.050686 to 0.056097).

measure seems more "correct"; to answer this question, though, we need some ground truth on which categories are "relevant" in a broad sense.

Following Salah *et al.* [161] we decided to use an expert-curated bibliography classification. We decided to make use of the Library of Congress Classification[5], (LoCC) through the outline (main classes and sub-classes) available from within Wikipedia. This choice allowed us to employ Wikipedia itself[6] to map LoCC classes to Wikipedia categories in the following way.

For each listed LoCC class (e.g., "Philosophy"), we followed the hyperlink (if any) to the related Wikipedia article (dropping the word "Outline" if needed; e.g., for the category tag "Philosophy" we ended up to the "Philosophy" article of Wikipedia); then, we joined all the categories of the article[7] ("Philosophy", "Academic disciplines", "Humanities" etc.). This process resulted in a set of 682 golden-truth categories assumed to be high-ranked, and we computed for each ranking the average precision (AP), the Normalized Discounted Cumulative Gain (NDCG), and the Area Under ROC curve (AUC) [129] in retrieving these categories; the results are displayed in Table 8.3, showing that Lin's index and harmonic centrality appear to be the best techniques under this viewpoint.

From the discussion so far we can conclude that harmonic centrality and Lin's index are the best centrality measures to identify milestones categories; we hereby preferred the former over the latter because harmonic centrality is more natural and it enjoys better theoretical properties [32].

8.3.4 Cleansing results

As justified above, harmonic centrality is apparently the most powerful among all centrality measures we tried. To give an idea about the effectiveness of harmonic centrality in capturing the generality of categories, we report in Table 8.4 the first and the last categories on our list,

[5] `http://www.loc.gov/aba/cataloging/classification/`

[6] `https://en.wikipedia.org/wiki/Library_of_Congress_Classification`

[7] As a more restrictive alternative, we only considered the category whose name matched that of the article, provided that it was present; this alternative approach produced a much smaller, less noisy, set of categories (205 instead of 682) but yielded essentially the same results.

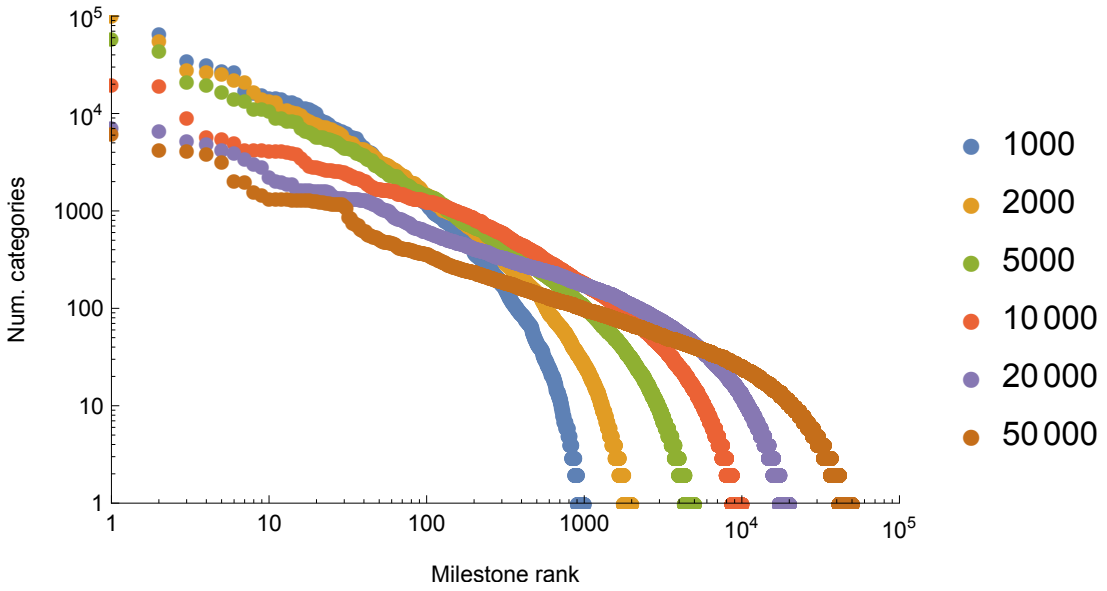


Figure 8.4: The milestone rank-size distribution for various choices of m, using harmonic centrality: for a fixed m, we sort the milestone categories c according to the decreasing size of $|\iota^{-1}(c)|$ (i.e., the number of categories that are remapped to c) and show how those values decrease. The most popular milestones attract less pages, in favor of a more balanced category remapping.

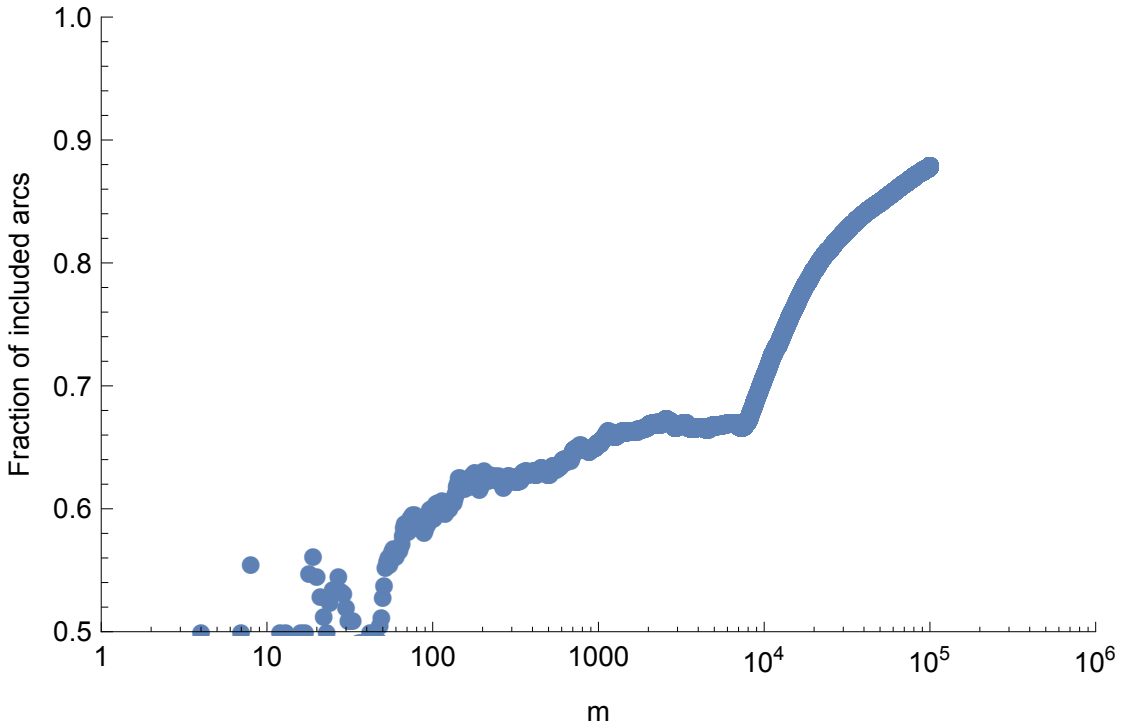


Figure 8.5: Fraction of arcs retained from the pseudo-forest induced by the m nodes with largest harmonic centrality: for a fixed m, we select the pseudo-forest that includes only the top m nodes; from this graph, we discard the arcs $x \to y$ such that the score of x is larger than the score of y (as explained in Section 8.3.3).

Rank	Category
1	Countries
2	Society
3	Nationality
4	Political geography
5	Culture
6	Humans
7	Social sciences
8	Structure
9	Human–geographic territorial entities
10	Contents
11	Geographic taxonomies
12	Fields of history
13	Places
14	Humanities
15	Continents
16	Political concepts
17	Human geography
18	Subfields of political science
19	Articles
20	Subfields by academic discipline

Rank	Category
19981	Maldives
19982	Government buildings...
19983	Illinois waterways
19984	Bodies of water of Illinois
19985	2002 in association football
19986	Electronica albums by British artists
19987	Visitor attractions in Arkansas by county
19988	Years of the 20th century in Europe
19989	Commonwealth Games events
19990	Albums by English artists by genre
19991	American football in Pennsylvania
19992	Ethnic groups in Poland
19993	Card games
19994	Central African people
19995	Deaths by period
19996	Visitor attractions in Vermont
19997	Ancient roads and tracks
19998	People in finance by nationality
19999	Populated places in Greater St. Louis
20000	Religion in Poland

Table 8.4: Topmost and bottommost twenty Wikipedia categories (with $m = 20\,000$) according to their harmonic centrality in the Wikipedia category pseudo-forest.

Original category c	**Substitution milestone** $\iota(c)$
Southern Tang poets	Poets by nationality
Antsiranana Province	Country subdivisions of Africa
Fellows of Magdalen College, Oxford	University of Oxford
Actresses from Greater Manchester	Greater Manchester
Guyanese slaves	History of South America
Swiss manuscripts	Swiss culture
Wilson Pickett songs	Songs by artist
Baroque architecture in Austria	Baroque architecture by country
Eastern Collegiate Roller Hockey Association	♯
Art schools in Washington (state)	Washington (state) culture
Rivers of Kostroma Oblast	Rivers by country
Flamenco compositions	Spanish music
Oil fields of Gabon	Geology of Africa
Basketball teams in Georgia (U.S. state)	Basketball teams in the United States by state
2004 in Australian motorsport	2004 in sports
Populated places established in 1821	♯
Elections in Southwark	Local government in London
Permanent Representatives of Norway to NATO	Ambassadors of Norway
Basketball in Turkey	Basketball by country
Balli Kombëtar	♯

Table 8.5: An excerpt of the category remapping process (using harmonic centrality with $m = 20\,000$ milestones). We write ♯ if there is no milestone category reachable from c.

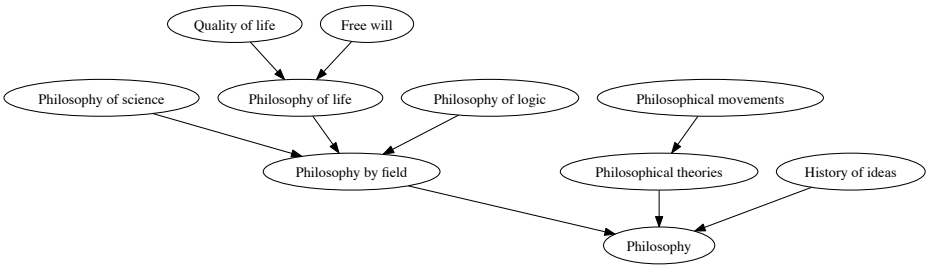


Figure 8.6: Sample from the cleansed category hierarchy.

when $m = 20\,000$. In Table 8.5 we show some examples of the induced category remapping: on average, each article belongs to 4 categories. As the examples show, this cleansing process yields very clean remapped features. Figure 8.4 shows the rank-size distribution of the number of categories remapped to each milestone category for various choices of m.

The choice of m depends on the level of granularity we desire in the final output. Depending on the application, we may want different levels of aggregation among categories. This number will end up to be the number of features in our feature-rich graph; as we said in section 3.2.3, it will represent a candidate intrinsic dimensionality for the graph. We will consider various choice of $m = |F|$ for our model in section 8.4.

