

Network Growth and the Spectral Evolution Model

Jérôme Kunegis
University of Koblenz-Landau
Koblenz, Germany
kunegis@uni-koblenz.de

Damien Fay
University of Cambridge
Cambridge, United Kingdom
damien_fay@yahoo.co.uk

Christian Bauckhage
Fraunhofer IAIS
St. Augustin, Germany
christian.bauckhage@iais.fraunhofer.de

ABSTRACT

We introduce and study the spectral evolution model, which characterizes the growth of large networks in terms of the eigenvalue decomposition of their adjacency matrices: In large networks, changes over time result in a change of a graph's spectrum, leaving the eigenvectors unchanged. We validate this hypothesis for several large social, collaboration, authorship, rating, citation, communication and tagging networks, covering unipartite, bipartite, signed and unsigned graphs. Following these observations, we introduce a link prediction algorithm based on the extrapolation of a network's spectral evolution. This new link prediction method generalizes several common graph kernels that can be expressed as spectral transformations. In contrast to these graph kernels, the spectral extrapolation algorithm does not make assumptions about specific growth patterns beyond the spectral evolution model. We thus show that it performs particularly well for networks with irregular, but spectral, growth patterns.

Categories and Subject Descriptors: H.4 Information Systems Applications: Miscellaneous

General Terms: Algorithms, Theory, Performance

Keywords: Graph kernels, link prediction, spectral graph theory

1. INTRODUCTION

Many learning problems on networks can be described as link prediction: finding movies a user might like, recommending friends, predicting communication events, or finding related topics in a collaboration graph. In each of these cases, a given network is assumed to *grow* over time and the task is to predict where new edges will appear and, in some cases, to also predict their weights.

Any given link prediction algorithm can be interpreted as a model of graph growth, by assuming that networks grow according to its predictions. In this paper, we are concerned with those link prediction algorithms that have an algebraic

description. We show that they imply a network growth model that we call the *spectral evolution model*. The spectral evolution model states that in terms of the eigenvalue decomposition of a network's adjacency matrix, growth can be described as a transformation of the eigenvalues, without significant change in the eigenvectors. Several common link prediction functions are of this form, such as rank reduction and the matrix exponential. These functions are graph kernels, and each of these algorithms represents a specific assumption about network growth, leading to a different spectral transformation function. The goal of this study is thus to give a method of choosing graph kernels, and furthermore to generalize them to a generic spectral link prediction algorithm that only assumes the spectral evolution model.

By observing the spectral evolution of large networks, we arrive at two results:

- First, we validate individual spectral link prediction functions by comparing them to actual spectral growth. This allows us to observe in a simple way whether a particular spectral link prediction function is suitable for a given network.
- Second, instead of using a specific spectral link prediction function which may rely on invalid assumptions, we extrapolate the spectral growth of the network. This has two advantages: If a network grows in accordance to a given link prediction function, we can observe this. If a network does not, then we obtain a link prediction algorithm that still works better than any simple spectral link prediction function.

In the examples we study, the majority of networks grow regularly when we look at single latent dimensions, but when we look at the whole spectrum they grow irregularly. This observation leads to the spectral extrapolation algorithm that we describe in this paper. As we will show, this algorithm has the advantage of being data-driven and free of any kernel-induced parameter, making it generalizable to networks of many different types.

We study the spectral evolution model in unipartite and bipartite, weighted, unweighted and signed network datasets from different application areas. We begin by introducing a motivational example and then define the spectral evolution model in Section 2. The model is tested on real networks in Section 3 and on common graph growth models in Section 4. As an application, we introduce the spectral extrapolation algorithm for link prediction in Section 5. Section 6 reviews related work and extensions to our approach, and Section 7 concludes this paper.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.
Copyright 2010 ACM 978-1-4503-0099-5/10/10 ...\$10.00.

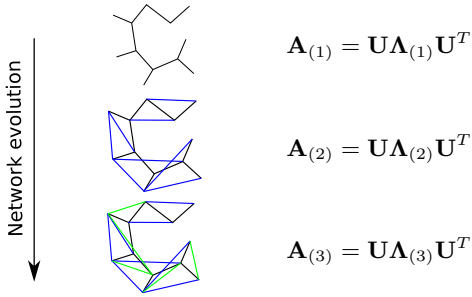


Figure 1: As edges appear in a network, the eigenvalue decomposition of the network’s adjacency matrix evolves spectrally according to the spectral evolution model.