Although not directly relevant to our goals, we also tested the cycle-removing procedure that we explained in Section 8.3.3: by combining the original pseudo-forest with the total order given by our rank, we can obtain a Directed Acyclic Graph. In Figure 8.5 we show how many arcs

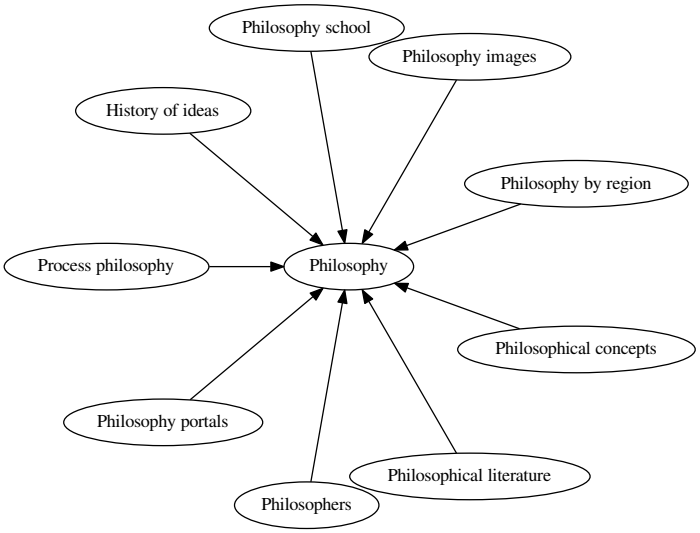


Figure 8.7: For comparison, sample from the cleansed category hierarchy if "distance from the root" was used to select milestone categories.

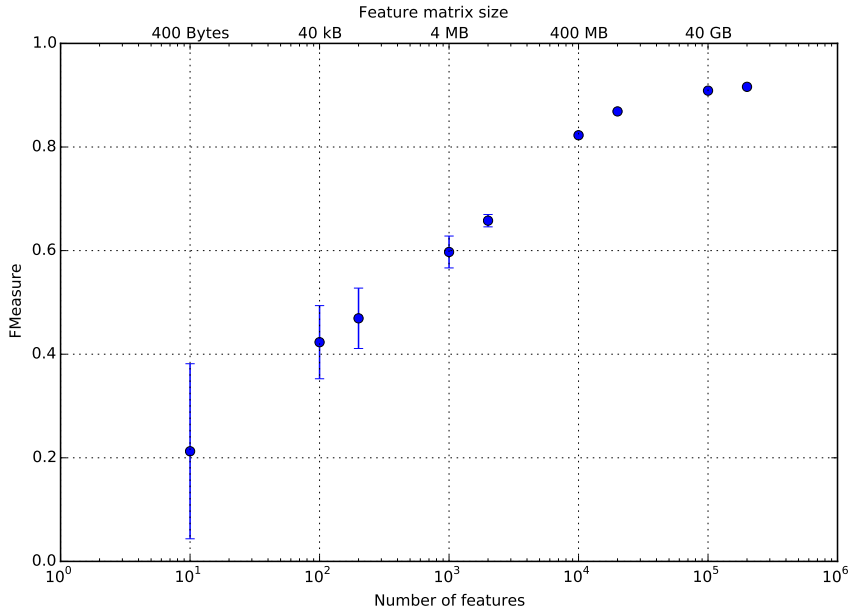


Figure 8.8: F-Measure of Alg. 3 in 10-fold cross-validation, with different number of considered categories. On the X-axis, we report both the cardinality of the considered categories set, and the size required to store an explicit, dense representation of $\mathbf{W}$. The depicted error bars represent two standard deviations measured on the ten folds.

are discarded by our approach, with respect to the number m of categories to preserve. The plot indicates that with $m < 10\,000$ categories, the fraction of discarded arcs is approximately between 30% and 40%; increasing m, we discard fewer and fewer arcs. In other words, for the top thousands of categories removing cycles means discarding a significant amount of arcs: the difference between the cleansed and the original hierarchy is not trivial.

In Figure 8.6 we depict a sample extracted from the cleansed category hierarchy, showing the first (in order of harmonic centrality) ten direct or indirect subcategories of "Philosophy". For comparison, Figure 8.7 shows how the same sample would appear if we had used "Distance from the root" as measure of centrality.

8.4 Latent category interaction

We are now ready to bring into play our model (section 3.2) on this Wikipedia dataset. In particular:

- our nodes N will be the 4 514 662 Wikipedia entities;
- our graph G will be the Wikipedia Link graph (110 699 703 links);
- our feature set F will be the cleansed set of categories – whose number will be reduced from 1 134 715 to m (to be defined later);

Measure	Results
Accuracy	$87.3\% \pm 0.06$
Precision	$90.0\% \pm 0.05$
Recall	$84.0\% \pm 0.14$
F-Measure	$86.9\% \pm 0.07$

Table 8.6: Average and standard deviation of results from a 10-fold cross-validation of Alg. 3 on the links of Wikipedia, with $m = 20\,000$.

- our node-feature matrix $\mathbf{Z}$ will be the cleansed categorization of pages described before, associating every Wikipedia article $i \in N$ to a set $F_i \subseteq F$ of (cleansed) categories.

Now, we want to fix m in order to obtain a good fit for our model. In order to do this, let us employ our *Llama* algorithm (Alg. 3) to find $\mathbf{W}$, for some different values for m. On this dataset, $\mathbf{W}$ will represent how Wikipedia categories link to each other: we expect, for example, a high value for $W_{\textit{U.S. governors},\ \textit{U.S. states}}$ (but maybe not the other way around). As in all our real-world examples, $\mathbf{W}$ can be only estimated, since it is not directly observable.

We will also employ the row-normalized version of our model we have described on page 20; in this case in fact it is quite natural, since it corresponds to articles belonging to fewer categories providing stronger signals than those that belong to many categories. This kind of normalization often gives better results in practice for learning [58]. We will therefore adopt the corresponding version of *Llama*, described by (5.7).

In order to choose a value of m (the number of categories we are going to employ) let us remind that we can see our deterministic model (described by (5.3) in section 5.4) as a binary classifier: we are making it learn a matrix $\mathbf{W}$ on a set of node pairs, and then measuring how well the model is behaving in predicting if a new node pair (i, j) forms a link. In doing so, we used a 10-fold cross-validation technique to assess how well the algorithm is generalizing: specifically, we divided in 10 folds the space of node pairs $N \times N$. We report the results of this experiment for different values of m in Fig. 8.8. There, we also report how much space we need in practice to store explicitly a dense representation of the learned matrix $\mathbf{W}$.

From these results, we see that we have a jump in the robustness of our model by setting $m = 10\,000$ and an even better behavior with $m = 20\,000$. Larger values of m still slightly increase the fitness of the model, but for them the space needed becomes hardly manageable, since it grows with m^2. For this reason, we chose $m = 20\,000$ as a good compromise between the significance of these categories and the needed resources.

Further results for this set F of 20 000 categories, reported in Table 8.6, are consistent with our expectations. In particular, the average of F-Measure on unknown node pairs is 86.9%, showing that the model we have learned is robust and not threatened by overfitting. Learning on the whole graph, the ratio $|L \cap L^*|/|L|$ – that is, how many existing links are explained by $\mathbf{W}$ – is equal to 86%. With this set of features, on average, each Wikipedia article belongs to 4 categories.

Let us remark how the *Llama* algorithm scales very well with the size of data: running it on the whole Wikipedia graph (110 699 703 arcs, plus an equal number of non-linking pairs) with

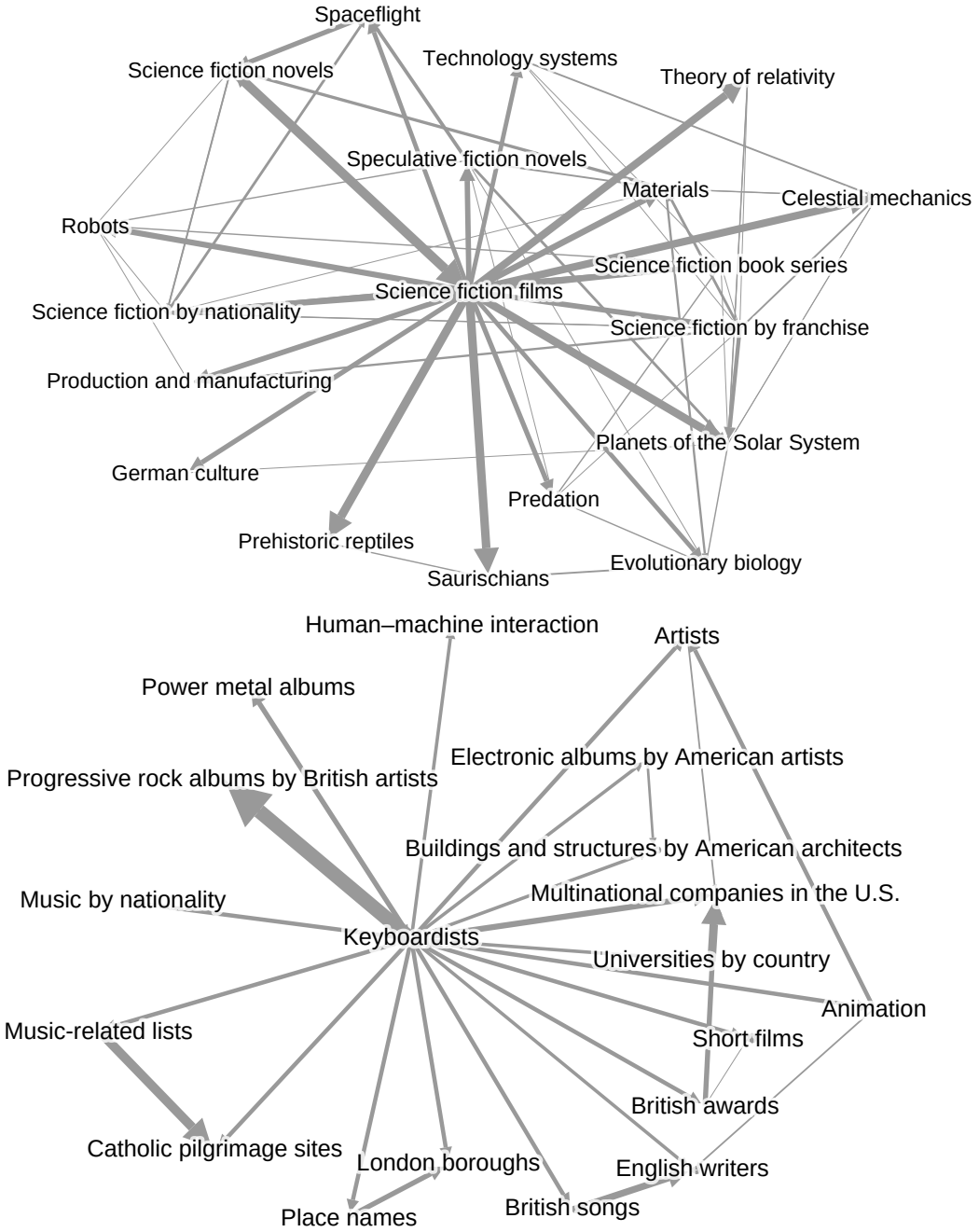


Figure 8.9: Two fragments of the latent category graph induced by Llama matrix $\mathbf{W}$, representing the 18 closest neighbors of categories "Science Fiction Films" and "Keyboardists", respectively. The width of the arc from c to c' is proportional to $W_{c,c'}$, and arcs with $W_{c,c'} \leq 1$ are not shown.

the categorization we chose (20 000 categories) required only 9 minutes on an Intel Xeon CPU with 2.40GHz.

The matrix $\mathbf{W}$ we extracted in this way can be as well seen as a weighted, directed graph among categories (a *latent* graph, behind the observable link graph). In this way, we illustrated in Fig. 8.9 some fragments of $\mathbf{W}$. In the picture, we display the 18 neighbors closer to two starting categories ("Science Fiction Films" and "Keyboardists"); the width of the arc from c to c' is proportional to $W_{c,c'}$, and arcs with $W_{c,c'} \leq 1$ are not shown. For example, from the picture it is clear that a link from a page of a science-fiction film to a page of a science-fiction novel is highly expected, as it is one from a page of a keyboardist to one of a British progressive rock album. From these representations we can catch a glimpse of how this method is able to build a model for the graph, capturing meaningful relations between categories.

The rougher version of the same neighborhood as induced by the Naive Bayes algorithm presented in section 5.3 is shown in Figure 8.10: even from this small example, it is clear that the naive version introduces more noise (epitomized by the inclusion of "Language of the Caribbean" and "Languages of Singapore" among the 18 closest neighbors of "Science fiction films").

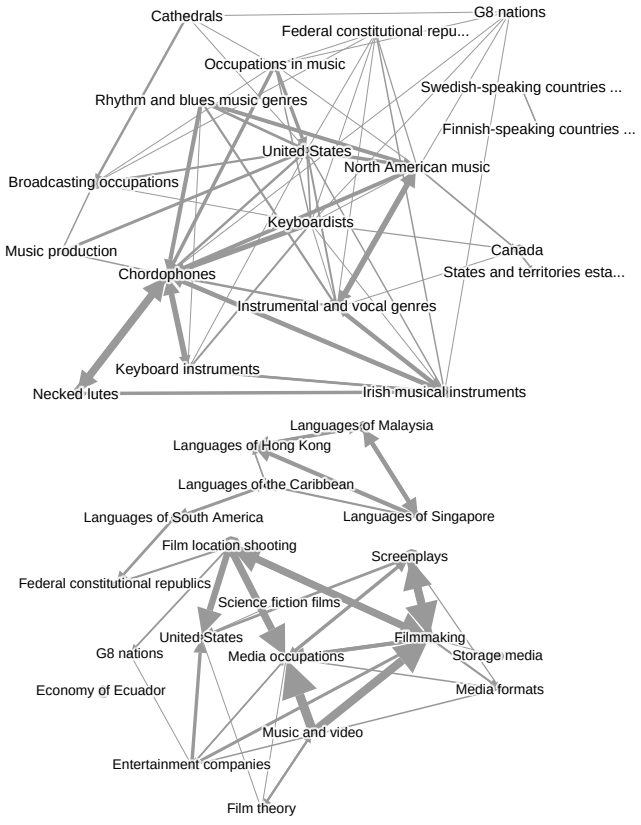


Figure 8.10: Two fragments of the latent category graph induced by our Naive approach matrix (section 5.3), representing the 18 closest neighbors of categories "Keyboardists" and "Science Fiction Films", respectively. The width of the arc from c to c' is proportional to $W_{c,c'}$, and the lighter arcs are not shown. For comparison with the *Llama* algorithm, see Figure 8.9.

8.5 Finding unexpected relations

Besides finding trends and unveiling typical patterns, modern information retrieval is increasingly more interested in the discovery of serendipity and surprising information in textual datasets. In this section, we will focus on finding *unexpected links* in hyper-linked document corpora when documents are assigned to categories, using Wikipedia as our case study. We will show that our model provides better accuracy in finding *surprising* information than most existing text-based techniques, with higher efficiency and relying on a much smaller amount of information.

In general, data mining (text mining, if the data involved take the form of textual documents) aims at extracting potentially useful information from some (typically unstructured, or poorly structured) dataset. The basic and foremost aim of data mining is discovering frequent patterns, and this problem attracted and still attracts a large part of the research efforts in this field. Nonetheless a quite important and somehow dual problem is that of finding unexpected (surprising, unusual, new, unforeseen...) information; it is striking that this line of investigation did not receive the same amount of attention. Albeit there is some research on the determination of surprising information in textual corpora (most often based on the determination of outliers in the distribution of terms or n-grams) there is essentially no work dealing with *unexpected links*. Even if some of the previous proposals exploiting text features can be adapted to this case, a simpler (and, as we here show, more effective) way to approach this problem is by using link prediction algorithms [128], stipulating that a link that is difficult to predict is unexpected.

We will prove that the availability of some form of *features* of linked documents can significantly improve the techniques described, leading to algorithms that are extremely efficient, use much less information than text-based methods, and offer better precision/recall trade-offs also compared to link prediction; we will also show that the two methods are in fact orthogonal to each other, and can be fruitfully combined.

It is worth noting that the discovery of unexpected links offers a chance to find unknown information: given a certain document, we can highlight text snippets containing unexpected links. Furthermore, in the case of knowledge graph (such as the Wikipedia link structure), it means finding unexpected relations in reality itself (or at least in the part modeled by the knowledge graph under consideration).

8.5.1 Related works

The most standard way to find *unexpected* links would be to employ an algorithm of link prediction [128] to identify links that are *expected*: the expectedness of a link e in a network G is the likelihood of the creation of e in $G - \{e\}$. In this sense, the problem of finding unexpected links is in some sense the dual of link prediction: any algorithm that aims at solving link prediction can be used to find unexpected links. In fact, we will later show that state-of-the-art link prediction algorithms [1] are very good at evaluating the (un)expectedness of links.

However, while a link prediction algorithm typically employ only the information coming from the network, other approaches—more explicitly tailored for the retrieval of unexpected information—could exploit other sources. Many approaches employed textual information; our

goal will be instead to make use of the categorical information we obtained, together with the model we presented.

Among the first authors trying to consider the problem in the context of text mining we cite Liu, Ma, and Yu [126]. In their work, two supposedly similar web sites are compared (ideally, two web sites of two competitors). The authors first try to find a match between the pages of the two web sites, and then propose a measure of unexpectedness of a term when comparing two otherwise similar pages. All measures are based on term (or document) frequencies; unexpected links are also dealt with but in a quite simplistic manner (a link in one of the two web sites is considered "unexpected" if it is not contained in the other). Note that finding unexpected information is crucial in a number of contexts, because of the fundamental role played by serendipity in data mining (see, e.g., [141, 155]).

This unexpectedness measure is taken up by Jacquenet and Largeron [101], with the aim of finding documents that are similar to a given set of samples and ordering the results based on their unexpectedness, using also the document structure to enhance the measures previously defined [126]. Finding outliers in web collections was also considered by other authors [3]; dissimilarity scores are computed based on word and n-gram frequency.

Some authors approach the strictly related problem of determining lacking content (called *content hole* in [142]) rather than unexpected information, using Wikipedia as knowledge base. A similar task is undertaken by [66], this time assuming the dual approach of finding content holes in Wikipedia using the web as a source of information.

More recently, [184] considers the problem of finding unexpected related terms using Wikipedia as source, and taking into account at the same time the relation between terms and their centrality.

To our knowledge, no algorithms explicitly employing categorical information was previously employed for this task. As we will show, it turns out that the signal obtained from the latent category matrix is even better and partly orthogonal to the one that comes from the graph alone (through state-of-the-art link prediction), and combining the two techniques greatly improves the accuracy of both.

8.5.2 Using our model to find outliers

Our idea is that if the documents within a linked corpora are tagged with features, one can learn how the feature-feature pairs influence the presence of links, and as a consequence determine which links are unusual (in the sense that they are not "typical"). This corresponds to applying our model to this scenario, and then marking the outliers with respect to the model as *unexpected*. In this way, the model found by our classifier will be used to detect anomalies [2].

Precisely, we will use the matrix $\mathbf{W}$ found by *Llama* (as shown in the previous section) to assign an expectedness score to each link, following the equation of the model behind the *Llama* algorithm (section 5.4). In fact, since equation (5.3) should give us a positive value for pair of nodes (i, j) predicted to be links, and a negative value for those who should not, we can use it as an expectedness score:

$$expectedness(i, j) := \sum_{h \in F_i} \sum_{k \in F_j} W_{h,k} \tag{8.1}$$

And, obviously, we can use its opposite as an *unexpectedness* score. We will call this approach *LlamaFur*, "Learning LAtent MAtrix to Find Unexpected Relations". We will also call the same approach but with the matrix learned in the naïve way (section 5.3) *Naïve LlamaFur*.

The intuition behind this method is that we hypothesize our model can capture what a human reader would consider "usual" and what would be considered "surprising" in a better way than other methods: that, in other words, what is unexpected by our model in a semantic network, would be unexpected as well for a human. For example, documents of the category "Actor" often contain links to documents of the category "Movie". The fact that George Clooney used to own Max, a 300-pound pig, for 18 years, presents itself as a link from an "Actor" to an entity belonging to the category "Pigs"/"Coprophagous animals", which is atypical in the sense above. We will put this assumption under test in the next section.

We wish to remark that the *LlamaFur* approach could be in principle applied to a plethora of different kinds of objects. The only assumptions on the input are the same of our model: a (possibly directed) graph, and meaningful features on its nodes; features can be overlapping as well, so in fact they may just be some observed properties of each object. These assumptions are quite general, and could be useful in many real-world use cases, from the detection of unexpected collaborations between grouped individuals to finding surprising travel habits from geo-tagged data. Other possible applications include finding unusual patterns for fraud detection [54] and data forensics [154].

8.5.3 Experimental results

We therefore proceeded to evaluate the unexpectedness of links in Wikipedia with *LlamaFur*. For example, for each page, we can find the most unexpected links it contains: we conjecture that these links will represent *surprising* information. An example of the largest and lowest scores for two Wikipedia articles is provided in Table 8.8 & 8.7.

Evaluation methodology. Now, we want to evaluate the effectiveness of LlamaFur in mining unexpected links using a standard approach commonly adopted in Information Retrieval. In our context, a *query* is a document, the possible *results* are the hyperlinks that the document contains, and a result is *relevant* for our problem if it represents an unexpected link. The scenario we have in mind is that of a user wishing to find surprising links in a certain Wikipedia page. What we are trying to assess is how well LlamaFur can identify an unknown set of unexpected links, having full knowledge of graph and categorization of nodes.

In order to compare the results obtained by LlamaFur with the existing state-of-the-art for similar problems, we performed a user study based on the same pooling method adopted for many standard collections such as TREC (`trec.nist.gov`): we considered a random sample of 237 queries (i.e., Wikipedia documents); for each query we took, among its t possible results (i.e., links), the top-$\lfloor \alpha \cdot t \rfloor$ most unexpected ones according to each system under comparison (see

Links of **Kim Jong-il** *(supreme leader of North Korea from 1994 to 2011)*			
Most Unexpected		**Most Expected**	
Elvis Presley	In a 2011 news story, The Sun reported Kim Jong-il was obsessed with Elvis Presley. His mansion was crammed with his idol's records and his collection of 20,000 Hollywood movies included Presley's titles – along with Rambo and Godzilla. He even copied the King's Vegas-era look of giant shades, jumpsuits and bouffant hairstyle.	**George W. Bush**	Kim's regime argued the secret *[nuclear]* production was necessary for security purposes – citing the presence of United States-owned nuclear weapons in South Korea and the new tensions with the United States under President George W. Bush.
Michael Jordan	Kim reportedly enjoyed basketball. Former United States Secretary of State Madeleine Albright ended her summit with Kim by presenting him with a basketball signed by NBA legend Michael Jordan.	**Kim Il-sung**	He succeeded his father and founder of the DPRK, Kim Il-sung.

Table 8.7: Most expected and most unexpected links in the Wikipedia article *Kim Jong-il*, according to LlamaFur.