2. MOTIVATION: FRIEND OF A FRIEND

We introduce the spectral evolution model here using the example of a social network and the common “friend of a friend” graph growth model. Given a network of n users connected by friendship links, let its adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ be defined as $A_{ij} = 1$ when users i and j are connected and $A_{ij} = 0$ otherwise. We then consider the eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. It can be shown that several common link prediction algorithms can be expressed as $\mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^T$, where F is a function that applies the same real function f to each diagonal element of $\mathbf{\Lambda}$ [16].

An example is given by the matrix exponential $\exp(\mathbf{A}) = \sum_{i=0}^{\infty} \mathbf{A}^i / i!$, which can be written as

$$\exp(\mathbf{A}) = \exp(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T) = \mathbf{U} \exp(\mathbf{\Lambda}) \mathbf{U}^T \quad (1)$$

where $\exp(\mathbf{\Lambda})$ can be computed by applying the exponential function to $\mathbf{\Lambda}$ ’s diagonal elements. The matrix exponential can be interpreted as a link prediction function as follows: Since the matrix power \mathbf{A}^i gives the number of paths of length i between any two nodes, the matrix exponential gives the sum over all paths between any two nodes, weighted by the inverse factorial of path length. This characterization of the matrix exponential makes it suitable as a realization of the “friend of a friend” model: All predicted links close triangles or longer cycles, and parallel paths between two nodes reinforce their connection while short paths make the connection stronger.

At the same time, Equation (1) shows that the matrix exponential is a spectral transformation of the adjacency matrix \mathbf{A} . As we will see later, several other link prediction functions can be expressed as spectral transformations too, and provide the basis of the spectral evolution model: that all network growth is of this form.

2.1 The Spectral Evolution Model

The spectral evolution model states that the evolution of a network can be described by a change in the network’s spectrum, while the network’s eigenvectors stay largely constant over time. If $\mathbf{A}_{(t)}$ is the adjacency matrix of a network at the time t , the spectral evolution model states that there is an orthogonal matrix \mathbf{U} and diagonal matrices $\mathbf{\Lambda}_{(t)}$ such that $\mathbf{A}_{(t)} = \mathbf{U}\mathbf{\Lambda}_{(t)}\mathbf{U}^T$ is a valid eigenvalue decomposition for all t . Figure 1 summarizes the proposed spectral evolution model. To examine the validity of this model, we follow two approaches:

First (in Section 3), we examine the evolution of real-world networks over time. To make our analysis as general as possible, we look at several types of networks: social networks, collaboration networks, communication networks, authorship networks, rating networks, citation networks and folksonomies. Some networks, such as rating graphs, have edge weights, others have negative edges, such as friendship and enmity networks, and others are unweighted. The spectral evolution model also applies equally to unipartite and to bipartite networks.

The second test of the spectral evolution model (in Section 4) is theoretical: Several existing models of network growth predict spectral network evolution. We derive this for common graph kernels, including the matrix exponential, the von Neumann kernel, rank reduction methods, path counting, and community models. We also show that spectral evolution is specifically *not* implied by a random permutation model, rendering the spectral evolution model non-trivial and its observation in so many network types significant.

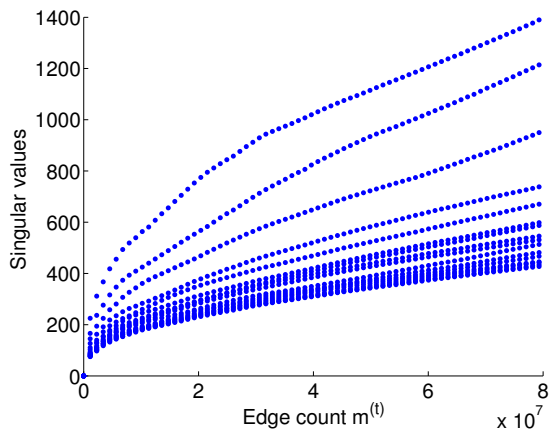
3. EXAMINING LARGE NETWORKS

In this section, we examine a collection of large network datasets in order to verify the spectral evolution model empirically for a large variety of network types. In total, we examined 119 large networks. A subset of these datasets is listed in Table 2. The networks under study are of the following types:

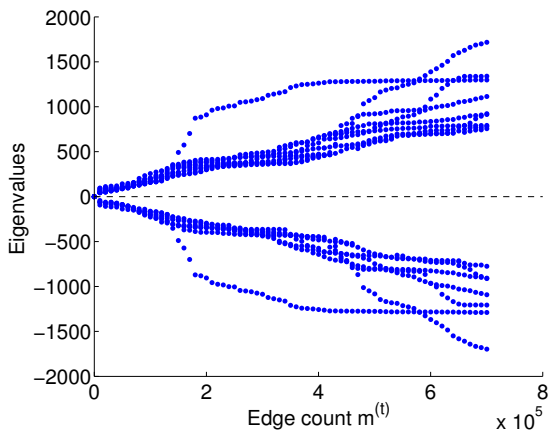
- **Social networks** are unipartite networks of persons connected by friendship links. A few social networks have enmity relations in addition to friendship relations, represented by the edge weights $+1/-1$. Only one edge is present between any node pair.
- **Collaboration networks** are networks of scientists connected by collaboration events, e.g. the publication of a common paper. Multiple edges may be present between any node pair.
- **Communication networks** are networks of computer users and emails sent to each other. Multiple edges may connect any two nodes.
- **Authorship graphs** are bipartite networks of Wikipedia project users and articles, where an edge represents a single edit. Multiple edges are possible.
- **Rating networks** are bipartite graphs between users and items, with edges representing a numerical assessment of an item by a user. Only one rating is possible for each user–item pair.
- **Citation networks** are unipartite networks of documents and links or citations between them. There are no multiple edges.
- **Folksonomies** are bipartite item–tag networks that denote assignments of tags to items. There may be multiple edges between any item–tag pair. The users of folksonomies are not included in our analysis.

In all cases, edge creation times are known, for instance the moment a friendship forms, or the publication date of a paper.

We will denote the symmetric adjacency matrix of a network by \mathbf{A} . For unweighted networks, this is a 0/1 matrix. For networks in which multiple edges are allowed, the entries are nonnegative integers. For signed networks, \mathbf{A} is



(a) Evolution of singular values



(b) Evolution of eigenvalues

Figure 2: The spectral evolution of large real-world networks. At each time, the graphs show the k dominant eigenvalues or singular values of the network at that time. (a) The bipartite Netflix rating graph, (b) The Facebook social network.

a $-1/0/+1$ matrix. In rating networks, the matrix entries are the rating values, from which the mean rating has been subtracted. Bipartite networks have special structure which can be exploited by working with the biadjacency matrix, whose singular values correspond to the absolute eigenvalues of the adjacency matrix. We will make all derivations for the eigenvalue decomposition, but understand that the singular value decomposition can be used analogously.

3.1 Spectral Evolution

For each network, we split the set of edges into $n = 71$ bins by edge arrival time. For each bin, we take a snapshot of the network after the arrival of that bin’s edges, and compute the first k eigenvalues or singular values of the resulting adjacency matrix, for symmetric and asymmetric networks respectively. We denote $\mathbf{A}_{(t)}$ the adjacency matrix of all edges present after time t . Thus $\mathbf{A}_{(n)}$ is the full adjacency matrix.

Figure 2 shows the spectra of several large networks as functions of time. The number of computed eigenvalues k is chosen in function of network size to give reasonable run-

times. A first inspection of these plots shows that the eigenvalues and singular values grow over time. The observed growth is sometimes regular, as in the case of Netflix, and sometimes irregular, as in the case of Facebook. A few observations can be made immediately: Eigenvalues can have growth patterns so different from each other that one may overtake another. In the Facebook network, one eigenvalue is even constant in the second half of the growth. These irregularities indicate that simple graph kernels such as the matrix exponential cannot work well, as they apply the same function to each eigenvalue, and thus cannot predict that a latent dimension is stagnant, even though this is easy to see by experiment.

For our analysis of spectral growth to be complete, spectra must not only grow, but eigenvectors must be stable. This is inspected in the next test.

3.2 Eigenvector Evolution

We have seen how network spectra change over time. We will now verify whether eigenvectors are stable as predicted by the spectral evolution model.