Links of **Jupiter**			
Most Unexpected		**Most Expected**	
Inquisition	*[Observation of Jupiter moons]* was a major point in favor of Copernicus' heliocentric theory of the motions of the planets; Galileo's outspoken support of the Copernican theory placed him under the threat of the Inquisition.	**Galileo Galilei**	Galilean moons were first discovered by Galileo Galilei in 1610.
Proto-Indo-European language	*[Jupiter]* name comes from the Proto-Indo-European vocative compound *Dyeus-pater (nominative: *Dyeus-pater, meaning "O Father Sky-God", or "O Father Day-God").	**E. E. Barnard**	In 1892, E. E. Barnard *[an American astronomer]* observed a fifth satellite of Jupiter with the 36-inch (910 mm) refractor at Lick Observatory in California.
Gan De	Gan De, a Chinese astronomer, made the discovery of one of Jupiter's moons in 362 BC with the unaided eye.	**Ptolemy**	*[Ptolemy]* constructed a geocentric planetary model based on deferents and epicycles to explain Jupiter's motion relative to the Earth.
Fish	In 1976, before the Voyager missions, it was hypothesized that ammonia or water-based life could evolve in Jupiter's upper atmosphere. This hypothesis is based on the ecology of terrestrial seas which have simple photosynthetic plankton at the top level, fish at lower levels feeding on these creatures, and marine predators which hunt the fish.	**Jupiter (mythology)**	The Romans named the planet after the Roman god Jupiter.

Table 8.8: Most expected and most unexpected links in the Wikipedia article *Jupiter*, according to LlamaFur.

below); all the resulting links were evaluated by human beings. We set $\alpha = 0.1$, and obtained about 3 698 links.

The human evaluators were asked to categorize each link into one of four classes ("totally expected", "slightly expected", "slightly unexpected" and "totally unexpected"). They were provided with the first paragraph of the two Wikipedia pages, and a link to the whole article if needed. The resulting dataset of 3 698 evaluated links is available for download[8] and inspection. (To obtain a sufficiently large number of evaluated links in a short amount of time, there was virtually no overlap between the links that the evaluators worked on; however, we manually inspected the dataset to find that the labels produced are quite robust—we invite the reader to do the same.) After the human evaluation, we only considered the queries that have at least one irrelevant ("totally/slightly expected") and one relevant ("totally/slightly unexpected") result according to the evaluation, obtaining a dataset with 117 queries. In this dataset, on average each query has 3.48 relevant results over 20.9 evaluated links. About 58.1% of the links were labeled as "totally expected", only 2.2% were "totally unexpected" and about 8.8% were labeled as "unexpected".

It is evident how unexpected links are very sparse (many pages did not present any unexpected link at all); this motivated us to employ bpref [38], a well-suited information retrieval measure. To compute it, we followed TREC specifications[9]; in a nutshell, bpref computes an index of how many judged relevant documents are retrieved ahead of judged non-relevant documents.

Baselines and competitors. In our comparison, LlamaFur is tested in combination and against a number of baselines and competitors. In particular, we considered LlamaFur and its naive variant, Naive-LlamaFur, along with some of the other (un)expectedness measures proposed in the literature.

All of those methods try to measure the unexpectedness of a document d among a set of retrieved documents R. In our application, we are considering a link (d', d) and taking R to be the set of all documents towards which d' has a hyperlink.

- *Text-based methods.* In the literature, all of the measures of unexpectedness are based on the textual content of the document under consideration.
  - The first index, called $M2$ by its authors [101] (a better variant of $M1$, the measure previously proposed the same authors [126]), is defined as:

    $$M2(d) = \frac{\sum_t U(d, t, R)}{m}$$

    where m is the number of terms in the dictionary, and $U(d, t, R)$ is the maximum between 0 and the difference between the normalized term frequency of term t in document d and the normalized term frequency of t in R (the set of all retrieved documents). The normalized term frequency is the frequency of a term divided by the frequency of the most frequent term.
  - The second index, called $M4$ [101] (and proved to works better than $M2$ in previous work), is the

[8] `http://git.io/vmChm`
[9] `http://trec.nist.gov/pubs/trec16/appendices/measures.pdf`

Algorithm	Average bpref	Input data
AA	0.286	graph
$M2$	0.179	bag of words
$M4$	0.293	bag of words
Naive-LlamaFur	0.251	graph, categories
LlamaFur	0.343	graph, categories
LlamaFur + AA	0.350	graph, categories

Table 8.9: Average values for bpref.

$$M4(d) = \max_t \mathrm{tf}(t) \cdot \log \frac{|R|}{\mathrm{df}(t)}$$

where $\mathrm{tf}(t)$ is the normalized term frequency of term t in d, and $\mathrm{df}(t)$ the number of documents in R where t appears.

- *Link-prediction methods.* A completely different, alternative approach to the problem is based on *link prediction*: how likely is it that the link (d', d) is created, if we assume that it is not there? Among the many techniques for link prediction [128], we tested the well-known *Adamic-Adar index* [1] (AA, in the following), defined[10] by

$$AA(d, d') = \sum_{d'' \in \Gamma(d) \cap \Gamma(d')} \frac{1}{\log |\Gamma(d'')|},$$

where $\Gamma(d)$ is the set of documents which d links to.

- *Combinations.* Besides testing all the described techniques in isolation, we tried to combine them linearly. Since each unexpectedness measure exhibits a different scale, we first need to normalize each measure by taking its *Studentized residual*[11] [56]. It is worth mentioning here that other previous methods [96, 136] could also be used as link-prediction algorithms, and would apparently fit better our approach because they predict links based on binary node features, but the crucial difference is that in those models features are *latent* and need to be reconstructed, whereas in our scenario features (categories) are readily available.

Results. In the following, we are only going to discuss the best algorithms and combinations, besides some of the most interesting alternatives. The raw average bpref values are displayed in Table 8.9.

Some complementary information about the behavior is provided by the precision-recall graph of Figure 8.11: first of all, LlamaFur, AA, $M4$ and their combinations have larger precision than the remaining ones at almost all the recall levels; on the other hand LlamaFur+AA is the best

[10] The formula is applied to the symmetric version of the graph, in our case; note that this (like LlamaFur) is a measure of expectedness, whereas $M2$ and $M4$ are measures of unexpectedness.

[11] The (internally) Studentized residual is obtained by dividing the residual (i.e., the difference from the sample mean) by the sample standard deviation.

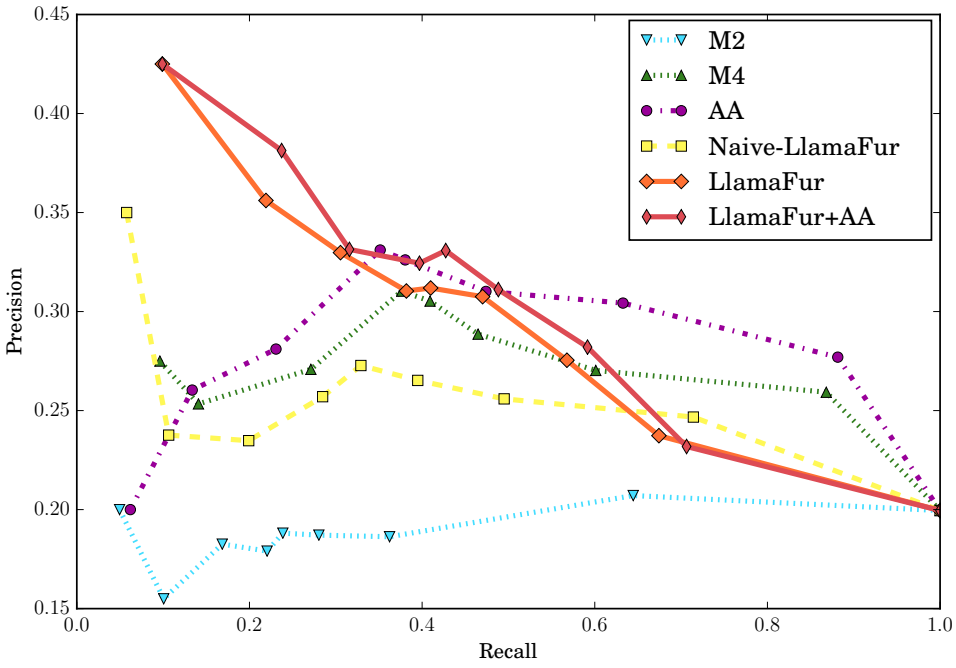


Figure 8.11: Average precision-recall values evaluated after the 1st, 2nd, 5th, 8th, 10th, 15th, 25th, 50th, and 100th percentiles for each query.

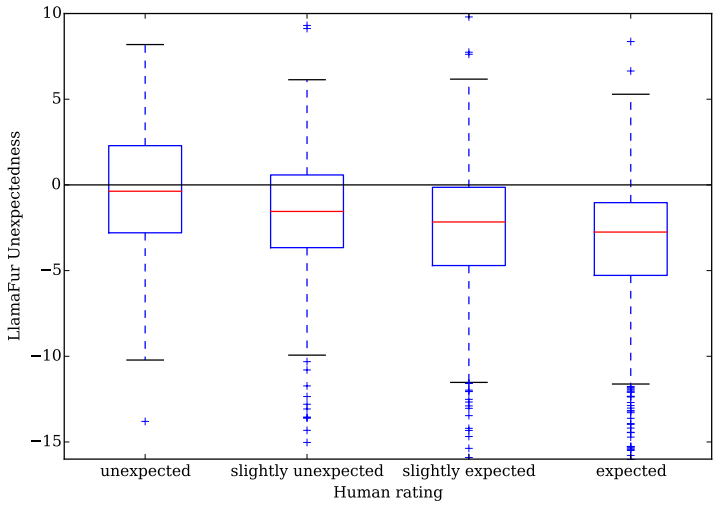


Figure 8.12: Comparison of the unexpectedness evaluated by LlamaFur (i.e., the opposite of (8.1)) over the different labels obtained from human evaluation.

method for recall values up to 50%, and LlamaFur has definitely better precision than AA until 30% of recall.

In fact, $M4$, AA and LlamaFur seem to be complementary to one another; in some sense, this is not surprising given that they stem from completely different sources of information: one is based on the textual content, another on the pure link graph and the latter on the category data.

Some further clue on the behavior of LlamaFur is provided by Figure 8.12, where the distribution of LlamaFur expectedness values is shown for each of the four labels provided by the human evaluation. The red line is the median and the upper/lower hinge represent the 75th/25th percentile.

Finally, let us remark that in order to enhance reproducibility and to foster further research on this problem, we are sharing all code and data needed to replicate our findings on `http://git.io/vmzjm`.

8.6 Discussion

In this chapter, we tested our methods on semantic networks; specifically, we chose the Wikipedia Link Graph as an archetypal example of open-access semantic network, widely employed in literature. In fact, as we have seen, categories in Wikipedia have been used as a type system when considering Wikipedia articles as concepts in a semantic network. Unfortunately, however, they are organized in a very noisy category graph, which is a pseudo-hierarchical graph that should describe which categories are a subset of the others.

We presented therefore a technique to prune and cleanse the Wikipedia category hierarchy. This method uses only the hierarchical information available within the category structure itself, making it particularly well-suited as a preprocessing phase to all the Wikipedia data mining tasks that benefit from a (clean) categorization of the Wikipedia articles. For example, our results suggest that building an ontology from Wikipedia categories would be more accurate and require less human fine-tuning if the cleansing procedure we propose is applied first.

The choice of the centrality measure to be used in the selection of milestones is crucial for our method; determining which centrality index gives the best performance is an arduous task. In this work we used a small excerpt of the Library of Congress Classification as ground truth, but we definitely think that this point needs more investigation, to be performed with a larger set of samples.

Thanks to this result, however, we were able to use the cleansed categories as features for our feature-rich network model. We employed therefore the 20 000 categories selected by our centrality measure as the features which describe the 4 514 662 articles in the English Wikipedia. With this set of categories, our algorithm runs in less than 10 minutes and it is able to reach a precision of 90% with a 84% recall, when measured on a balanced set of links and non-links, showing that our algorithm can process graphs with 10^8 links or more without effort.