We compare, for each time t , the eigenvectors $\mathbf{A}_{(t)}$ with the eigenvectors of $\mathbf{A}_{(n)}$ of the complete network in Figure 3. The plots show one curve for each latent dimension k , showing the number of edges added between times t and n on the x-axis and the absolute dot product of the k^{th} eigenvector of \mathbf{A} at times t and n . Eigenvectors corresponding to large eigenvalues are shown in bright colors and those corresponding to smaller eigenvalues in transparent colors. These plots suggest the following interpretation: The general trend leaves the eigenvector similarity as measured by the dot product near one for the lifetime of the network, with similarity being higher for those eigenvectors with larger eigenvalues. In some networks such as YouTube, the similarity of eigenvectors suddenly drops to zero at specific points in time. As we will see, this is due to eigenvectors exchanging places in the decomposition, or in other words, the eigenvalues passing each other. The next test will inspect these permutations.

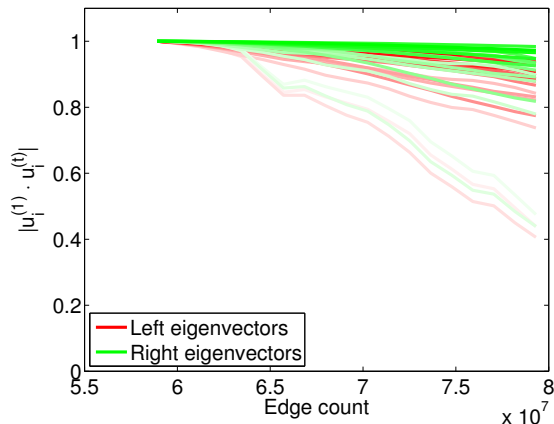
3.3 Eigenvector Stability

How stable are eigenvectors over time? To answer this question we compute the absolute eigenvalues of eigenvector pairs at two times t_1 and t_2 . At time t_1 , the networks contain 75% of all known edges, and at time t_2 they contain all known edges. Let

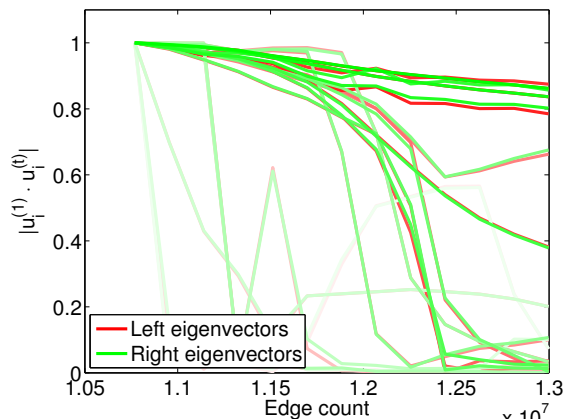
$$\begin{aligned}\mathbf{A}_{(1)} &= \mathbf{U}_{(1)}\mathbf{\Lambda}_{(1)}\mathbf{U}_{(1)}^T \\ \mathbf{A}_{(2)} &= \mathbf{U}_{(2)}\mathbf{\Lambda}_{(2)}\mathbf{U}_{(2)}^T\end{aligned}$$

be the eigenvalue decompositions at times t_1 and t_2 . We then compute the cosine similarities $|\mathbf{U}_{(1)\cdot i}^T\mathbf{U}_{(2)\cdot j}|$ between all pairs (i, j) of latent dimensions. We show the resulting matrices using white for zero and black for one, and continuous shades in between in Figure 4. These plots give an indication as to what extent eigenvalues are preserved over time. If all eigenvalues are distinct and network evolution is purely spectral, then these matrices are permutation matrices. In addition, they are diagonal unit matrices when the latent dimensions do not overtake each other. The derivation from a diagonal matrix then gives an indication as to the monotony of the underlying spectral transformation.

Testing the eigenvector stability in this way has one drawback. If two eigenvalues are (almost) equal, an exchange



(a) Stable eigenvector evolution



(b) Unstable eigenvector evolution

Figure 3: The evolution of eigenvectors: The similarity between the eigenvectors of the full graph and the eigenvectors of partial graphs, by increasing number of added edges. Each latent dimension is represented by one curve, with brighter colors for dominant latent dimensions. (a) The Netflix rating network, (b) The YouTube social network.

between their eigenvectors does not change the matrix by much. Therefore, we have to inspect more carefully the well-separated eigenvalues, which are larger. The next test is designed to be robust against such multiple eigenvalues.

3.4 Diagonality Test

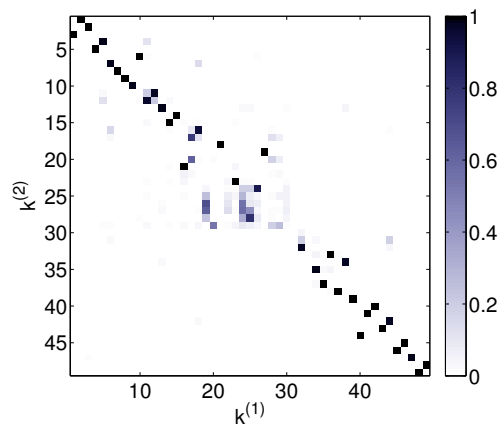
We apply our algorithm previously described in [16] to learn spectral transformations by finding a function of the diagonal eigenvalue matrix that best fits a certain spectral link prediction function. If

$$\mathbf{A}_{(1)} = \mathbf{U}_{(1)}\mathbf{\Lambda}_{(1)}\mathbf{U}_{(1)}^T$$

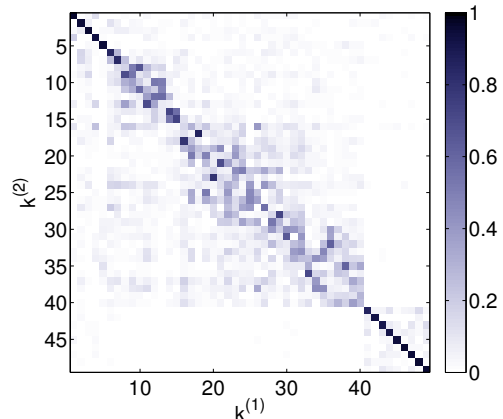
is the eigenvalue decomposition at time t_1 , then at time t_2 it is expected to become

$$\mathbf{A}_{(2)} = \mathbf{U}_{(1)}(\mathbf{\Lambda}_{(1)} + \mathbf{D})\mathbf{U}_{(1)}^T$$

where \mathbf{D} is diagonal.



(a) Precise spectral growth



(b) Imprecise spectral growth

Figure 4: The absolute dot product of all eigenvalue pairs at two times in network evolution. These plots show permutation matrices (with 0 in white and 1 in black) when the network evolution is purely spectral and eigenvalues are simple. The derivation from a diagonal matrix gives an indication as to the monotony of the underlying spectral transformation. (a) The Facebook social network, (b) The Flickr social network.

The best fit of \mathbf{D} in a least-squares sense is then given by

$$\tilde{\mathbf{D}} = \mathbf{U}_{(1)}(\mathbf{A}_{(2)} - \mathbf{A}_{(1)})\mathbf{U}_{(1)}^T.$$

The diagonality test plots in Figure 5 show the matrices $\tilde{\mathbf{D}}$ for several networks. These plots give an indication to what extent growth is spectral. If the growth between times t_1 and t_2 is purely spectral, then the matrix $\tilde{\mathbf{D}}$ is diagonal. The plots show this matrix to be diagonally dominant, indicating that the spectral evolution model is correct to a large extent, but not perfect.

In other words, there is a small amount of mixing between latent dimensions. The plots also show that the small off-diagonal values are not clustered near the main diagonal, implying that the small amount of mixing does not happen between adjacent latent dimensions specifically. We therefore interpret this mixing as noise rather than a feature of network evolution.

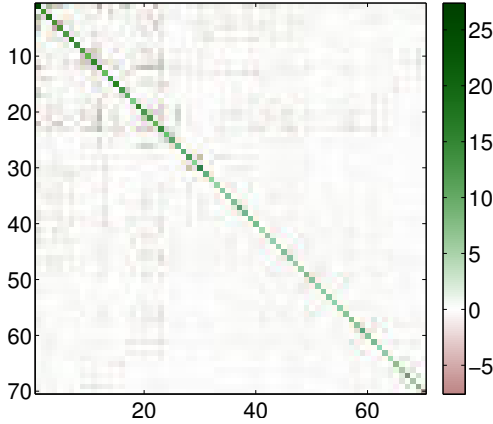


Figure 5: The diagonality test of spectral network evolution in the Facebook social network, showing a plot of the matrix \tilde{D} described in the text. If network evolution was perfectly spectral, the plot would show a clean diagonal.