We showed then that our model can be employed to find unexpected links in hyper linked document corpora, through the determination of the latent category matrix; the latter is built using the perceptron-like technique we previously presented. We demonstrated that our method

provides better accuracy than most existing text-based techniques, with higher efficiency and relying on a much smaller amount of information.

It would be easy to build, basing on our technique, a service able to show unexpected links for a given Wikipedia page. The matrix **W** should stay approximately the same over short periods of time, and therefore could be computed at periodic intervals. When the user requests a certain Wikipedia page, we could return the Wikipedia page highlighting the sentences that surround the unexpected links, since tables 8.7 and 8.8 show that those sentences often offer surprising information. For example, we might highlight the 10 links/sentences with the highest unexpectedness according to (8.1), and color them in different shades of red according to how unexpected they are. Such a service would be useful as a tool to explore unexpected facts contained in Wikipedia, as well as a tool for Wikipedia editors to check the correctness of those links. This tool, like our model, could be applied also to other scenarios: for example, in a social network, it could identify the friendship links that defy the social characteristics (political, ethnical, professional...) of the people involved, allowing the user to be more conscious about friends situated outside its filter bubble. We leave such experiments as future works.

Conclusions

In this thesis, we investigated large, feature-rich complex networks. With this name, we indicated the class of datasets representable as a graph whose nodes are also part of a bipartite graph (often known as an affiliation network). One can think of the whole data as being composed by a set of nodes, a set of links between them, and for each node, a collection of attributes, hereby called *features*.

Many different scenarios can be cast to this model. Among the one we directly applied, we can find semantic networks—where nodes are concepts, links are semantic relations, and features are (overlapping) categories of concepts; or many real-world social networks—e.g. in co-authorship network, features can be institutions or fields of research. However, many other examples can be found in bioinformatics, where biological functions of open reading frames of RNA and DNA can be viewed as features in the network of their interactions. Economic relations, or tagged web contents, could give us more examples.

Our ambition was to overcome the special role occupied by homophily in previous analysis of large feature-rich networks: the idea that the only way two nodes could form a link is by showing near-identical features in many cases might not be of much use. Instead, we wanted to analyze a model where nodes with *diverse* features can, as a result of them, form a link. This hypothesis is reasonable in many scenarios: in a semantic network, a movie will not be linked just to other movies (if any), but it will for sure form links with actors, directors, and so on. In an academic network, results from mathematics and statistics are often used and cited in data mining works; biology papers might in turn cite data mining, or physics, *et cetera*.

However, as we saw in chapter 2 (as well as in section 6.2), many of the previous works in literature were focusing on homophily; in machine learning, e.g., much more attention was devoted to the prediction of single labels on graphs than on predicting multiple, possibly interconnected features.

We tried, however, to recap the previous work done in dealing with networks where features of the nodes can interact between them, and be the reason a link is formed. Of the work dealing with such network models, we found that many of the proposed models and techniques that were able to explain possible connections between features of nodes were very complex and not able to scale to the size of many interesting real world and web-scale networks; for examples, current literature on multiplicative attribute graph models report examples ranging from dozen to a few thousands of nodes; methods based on graphical models, and Markov Chain Monte Carlo methods, treated even smaller network. This way of modeling networks resulted however in very accurate models, able to deal with features interactions beyond simple homophily, modeling also complex interactions between different features, or negative interactions as well.

For this reason, we chose to adopt the model developed by Miller, Griffiths and Jordan as our starting point. We described our framework in detail in chapter 3. Our wish in this work was to use this model as an instrument to build tools able to scale, moving the area of operations of this model to feature-rich networks whose size is in the range of millions—or tens of millions—of nodes.

First, we investigated how to simulate realistically feature-rich networks. To do this, we expanded in different ways the model, as we saw in chapter 4. We first provided a way to model the evolution of features, basing on the known strategy of Indian Buffet Process; in this way, nodes

evolve by copying features one from each other, propagating them in a rich-get-richer fashion. On top of this behavior, we added a way to model competition between nodes in spreading their features, by introducing a fitness parameter for each node, that summed up how able is that node in spreading its features. In this way, older nodes might not be the only successful nodes in this propaganda of features, but young-but-fit nodes could well "defeat" them. We also showed how this new parameters can be seen as a new way to rank nodes on a feature-rich network, by inferring how fit this node was in spreading its features. We have shown a Gibbs sampling algorithm attempting to do that, and we saw in which cases it can distinguish fit from unfit nodes.

After having defined this model, we provided proper statistical estimators able to carefully measure all the others parameters of our model. In this way, given a real complex network, we are able to reconstruct its parameters and construe a different network with similar characteristics (i.e., drawing a different network from the plausible distribution of the original).

Then, we showed how our model is able, when given the right set of parameters, to generate synthetic feature-rich networks with realistic features. Its degree distribution can easily reflect classical power laws distributions (but also different ones, like the log-normal distribution many networks seems to display, as we saw in section 7.3). Also its distance distribution, a more complex network property, presents striking similarities with real ones. In the end, our investigation validated this tool as a good way to experiment with feature-rich networks, with all the advantages of practicing on synthetic datasets, before trying our techniques on real data. That was our *modus operandi* in the second part of this work, where we presented data mining instruments and algorithms based on this model, and able to scale to large networks.

The first problem we dealt with, in chapter 5, was how to infer the latent feature-feature matrix. This matrix is the main unknown of the model; it determines how features interact between each other. For example, a positive value for its entry h, k indicates that it is likely that nodes with feature h form links towards nodes with feature k. To estimate this matrix means getting a representation of the deep, latent motives that form the network. Possible applications include dimensionality reduction of the features, measuring semantic distance, discovering hidden relationships, and so on. While many possible methods are available in literature for this problem, they all dealt with small to medium sized network, while we are interested in large-scale networks. Our first approach was guided by a Naïve Bayes scheme. We demonstrated that a very simple equation to estimate the matrix can be derived by assuming (naively) independence between features, and by making a few assumptions to restrict our model equation (specifically, we fix its activation function to be the exponential). However, it is apparent that the naïve assumptions could not hold in many real cases, and that could impact the results of this algorithm.

For this reason, we proposed a second algorithm. Incidentally, this algorithm can also be seen as a possible way to predict links in our model. In fact, by restricting our model to be deterministic (once again, fixing its activation function as the step function), the rule governing the model becomes identical to the prediction rule of a perceptron classifier. This fact means that by learning to predict links with a perceptron, fed with the outer product of the feature vectors of a pair of nodes, the internal state of the perceptron has to converge to the correct

feature-feature matrix of our model. We showed how to interpret classical bounds on the number of errors of the perceptron in this case; particularly, we saw which factors increase this bound, and we saw that if a perfect latent feature-feature matrix exists, the perceptron will find it. Then, we employed state-of-the-art perceptron-like classifiers, and in particular the Passive-Aggressive algorithm, to design a very fast and accurate algorithm to estimate the feature-feature matrix by means of predicting links. Finally, we employed our simulation tools to show how accurate this estimate can be, showing that when used on synthetic data generated by our model (with or without determinism), its prediction are usually beyond 90% in both precision and recall. The synthetic data we used was composed by 10 000 nodes, for which the algorithm required seconds to run; as we will recap later, we also showed experiments dealing with more than 10 millions nodes, for which it it required minutes.

In chapter 6, we dedicated ourselves to the dual problem with respect to link prediction: how to predict features basing on the links. In order to be scalable and to offer a novel point of view on multi-label prediction on graphs, we chose to design a neural network for this task. The neural network operates by considering one node at a time, receiving as input the features appearing in its in- and out-neighborhood, and returning a ranking of the features in a way that plausible features are high-ranked and implausible ones are low-ranked. We show that, also in this task, the latent feature-feature matrix appears as the hidden layer of the neural network. As in the previous case, we tested this algorithm on synthetic data, observing a Mean Average Precision of the output ranking between 0.49 and 0.68. We also measured the weighted average of the AUC ROC, obtaining values between 0.68 and 0.84, depending on the complexity of the latent matrix used for the synthetic data.

In the final part of this thesis, we applied the techniques we developed to real datasets and use cases. First, in chapter 7, we experimented those techniques on citation networks.

To test our simulation approach, we took a citation network, consisting of 27 770 papers, and used as features the non-common words that appear in each paper title and abstract. Our goal on this dataset was to use our estimators and, in turn, our simulation process in order to obtain a new feature-rich complex network that looks overall similar in its global properties to the original one, despite being different in its actual values. This means identifying the distribution this network might come from, and drawing another network from the same distribution, thus validating our process as a good way to generate realistic data. Then, we used this newly generated synthetic matrix to generate a network, obtaining a graph with a distance and degree distribution close to the one we found in real complex network.

Then, we conducted a different kind of experiment on a different citation network, composed by 18 939 155 nodes and 189 465 540 links. We used the tools we developed for estimating the feature-feature matrix in order to: (1), validate their performance on real data, and (2), to show how they can be used to assess which feature set can be more useful in explaining the links of a network. In fact, we considered different feature sets: one composed by the institutions of the authors of each paper; the other one, by the fields of studies each paper has been associated to. First, we observed that, also on real data, our perceptron-like algorithm works definitely better than its Naïve Bayes counterpart—indicating that naïve independence assumptions are

detrimental. Second, we observed a great difference in the prediction abilities of the two feature sets; our approach can show that the ability to predict citation links is much higher for fields of study than for institutions. In particular, we obtained an area under Precision-Recall curve of our algorithm of 0.92 for fields of studies.

Finally, in chapter 8, we experimented with semantic networks; in particular, we considered the Wikipedia link graph and the category graph. The categories in Wikipedia are associated to each article; they have been used as a type system when considering Wikipedia articles as concepts in a semantic networks. The category graph is a very noisy, pseudo-hierarchical graph that should describe which categories are a subset of the others. First, we used network centrality measures to prune this graph, obtaining a clearer version of it, leading to a clear-cut association between articles and categories. By comparing different ranking with a standard, human-curated bibliographical classification system, we showed how harmonic centrality in this graph is able to capture the importance of each category, with an AUC ROC of 0.94.

Then, we were able to use the cleansed category system for our graph. We employed only 20 000 categories (the most important according to our centrality measure) as features, for the 4 514 662 articles in the English Wikipedia. With this set of categories, our algorithm runs in less than 10 minutes and it is able to reach a precision of 90% with a 84% recall.

As a use case of these results, we employed the reconstructed latent feature-feature matrix to find *unexpected relations* among the semantic relations described by Wikipedia. Catching serendipity in documental corpus is an area of growing interest in Information Retrieval, and we found out that our approach could be of help. In fact, by simply using the scores computed by our model (when fed with the feature-feature matrix found by our algorithm), one can obtain a score of (un-) expectedness for each link. For example, the most unexpected relation found by our method relating to "Kim Jong-il", former dictator of North Korea, is "Elvis Presley". We therefore compared our technique with previous approaches, based on different sources of information: text-based approaches, and classical link prediction. As a golden truth, we obtained human evaluations of how surprising each link in the test set was. To determine the test set and how to conduct the evaluation, we followed the same specifications of the TREC challenge, a classical Information Retrieval confrontation, and thus we used their same evaluation measure, bpref. Our method obtained the highest bpref among all considered methods.

To sum up, in this thesis we have showed how considering features in complex networks might be highly valuable for their understanding. The scenarios in which (overlapping) features are available for nodes seem almost endless. Our ambition was to provide tools able to scale up to the size of the largest of these networks. Many believe complexity and networks are keys to reach a greater comprehension in many areas of vital importance – from cancer research [50], to tracking the spread of epidemics [36], or even to predict the stability of ecosystems under climate change [62, 82] or to reach more trustworthy economical structures [20]. We hope that the theoretical framework of large scale feature-rich networks, together with the tools we designed and analyzed, could provide novel, attractive insights in research on these much needed matters.