3.5 Learning Graph Kernels

In Figure 2 we observed some graph spectra to evolve in a regular fashion and others not. However, these plots do not reveal which link prediction functions they match. To find out, we apply the method described in [16]. This method takes a network at an intermediate timeslice (the endpoint of the training set), and compares the intermediate and the final spectra and corresponding eigenvalues by reducing the comparison to a one-dimensional curve fitting problem. This is achieved by computing how the spectrum *should* look like in the final graph. The underlying idea is that if network growth corresponds to a certain spectral link prediction function, the new spectrum is a corresponding real function of the intermediate spectrum.

Let $\mathbf{A}_{(1)}$ and $\mathbf{A}_{(2)}$ be the adjacency matrices at times t_1 and t_2 , and $\mathbf{A}_{(1)} = \mathbf{U}_{(1)}\mathbf{\Lambda}_{(1)}\mathbf{U}_{(1)}^T$ the eigenvalue decomposition of $\mathbf{A}_{(1)}$. Assuming that graph growth follows a spectral transformation $F(\mathbf{A}_{(1)}) = \mathbf{A}_{(2)}$ leads to the optimization problem

$$\min_F \|\mathbf{F}(\mathbf{\Lambda}_{(1)}) - \mathbf{U}_{(1)}^T \mathbf{A}_{(2)} \mathbf{U}_{(1)}\|_2^2.$$

Since this problem is independent of the off-diagonal entries in $\mathbf{U}_{(1)}^T \mathbf{A}_{(2)} \mathbf{U}_{(1)}$, it can be reduced to the following form:

$$\min_f \sum_i [f(\mathbf{\Lambda}_{(1)ii}) - (\mathbf{U}_{(1)}^T \mathbf{A}_{(2)} \mathbf{U}_{(1)})_{ii}]^2,$$

where $f(\lambda)$ is the real function such that $F(\mathbf{\Lambda}_{(1)})_{ii} = f(\mathbf{\Lambda}_{(1)ii})$.

As an example, estimating the parameters of an exponential graph kernel $F(\mathbf{A}_{(1)}) = \beta \exp(\alpha \mathbf{A}_{(1)})$ amounts to solving

$$\min_{\alpha, \beta} \sum_i [\beta \exp(\alpha \mathbf{\Lambda}_{(1)ii}) - (\mathbf{U}_{(1)}^T \mathbf{A}_{(2)} \mathbf{U}_{(1)})_{ii}]^2.$$

This minimization problem is a one-dimensional nonlinear curve fitting problem that can be easily solved by existing methods¹. Figure 6 shows the results of curve fitting for several common graph kernels. While some networks

¹e.g., the function `lsqnonlin` in MATLAB

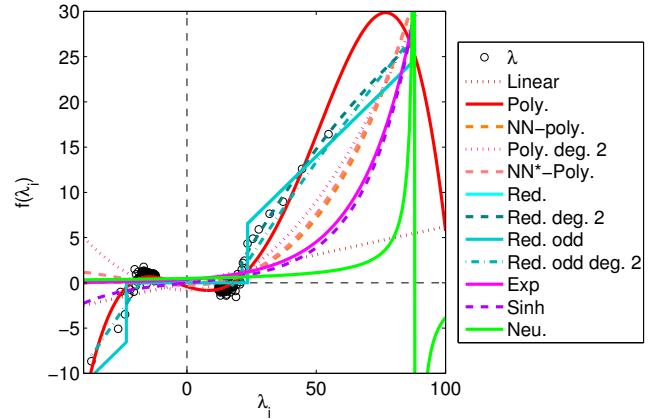


Figure 6: Comparing actual spectral transformations and several known graph kernels in the signed Wikipedia vote graph. In this plot, each curve represents a specific spectral transformation. If curve fitting is tight for a particular curve, the corresponding link prediction function describes the network growth well. On the other hand, outliers represent latent dimensions whose growth is not well described by known link prediction functions.

indeed display growth patterns that follow specific spectral link prediction functions, others do not. In particular, some networks seem to follow a regular spectral transformation except for single latent dimensions, which grow differently. This observation is the basis of the method described in Section 5: The spectral growth has to be learned independently of any graph kernel. First however, we test whether the spectral evolution model is justified by theoretical graph growth models.

4. SPECTRAL GROWTH MODELS

We have seen in Section 2 that the matrix exponential is a *spectral transformation*. In fact, several other link prediction algorithms are also spectral transformations. These methods show that the spectral evolution model arises naturally in different graph growth models such as diffusion models, path closing models and rank reduction models. At the same time, it is important to verify that the spectral evolution model is not trivial, i.e. that it does not arise simply by a random graph growth model. To do this, we first show how it follows from common link prediction models, and then show that it does *not* arise from a simple random graph growth model.

4.1 Graph Kernels

A certain number of link prediction algorithms are known as graph kernels [8]. In this context, a graph kernel is a positive-semidefinite function of two nodes of a given graph that denotes similarity or proximity². A graph kernel can be understood as a positive-semidefinite function $K(\mathbf{A})$ of a graph's adjacency matrix. If $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ is the eigenvalue decomposition of the adjacency matrix \mathbf{A} , then a spectral graph kernel can be written as $K(\mathbf{A}) = \mathbf{U}\mathbf{F}(\mathbf{\Lambda})\mathbf{U}^T$ for var-

²In another context, a graph kernel is a positive-semidefinite function of two (usually small) graphs.

ious functions $F(\mathbf{A})$ which apply a real function $f(\lambda)$ to each eigenvalue in \mathbf{A} . By construction, graph kernels can be computed from the eigenvalue decomposition of \mathbf{A} . The following are common choices for graph kernels:

Matrix Exponential.

The matrix exponential of the adjacency matrix can be used as a link prediction function [15].

$$K(\mathbf{A}) = \exp(\alpha \mathbf{A}) = \sum_{i=0}^{\infty} \frac{\alpha^i}{i!} \mathbf{A}^i$$

As mentioned in Section 2, the exponential graph kernel denotes the sum over all paths between any two nodes, weighted by the inverse factorial of path length. The exponential graph kernel corresponds to the spectral transformation $f(\lambda) = e^{\alpha\lambda}$.

The von Neumann Kernel.

The von Neumann kernel arises when considering a diffusion process in which a certain amount of flow is lost on each edge [13].

$$K(\mathbf{A}) = (\mathbf{I} - \alpha \mathbf{A})^{-1}$$

where α is chosen such that $\alpha^{-1} > |\lambda_1|$, where $|\lambda_1|$ is the largest absolute eigenvalue of \mathbf{A} . The resulting spectral transformation is $f(\lambda) = 1/(1 - \alpha\lambda)$. The von Neumann kernel is also called the Katz index when it is used for link prediction [22].

Rank Reduction.

For large graphs, the eigenvalue and singular value decompositions can only be computed up to a small rank k , in practice not greater than about 100. The fact that link prediction using a reduced rank achieves reasonable results is explained by rank reduction being a good link prediction method in itself. Thus, varying k results in varying link prediction accuracy [28]. Reduction to a rank k can be interpreted as the spectral transformation $f(\lambda) = \lambda$ when $|\lambda| \geq |\lambda_k|$, and $f(\lambda) = 0$ otherwise, where $|\lambda_k|$ is the k^{th} largest absolute eigenvalue of \mathbf{A} .

Path Counting.

We discussed that the exponential graph kernel can be understood as giving each path an importance equal to the inverse factorial of its length. This can be generalized to any weight function, and results in matrix polynomials $P(\mathbf{A})$ and matrix powers as special cases. In the simplest case, this is the square \mathbf{A}^2 , which corresponds to the *triangle closing* model of link prediction [20], since it counts the number of common neighbors of all vertex pairs. The corresponding spectral transformation is simply the polynomial P applied to each eigenvalue separately.

4.2 Community Model

If a symmetric matrix \mathbf{A} is updated by a rank-one matrix uu^T , then that update is a spectral transformation when u is an eigenvector of \mathbf{A} [6]. This spectral transformation updates the eigenvalue corresponding to u by $|u|^2$. If a latent dimension is interpreted as a subcommunity of a network described by an eigenvector u , an update of uu^T can be understood as a change within that community. In a user–user email network for instance, an update of uu^T cor-

responds to new email events among the subcommunity of users represented by u . The relative growth of individual latent dimensions is not predicted by this community model, justifying the spectral network model but not any specific graph kernel.