Thanks

Here I wish to thank all the people without whom this work would have been impossible: first, Paolo Boldi, my advisor and mentor, for all the efforts he took to support my research, since the very beginning of my experience as a PhD student; his clearness of vision always helped in guiding the path, like a compass; his brightness of spirit also helped in the roller coaster of doctorate. Then, let me thank Sebastiano Vigna, my co-advisor, for his intuitions and his brainwaves, but also for his remarks about science and life. I am proud to have been a member of their Laboratory of Web Algorithmics in Milan.

I want to thank my reviewers Ricardo Baeza-Yates, Francesco Bonchi, Guido Caldarelli and Carlos Castillo, for all the time and energy they spent to improve my work.

Let me thank all the other contributors to this work: Irene Crimaldi, from the IMT School for Advanced Studies in Lucca, who provided ideas and precious help; Massimo Santini, again from the Laboratory of Web Algorithmics, for all of his scientific, technological, academic and human support over the years; all the bright researchers I worked with from the Center for Complexity & Biosystems: Stefano Zapperi, Caterina La Porta, and Francesc Font Clos; and Dirk Brockmann, together with all of his students from the Research on Complex Systems laboratory in Berlin, for their warm hospitality.

I wish to thank Claudio Ceruti, Marco Genuzio, Andrea Marino, and all the other fellow colleagues and friends in via Comelico, for their friendship and their helpful and thought-provoking everyday discussions.

Finally, let me thank my family, my friends, and Giulia. The role you all had in this is so profound it cannot be put into words.

Bibliography

[1] L.A. Adamic and E. Adar, *Friends and neighbors on the web*, Social Networks **25** (2001), 211–230.

[2] C.C. Aggarwal, *Outlier analysis*, Springer Science & Business Media, 2013.

[3] M. Agyemang, K. Barker, and R. Alhajj, *Hybrid approach to web content outlier mining without query vector*, Dawak, 2005, pp. 285–294.

[4] William Aiello and Fan Chung, *Random evolution in massive graphs*, Proc. of the 42nd ieee symposium on foundations of computer science, 2001, pp. 510–.

[5] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing, *Mixed membership stochastic blockmodels*, J. Mach. Learn. Res. **9** (June 2008), 1981–2014.

[6] Omar Ali, Giovanni Zappella, Tijl De Bie, and Nello Cristianini, *An empirical comparison of label prediction algorithms on automatically inferred networks*, Icpram (2), 2012, pp. 259–268.

[7] Frank Rosenblatt and, *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*, DTIC Document, 1961.

[8] Paolo Boldi and Corrado Monti, *Llamafur: learning latent category matrix to find unexpected relations in wikipedia*, Proceedings of the 8th acm conference on web science, 2016, pp. 218–222.

[9] Myunghwan Kim and Jure Leskovec, *Nonparametric multi-group membership model for dynamic networks*, Advances in neural information processing systems, 2013, pp. 1385–1393.

[10] Thomas L. Griffiths and Zoubin Ghahramani, *The indian buffet process: An introduction and review*, J. Mach. Learn. Res. **12** (2011), 1185–1224.

[11] Jac M. Anthonisse, *The rush in a graph*, Amsterdam: University of Amsterdam Mathematical Centre, 1971.

[12] S. O Aral, J.P. Hughes, B. Stoner, W. Whittington, H.H. Handsfield, R.M. Anderson, and K.K. Holmes, *Sexual mixing patterns in the spread of gonococcal and chlamydial infections*, American Journal of Public Health **89** (1999), no. 6, 825–833.

[13] Hagai Attias, *A variational bayesian framework for graphical models*, Advances in neural information processing systems **12** (2000), no. 1-2, 209–215.

[14] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, *Dbpedia: A nucleus for a web of open data*, Springer, 2007.

[15] Andrea Ballatore, Michela Bertolotto, and David C Wilson, *Geographic knowledge extraction and semantic similarity in openstreetmap*, Knowledge and Information Systems **37** (2013), no. 1, 61–81.

[16] Albert-Laszlo Barabasi, *The origin of bursts and heavy tails in human dynamics*, Nature **435** (2005), no. 7039, 207–211.

[17] Albert-László Barabási and Réka Albert, *Emergence of scaling in random networks*, Science **286** (1999October), 509–512.

[18] Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya, *Hierarchical multi-label prediction of gene function*, Bioinformatics **22** (2006), no. 7, 830–836.

[19] Federico Bassetti, Irene Crimaldi, and Fabrizio Leisen, *Conditionally identically distributed species sampling sequences*, Advances in Applied Probability **42** (201006), no. 2, 433–459.

[20] Stefano Battiston, Michelangelo Puliga, Rahul Kaushik, Paolo Tasca, and Guido Caldarelli, *Debtrank: Too central to fail? financial networks, the fed and systemic risk*, Scientific reports **2** (2012).

[21] A. Bavelas, *A mathematical model for group structures*, Human Organization **7** (1948), 16–30.

[22] Tim Berners-Lee, James Hendler, and Ora Lassila, *The semantic web*, Scientific american **284** (2001), no. 5, 28–37.

[23] Patrizia Berti, Irene Crimaldi, Luca Pratelli, and Pietro Rigo, *Central limit theorems for an indian buffet model with random weights*, The Annals of Applied Probability (2014). (forthcoming). Currently available on http://imstat.org/aap/future_papers.html and on arXiv (1304.3626, 2013).

[24] Ginestra Bianconi and Albert-László Barabási, *Competition and multiscaling in evolving networks*, EPL (Europhysics Letters) **54** (2001), no. 4, 436.

[25] Halil Bisgin, Nitin Agarwal, and Xiaowei Xu, *A study of homophily on social media*, World Wide Web **15** (2012), no. 2, 213–232.

[26] C.M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, Springer-Verlag New York, Inc., 2006.

[27] David M Blei and Michael I Jordan, *Modeling annotated data*, Proceedings of the 26th annual international acm sigir conference on research and development in informaion retrieval, 2003, pp. 127–134.

[28] D.M. Blei, A.Y. Ng, and M.I. Jordan, *Latent dirichlet allocation*, JMLR **3** (2003), 993–1022.

[29] Duane C Boes, FA Graybill, and AM Mood, *Introduction to the theory of statistics*, Series in probability (1974).

[30] Paolo Boldi, Irene Crimaldi, and Corrado Monti, *A network model characterized by a latent attribute structure with competition*, Information Sciences (2016), –.

[31] Paolo Boldi and Corrado Monti, *Cleansing wikipedia categories using centrality*, Proc. 24th international conference on www, 2016.

[32] Paolo Boldi and Sebastiano Vigna ., *Axioms for centrality*, Internet Mathematics **10** (2014), no. 3-4, 222–262.

[33] Paolo Boldi and Sebastiano Vigna, *In-core computation of geometric centralities with HyperBall: A hundred billion nodes and beyond*, Proc. of 2013 ieee 13th international conference on data mining workshops (icdmw 2013), 2013.

[34] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, *Freebase: a collaboratively created graph database for structuring human knowledge*, Proceedings of the 2008 acm sigmod international conference on management of data, 2008, pp. 1247–1250.

[35] Ronald L. Breiger, *The Duality of Persons and Groups*, Social Forces **53** (1974), no. 2, 181–190.

[36] Dirk Brockmann, *Understanding and predicting the global spread of emergent infectious diseases*, Public health forum, 2014, pp. 4–e1.

[37] Tamara Broderick, Michael I. Jordan, and Jim Pitman, *Beta processes, stick-breaking and power laws*, Bayesian Analysis **7** (201206), no. 2, 439–476.

[38] Chris Buckley and Ellen M Voorhees, *Retrieval evaluation with incomplete information*, Proc. of the 27th acm sigir conference on research and development in information retrieval, 2004, pp. 25–32.

[39] Guido Caldarelli, Andrea Capocci, Paolo De Los Rios, and Miguel A Munoz, *Scale-free networks from varying vertex intrinsic fitness*, Physical review letters **89** (2002), no. 25, 258702.

[40] Andrea Capocci, Francesco Rao, and Guido Caldarelli, *Taxonomy and clustering in collaborative systems: the case of the on-line encyclopedia wikipedia*, EPL (Europhysics Letters) **81** (2007), no. 2, 28006.

[41] Andrea Capocci, Vito DP Servedio, Francesca Colaiori, Luciana S Buriol, Debora Donato, Stefano Leonardi, and Guido Caldarelli, *Preferential attachment in the growth of social networks: The internet encyclopedia wikipedia*, Physical Review E **74** (2006), no. 3, 036116.

[42] V.R. Carvalho and W.W. Cohen, *Single-pass online learning: Performance, voting schemes and online feature selection*, Proc. of the 12th acm sigkdd, 2006, pp. 548–553.

[43] George Casella and Edward I George, *Explaining the gibbs sampler*, The American Statistician **46** (1992), no. 3, 167–174.

[44] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile, *On the generalization ability of on-line learning algorithms*, IEEE Transactions on Information Theory **50** (2004), no. 9, 2050–2057.

[45] Nicolo Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella, *Random spanning trees and the prediction of weighted graphs*, Journal of Machine Learning Research **14** (2013), no. 5.

[46] Nicolo Cesa-Bianchi and Gábor Lugosi, *Prediction, learning, and games*, Cambridge university press, 2006.

[47] Venkat Chandrasekaran, Benjamin Recht, Pablo A Parrilo, and Alan S Willsky, *The convex geometry of linear inverse problems*, Foundations of Computational mathematics **12** (2012), no. 6, 805–849.

[48] J. Chang and D.M. Blei, *Relational topic models for document networks*, International conference on artificial intelligence and statistics, 2009, pp. 81–88.

[49] Michel Chein and Marie-Laure Mugnier, *Graph-based knowledge representation: computational foundations of conceptual graphs*, Springer Science & Business Media, 2008.

[50] Silvia Anna Ciafrè and Silvia Galardi, *micrornas and rna-binding proteins: a complex network of interactions and reciprocal regulations in cancer*, RNA biology **10** (2013), no. 6, 934–942.

[51] Giulio Cimini, Tiziano Squartini, Diego Garlaschelli, and Andrea Gabrielli, *Systemic risk analysis on reconstructed economic and financial networks*, Scientific reports **5** (2015).

[52] Dan Ciregan, Ueli Meier, and Jurgen Schmidhuber, *Multi-column deep neural networks for image classification*, Computer vision and pattern recognition (cvpr), 2012 ieee conference on, 2012, pp. 3642–3649.

[53] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman, *Power-law distributions in empirical data*, SIAM Rev. **51** (November 2009), no. 4, 661–703.

[54] D.G. Coderre, *Fraud detection: Using data analysis techniques to detect fraud*, Global Audit Publications, 1999.

[55] Eliot A Cohen and Albert-László Barabási, *Linked: The new science of networks*, JSTOR, 2002.

[56] R. Dennis Cook and Sanford Weisberg, *Residuals and Influence in Regression*, Monographs on Statistics and Applied Probability, 18, Chapman and Hall/CRC, 1983.

[57] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, *Online passive-aggressive algorithms*, J. Mach. Learn. Res. **7** (2006), 551–585.

[58] Nello Cristianini and John Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press, 2000.

[59] Derek J de Solla Price, *Networks of scientific papers*, Science **149** (1965), no. 3683, 510–515.

[60] Frank Dellaert, *The expectation maximization algorithm* (2002).

[61] Krzysztof Dembczynski, Wojciech Kotlowski, and Eyke Hüllermeier, *Consistent multilabel ranking through univariate losses*, Proceedings of the 29th international conference on machine learning (icml-12), 2012, pp. 1319–1326.

[62] Jonathan F Donges, Yong Zou, Norbert Marwan, and Jürgen Kurths, *Complex networks in climate dynamics*, The European Physical Journal Special Topics **174** (2009), no. 1, 157–179.

[63] Janardhan Rao Doppa, Jun Yu, Prasad Tadepalli, and Lise Getoor, *Learning algorithms for link prediction based on chance constraints*, Joint european conference on machine learning and knowledge discovery in databases, 2010, pp. 344–360.

[64] John Duchi, Elad Hazan, and Yoram Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, Journal of Machine Learning Research **12** (2011), no. Jul, 2121–2159.

[65] Leo Egghe and Ronald Rousseau, *Introduction to informetrics: Quantitative methods in library, documentation and information science* (1990).

[66] D. Eklou, Y. Asano, and M. Yoshikawa, *How the web can help wikipedia: A study on information complementation of wikipedia by the web*, Proc. of the 6th international conference on ubiquitous information management and communication, 2012, pp. 9:1–9:10.

[67] P. Erdős and A. Rényi, *On the evolution of random graphs*, Publ. Math. Inst. Hung. Acad. Sci **5** (1960), 17–61.

[68] B Everett, *An introduction to latent variable models*, Springer Science & Business Media, 2013.

[69] MS Fabian, K Gjergji, and W Gerhard, *Yago: A core of semantic knowledge unifying wordnet and wikipedia*, 16th international world wide web conference, www, 2007, pp. 697–706.

[70] S. Faralli, G. Stilo, and P. Velardi, *What women like: A gendered analysis of twitter users' interests based on a twixonomy*, Ninth international aaai conference on web and social media, 2015.

[71] Angela Fogarolli, *Word sense disambiguation based on wikipedia link structure*, Semantic computing, 2009. icsc'09. ieee international conference on, 2009, pp. 77–82.

[72] Eugene Garfield et al., *Science citation index-a new dimension in indexing*, Science **144** (1964), no. 3619, 649–654.

[73] ———, *Citation analysis as a tool in journal evaluation*, 1972.

[74] Diego Garlaschelli, Stefano Battiston, Maurizio Castri, Vito DP Servedio, and Guido Caldarelli, *The scale-free topology of market investments*, Physica A: Statistical Mechanics and its Applications **350** (2005), no. 2, 491–499.

[75] Diego Garlaschelli and Maria I Loffredo, *Fitness-dependent topological properties of the world trade web*, Physical review letters **93** (2004), no. 18, 188701.

[76] Stuart Geman and Donald Geman, *Stochastic relaxation, gibbs distributions, and the bayesian restoration of images*, IEEE Trans. Pattern Anal. Mach. Intell. **6** (1984), no. 6, 721–741.

[77] C. Gentile, *A new approximate maximal margin classification algorithm*, J. Mach. Learn. Res. **2** (2002), 213–242.

[78] Lise Getoor, *Introduction to statistical relational learning*, MIT press, 2007.

[79] Charles J. Geyer and Minnesota Univ. (Minneapolis School Of Statistics), *Markov Chain Monte Carlo Maximum Likelihood*, Defense Technical Information Center, 1992.

[80] Zoubin Ghahramani, *Bayesian nonparametrics and the probabilistic approach to modelling*, Philosophical Transactions of the Royal Society A **371** (2012), no. 20110553.

[81] Edgar N Gilbert, *Random graphs*, The Annals of Mathematical Statistics **30** (1959), no. 4, 1141–1144.

[82] Sarah E Gilman, Mark C Urban, Joshua Tewksbury, George W Gilchrist, and Robert D Holt, *A framework for community interactions under climate change*, Trends in Ecology & Evolution **25** (2010), no. 6, 325–331.

[83] Neil Zhenqiang Gong, Wenchang Xu, Ling Huang, Prateek Mittal, Emil Stefanov, Vyas Sekar, and Dawn Song, *Evolution of social-attribute networks: measurements, modeling, and implications using google+*, Proceedings of the 2012 acm conference on internet measurement conference, 2012, pp. 131–144.

[84] Leo A Goodman, *Exploratory latent structure analysis using both identifiable and unidentifiable models*, Biometrika **61** (1974), no. 2, 215–231.

[85] Andrew Gregorowicz and Mark A Kramer, *Mining a large-scale term-concept network from wikipedia*, MITRE Corporation **202** (2006).

[86] Thomas L. Griffiths and Zoubin Ghahramani, *Infinite latent feature models and the indian buffet process*, Advances in neural information processing systems, 2005, pp. 475–482.

[87] Ben Hachey, Will Radford, and James R Curran, *Graph-based named entity linking with wikipedia*, International conference on web information systems engineering, 2011, pp. 213–226.

[88] Peter Hall and C. C. Heyde, *Martingale limit theory and its application*, Academic Press, New York, NY, 1980.

[89] Xianpei Han and Jun Zhao, *Named entity disambiguation by leveraging wikipedia semantic knowledge*, Proceedings of the 18th acm conference on information and knowledge management, 2009, pp. 215–224.

[90] David Heckerman, Dan Geiger, and David M Chickering, *Learning bayesian networks: The combination of knowledge and statistical data*, Machine learning **20** (1995), no. 3, 197–243.

[91] K. Henderson and T. Eliassi-Rad, *Applying latent dirichlet allocation to group discovery in large graphs*, Proc. 2009 acm symposium on applied computing, 2009, pp. 1456–1461.

[92] Neil W Henry, *Latent structure analysis*, Encyclopedia of statistical sciences (1983).

[93] N. Hens, N. Goeyvaerts, M. Aerts, Z. Shkedy, P. Van Damme, and P. Beutels, *Mining social mixing patterns for infectious disease models based on a two-day population survey in belgium*, BMC Infectious Diseases **9** (2009), no. 1, 5.

[94] Mark Herbster and Massimiliano Pontil, *Prediction on a graph with a perceptron*, Advances in neural information processing systems, 2006, pp. 577–584.

[95] John Hertz, Anders Krogh, and Richard G Palmer, *Introduction to the theory of neural computation*, Vol. 1, Basic Books, 1991.

[96] P.D. Hoff, *Multiplicative latent factor models for description and prediction of social networks*, Computational and Mathematical Organization Theory **15** (2009), no. 4, 261–272.

[97] Jake M Hofman and Chris H Wiggins, *Bayesian approach to network modularity*, Physical review letters **100** (2008), no. 25, 258701.

[98] Lorenzo Isella, Mariateresa Romano, Alain Barrat, Ciro Cattuto, Vittoria Colizza, Wouter Van den Broeck, Francesco Gesualdo, Elisabetta Pandolfi, Lucilla Ravà, Caterina Rizzo, and Alberto Eugenio Tozzi, *Close encounters in a pediatric ward: measuring face-to-face proximity and mixing patterns with wearable sensors*, PloS one **6** (2011), no. 2, e17144.

[99] Matthew O. Jackson, *Social and economic networks*, Princeton University Press, Princeton, NJ, USA, 2008.

[100] Matthew O. Jackson and Brian W. Rogers, *Meeting strangers and friends of friends: How random are social networks?*, American Economic Review **97** (2007), no. 3, 890–915.

[101] F. Jacquenet and C. Largeron, *Discovering unexpected documents in corpora*, Knowledge-Based Systems **22** (2009), no. 6, 421 –429.

[102] P. Jain, P.Z. Yeh, K. Verma, R.G. Vasquez, M. Damova, P. Hitzler, and A.P. Sheth, *Contextual ontology alignment of lod with an upper ontology: A case study with proton*, The semantic web: Research and applications, 2011, pp. 80–92.

[103] Lancelot F James, *Poisson latent feature calculus for generalized indian buffet processes*, arXiv preprint arXiv:1411.2936 (2014).

[104] N. Japkowicz and S. Stephen, *The class imbalance problem: A systematic study*, Intelligent data analysis **6** (2002), no. 5, 429–449.

[105] Yookyung Jo, John E Hopcroft, and Carl Lagoze, *The web of topics: discovering the topology of topic evolution in a corpus*, Proceedings of the 20th international conference on world wide web, 2011, pp. 257–266.

[106] Yookyung Jo, Carl Lagoze, and C Lee Giles, *Detecting research topics via the correlation between graphs and texts*, Proceedings of the 13th acm sigkdd international conference on knowledge discovery and data mining, 2007, pp. 370–379.

[107] P. Kapanipathi, P. Jain, C. Venkataramani, and A. Sheth, *User interests identification on twitter using a hierarchical knowledge base*, The semantic web: Trends and challenges, 2014, pp. 99–113.

[108] Dimitris Karlis and Ioannis Ntzoufras, *Analysis of sports data by using bivariate poisson models*, Journal of the Royal Statistical Society: Series D (The Statistician) **52** (2003), no. 3, 381–393.

[109] SK Katti and A Vijaya Rao, *Handbook of the poisson distribution*, Technometrics **10** (1968), no. 2, 412–412.

[110] Leo Katz, *A new status index derived from sociometric analysis*, Psychometrika **18** (1953), no. 1, 39–43.

[111] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda, *Learning systems of concepts with an infinite relational model*, Aaai, 2006, pp. 5.

[112] Myunghwan Kim and Jure Leskovec ., *Multiplicative attribute graph model of real-world networks*, Internet Mathematics **8** (2012), no. 1-2, 113–160.

[113] Myunghwan Kim and Jure Leskovec, *Modeling social networks with node attributes using the multiplicative attribute graph model*, arXiv preprint arXiv:1106.5053 (2011).

[114] John F. Charles Kingman, *Completely random measures*, Pacific Journal of Mathematics **21** (1967), no. 1, 59–78.

[115] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi, *Optimization by simmulated annealing*, science **220** (1983), no. 4598, 671–680.

[116] Istvan Z Kiss, Mark Broom, Paul G Craze, and Ismael Rafols, *Can epidemic models describe the diffusion of topics across disciplines?*, Journal of Informetrics **4** (2010), no. 1, 74–82.

[117] A. Kittur, E.H. Chi, and B. Suh, *What's in wikipedia?: mapping topics and conflict using socially annotated category structure*, Proc. sigchi conference on human factors in computing systems, 2009, pp. 1509–1512.

[118] Jon M Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of the ACM (JACM) **46** (1999), no. 5, 604–632.

[119] Ravi Kumar, Jasmine Novak, and Andrew Tomkins, *Structure and evolution of online social networks*, Proc. of the 12th acm sigkdd international conference on knowledge discovery and data mining, 2006, pp. 611–617.

[120] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D Sivakumar, Andrew Tomkins, and Eli Upfal, *Stochastic models for the web graph*, Foundations of computer science, 2000. proceedings. 41st annual symposium on, 2000, pp. 57–65.

[121] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio, *Basic notions for the analysis of large two-mode networks*, Social Networks **30** (2008), no. 1, 31 –48.

[122] Silvio Lattanzi and D. Sivakumar, *Affiliation networks*, Proc. of the forty-first annual acm symposium on theory of computing, 2009, pp. 427–434.

[123] Paul F Lazarsfeld, *Latent structure analysis*, Psychology: A study of a science **3** (1959), 476–543.

[124] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani, *Kronecker graphs: An approach to modeling networks*, Journal of Machine Learning Research **11** (2010), no. Feb, 985–1042.

[125] Nan Lin, *Foundations of social research*, McGraw-Hill, New York, 1976.

[126] B. Liu, Y. Ma, and Philip S. Yu, *Discovering unexpected information from your competitors' web sites*, Kdd, 2001, pp. 144–153.

[127] Y. Liu, A. Niculescu-Mizil, and W. Gryc, *Topic-link lda: joint models of topic and author community*, Proc. 26th annual international conference on machine learning, 2009, pp. 665–672.

[128] L. Lü and T. Zhou, *Link prediction in complex networks: A survey*, Physica A: Statistical Mechanics and its Applications **390** (2011), no. 6, 1150 –1170.

[129] C.D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*, Cambridge University Press, New York, NY, USA, 2008.

[130] Alexa T McCray, *The umls semantic network*, Proc 13th annu symp comput appl med care, 1989, pp. 503–7.

[131] Miller McPherson, Lynn Smith-Lovin, and James M Cook, *Birds of a feather: Homophily in social networks*, Annual Review of Sociology **27** (2001), no. 1, 415–444.

[132] Edward Meeds, Zoubin Ghahramani, Radford M Neal, and Sam T Roweis, *Modeling dyadic data with binary latent factors*, Advances in neural information processing systems, 2006, pp. 977–984.

[133] Jörg Menche, Amitabh Sharma, Maksim Kitsak, Susan Dina Ghiassian, Marc Vidal, Joseph Loscalzo, and Albert-László Barabási, *Uncovering disease-disease relationships through the incomplete interactome*, Science **347** (2015), no. 6224, 1257601.

[134] Aditya Krishna Menon and Charles Elkan, *Link prediction via matrix factorization*, Joint european conference on machine learning and knowledge discovery in databases, 2011, pp. 437–452.

[135] T Mikolov and J Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems (2013).

[136] K.T. Miller, T.L. Griffiths, and M.I. Jordan, *Nonparametric latent feature models for link prediction*, In nips, 2009, pp. 1276–1284.

[137] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, *Playing atari with deep reinforcement learning*, arXiv preprint arXiv:1312.5602 (2013).

[138] Corrado Monti, A. Rozza, G. Zappella, M. Zignani, A. Arvidsson, and E. Colleoni, *Modelling political disaffection from twitter data*, Proc. of the 2nd int. wisdom, 2013, pp. 3.

[139] Morten Mørup, Mikkel N. Schmidt, and Lars Kai Hansen, *Infinite multiple membership relational modeling for complex networks*, CoRR **abs/1101.5097** (2011).

[140] Joel Mossong, Niel Hens, Mark Jit, Philippe Beutels, Kari Auranen, Rafael Mikolajczyk, Marco Massari, Stefania Salmaso, Gianpaolo Scalia-Tomba, Jacco Wallinga, Janneke Heijne, Malgorzata Sadkowska-Todys, Magdalena Rosinska, and W. John Edmunds, *Social contacts and mixing patterns relevant to the spread of infectious diseases*, PLoS Med **5** (2008), no. 3, e74.

[141] T. Murakami, K. Mori, and R. Orihara, *Metrics for evaluating the serendipity of recommendation lists*, Proc. of the 2007 conf. on new frontiers in artificial intelligence, 2008, pp. 40–46.

[142] A. Nadamoto, E. Aramaki, T. Abekawa, and Y. Murakami, *Content hole search in community-type content*, Proc. of the 18th international conference on www, 2009, pp. 1223–1224.

[143] Kotaro Nakayama, Takahiro Hara, and Shojiro Nishio, *Wikipedia link structure and text mining for semantic relation extraction*, Semsearch, 2008, pp. 59–73.

[144] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz, *Large-scale multi-label text classification—revisiting neural networks*, Joint european conference on machine learning and knowledge discovery in databases, 2014, pp. 437–452.

[145] Christie Nelson and William M Pottenger, *Nuclear detection using higher order learning*, Technologies for homeland security (hst), 2011 ieee international conference on, 2011, pp. 319–324.

[146] Mark Newman, *Networks: an introduction*, Oxford university press, 2010.

[147] Krzysztof Nowicki and Tom A B Snijders, *Estimation and prediction for stochastic blockstructures*, Journal of the American Statistical Association **96** (2001), no. 455, 1077–1087.

[148] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd, *The PageRank citation ranking: Bringing order to the web*, Technical Report SIDL-WP-1999-0120, Stanford Digital Library Technologies Project, Stanford University, 1998.

[149] Konstantina Palla, David A. Knowles, and Zoubin Ghahramani, *An Infinite Latent Attribute Model for Network Data*, Proc. of the 29th international conference on machine learning, 2012, pp. 1–8.

[150] Romualdo Pastor-Satorras and Alessandro Vespignani, *Evolution and structure of the internet: A statistical physics approach*, Cambridge University Press, 2007.

[151] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher, *From freebase to wikidata: The great migration*, Proceedings of the 25th international conference on world wide web, 2016, pp. 1419–1428.

[152] Joseph J Pfeiffer III, Sebastian Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher, *Attributed graph models: Modeling network structure with correlated attributes*, Proceedings of the 23rd international conference on world wide web, 2014, pp. 831–842.

[153] Jim Pitman, *Combinatorial stochastic processes*, Lecture Notes in Mathematics, vol. 1875, Springer-Verlag, Berlin, 2006.

[154] C. Plackner and V. Primoli, *Data forensics: A compare-and-contrast analysis of multiple methods*, Proc. of the conference on statistical detection of potential test fraud, 2012.

[155] N. Ramakrishnan and A.Y. Grama, *Data mining-guest editors' introduction: From serendipity to science*, Computer **32** (1999), no. 8, 34–37.

[156] Georg Rasch, *The poisson process as a model for a diversity of behavioral phenomena*, International congress of psychology, 1963, pp. 2.

[157] Richard H Richens, *Preprogramming for mechanical translation*, Mechanical Translation **3** (1956), no. 1, 20–25.

[158] Frank Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain*, Psychological review **65** (1958), no. 6, 386.

[159] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, *Learning internal representations by error propagation*, DTIC Document, 1985.

[160] Stuart Russell and Peter Norvig, *Artificial intelligence: A modern approach. 2010*, Prentice Hall, 2010.

[161] A.A. Salah, C. Gao, K. Suchecki, and A. Scharnhorst, *Need to categorize: A comparative look at the categories of universal decimal classification system and wikipedia*, Leonardo **45** (2012), no. 1, 84–85.

[162] Purnamrita Sarkar, Deepayan Chakrabarti, and Michael I. Jordan, *Nonparametric link prediction in dynamic networks*, Proc. of the 29th international conference on machine learning, 2012, pp. 1–8.

[163] Leander Schietgat, Celine Vens, Jan Struyf, Hendrik Blockeel, Dragi Kocev, and Sašo Džeroski, *Predicting gene function using hierarchical multi-label decision tree ensembles*, BMC bioinformatics **11** (2010), no. 1, 1.

[164] Jürgen Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks **61** (2015), 85–117.

[165] P. Schonhofen, *Identifying document topics using the wikipedia category network*, Web intelligence, 2006. wi 2006. ieee/wic/acm international conference on, 2006, pp. 456–462.

[166] Stuart C Shapiro, *The sneps semantic network processing system*, State University of New York at Buffalo, Department of Computer Science, 1978.

[167] Stuart C Shapiro and William J Rapaport, *Sneps considered as a fully intensional propositional semantic network*, The knowledge frontier, 1987, pp. 262–315.

[168] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis, *Mastering the game of go with deep neural networks and tree search*, Nature **529** (2016), no. 7587, 484–489.

[169] Amit Singhal, *Introducing the knowledge graph: things, not strings*, Official google blog (2012).

[170] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang, *An overview of microsoft academic service (mas) and applications*, Proceedings of the 24th international conference on world wide web, 2015, pp. 243–246.

[171] Tom A.B. Snijders and Krzysztof Nowicki, *Estimation and prediction for stochastic blockmodels for graphs with latent block structure*, Journal of Classification **14** (1997), no. 1, 75–100.

[172] John F Sowa, *Semantic networks*, Encyclopedia of Cognitive Science (2006).

[173] Paolo Boldi, Marco Rosa, and Sebastiano Vigna, *Hyperanf: approximating the neighbourhood function of very large graphs on a budget* (Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M.P. Ravindra, Elisa Bertino, and Ravi Kumar, eds.), ACM, 2011.

[174] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15** (2014), no. 1, 1929–1958.

[175] Samuel A Stouffer, Louis Guttman, Edward A Suchman, Paul F Lazarsfeld, Shirley A Star, and John A Clausen, *Measurement and prediction* (1950).

[176] M. Strube and S.P. Ponzetto, *Wikirelate! computing semantic relatedness using wikipedia*, Aaai, 2006, pp. 1419–1424.

[177] F.M. Suchanek, G. Kasneci, and G. Weikum, *Yago: A large ontology from wikipedia and wordnet*, Web Semantics: Science, Services and Agents on the World Wide Web **6** (2008), no. 3, 203 –217.

[178] Johan AK Suykens and Joos Vandewalle, *Least squares support vector machine classifiers*, Neural processing letters **9** (1999), no. 3, 293–300.

[179] Z.S. Syed, T. Finin, and A. Joshi, *Wikipedia as an ontology for describing documents*, Icwsm, 2008.

[180] Benjamin Taskar, Eran Segal, and Daphne Koller, *Probabilistic classification and clustering in relational data*, International joint conference on artificial intelligence, 2001, pp. 870–878.

[181] Yee W. Teh and Dilan Gorur, *Indian buffet processes with power-law behavior*, Advances in neural information processing systems 22, 2009, pp. 1838–1846.

[182] Romain Thibaux and Michael I. Jordan, *Hierarchical beta processes and the indian buffet process*, Proc. 11th conference on artificial intelligence and statistics (aistat), 2007, pp. 1–8.

[183] Lloyd N Trefethen and David Bau III, *Numerical linear algebra*, Vol. 50, Siam, 1997.

[184] K. Tsukuda, H. Ohshima, M. Yamamoto, H. Iwasaki, and K. Tanaka, *Discovering unexpected information on the basis of popularity/unpopularity analysis of coordinate objects and their relationships*, Proc. of the 28th annual acm symposium on applied computing, 2013, pp. 878–885.

[185] Bram van der Vos, Jon Atle Gulla, and Reind van de Riet, *Verification of conceptual models based on linguistic knowledge*, Data & knowledge engineering **21** (1997), no. 2, 147–163.

[186] Sebastiano Vigna, *A weighted correlation index for rankings with ties*, Proceedings of the 24th international conference on world wide web, 2015.

[187] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, *Extracting and composing robust features with denoising autoencoders*, Proceedings of the 25th international conference on machine learning, 2008, pp. 1096–1103.

[188] Jakob Voss, *Collaborative thesaurus tagging the wikipedia way*, arXiv preprint cs (2006).

[189] Denny Vrandečić and Markus Krötzsch, *Wikidata: a free collaborative knowledgebase*, Communications of the ACM **57** (2014), no. 10, 78–85.

[190] Claus Wilke, Stephan Altmeyer, and Thomas Martinetz, *Large-scale evolution and extinction in a hierarchically structured environment*, Proceedings of the 6th international conference on artificial life (alife-98), mit press, cambridge, ma, usa, 1998, pp. 266–274.

[191] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok, *Interpreting tf-idf term weights as making relevance decisions*, ACM Transactions on Information Systems (TOIS) **26** (2008), no. 3, 13.

[192] Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel, *Learning infinite hidden relational models*, Uncertainity in Artificial Intelligence (UAI2006) (2006).

[193] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla, *Evaluating link prediction methods*, Knowledge and Information Systems **45** (2015), no. 3, 751–782.

[194] T. Zesch and I. Gurevych, *Analysis of the wikipedia category graph for nlp applications*, Proc. textgraphs-2 workshop (naacl-hlt 2007), 2007, pp. 1–8.

[195] Min-Ling Zhang and Zhi-Hua Zhou, *A review on multi-label learning algorithms*, IEEE transactions on knowledge and data engineering **26** (2014), no. 8, 1819–1837.

[196] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty, *Semi-supervised learning using gaussian fields and harmonic functions*, Icml, 2003, pp. 912–919.