4.3 Irregular Growth

In several real-world networks, the growth is a spectral transformation, but does not follow any specific spectral transformation function. For instance, a latent dimension may stop growing and then level out, as in Figure 2(b) for the Facebook social network. As this plot shows, a stagnating latent dimension may also restart growing. There may be several possible explanations for this behavior. The Wiki authorship datasets are particularly useful in this regard, because the names of users and items are known, allowing us to give interpretations of network growth. For many Wiki authorship networks, these dimensions represent automatic bots or other exceptional editing. We made the following observations:

- Simple change in the network: A certain book on the French Wikibooks becoming inactive (build your house—construire sa maison), and large but temporal discussions (Italian and Japanese Wikipedias)
- Automatic addition of edges: Wikipedia bots editing articles and mass imports of articles (Russian Wikipedia)
- Global disruptions in the network (Enron)

In most cases however, the reason for irregular but spectral growth is unknown because no vertex labels are provided with the datasets. In these cases, the irregularity can nonetheless be learned because it is spectral.

4.4 Random Perturbation Model

The graph kernels reviewed in this section all predict spectral growth, which leads to the question whether non-spectral growth is possible under other graph growth models. Let $\mathbf{A}_{(t)} = \mathbf{U}\mathbf{A}\mathbf{U}^T$ be the adjacency matrix of a network, and $\mathbf{A}_{(t+1)} = \mathbf{A}_{(t)} + \mathbf{E}$ a perturbation of $\mathbf{A}_{(t)}$ where $\|\mathbf{E}\|_2 = \epsilon$ is small and \mathbf{E} can be thought of as an infinitesimally small edge added to the network. A perturbation argument then shows that the eigenvalue decomposition of $\mathbf{A}_{(t+1)} = \tilde{\mathbf{U}}\tilde{\mathbf{A}}\tilde{\mathbf{U}}^T$ has the following bounds [30]:

$$\begin{aligned} \|\mathbf{A} - \tilde{\mathbf{A}}\|_F &= O(\epsilon^2) \\ |\mathbf{U}_{\cdot k}^T \tilde{\mathbf{U}}_{\cdot k}| &= O(\epsilon) \end{aligned}$$

As a result, eigenvectors are expected to change faster than eigenvalues for random additions to the adjacency matrix. We were able to confirm this prediction in synthetic random tests, which shows that spectral growth is not explained by a random graph model and is thus a nontrivial emergent property of real-world networks.

5. SPECTRAL EXTRAPOLATION

We now derive, as an application of the spectral evolution model, a new algorithm for link prediction. According to the spectral evolution model established in the previous sections, the spectrum of a network evolves over time, while the eigenvectors remain constant. We saw that this assumption is true to a good approximation in large real-world networks.

In this section, we will exploit this fact to derive a new algorithm for link prediction. We assume that a given network grows spectrally, but do not make any other assumptions. In particular, we will not assume that growth follows any spectral transformation function implied by a specific graph kernel. For this reason, our new link prediction algorithm is a generalization of spectral graph kernels.

5.1 Regular and Irregular Functions

Graph kernels such as the matrix exponential assume there is a real function $f(\lambda)$ which describes the growth of all eigenvalues. In the graph kernels of Section 4, this function is very regular; it is positive and convex. Learned spectral transformations such as the one in Figure 6 are however irregular, and no simple function solves the resulting curve fitting problem well. This is consistent with the observation that some eigenvalues cross each other, indicating that a non-monotonous spectral transformation function is needed.

For these reasons, we consider the growth of each latent dimension separately. If we knew that eigenvalues did not cross each other, we could compare the i^{th} eigenvalue at two points in time and extrapolate a third value. But because eigenvalues pass each other, we must learn the correspondence between new and old eigenvalues.

5.2 Algorithm Description

We now describe our algorithm for predicting links by extrapolation of spectral growth. We describe our method for symmetric graphs and the eigenvalue decomposition. For bipartite graphs, the singular value decomposition can be used analogously. To minimize the amount of computation needed, we base the extrapolation on only two times: t_1 , an intermediate time, and t_2 , the final time where the training set is complete. Therefore, our method only requires the computation of two eigenvalue decompositions.

We denote by $\mathbf{A}_{(1)}$ and $\mathbf{A}_{(2)}$ the adjacency matrices at times t_1 and t_2 , and then consider the eigenvalue decomposition at each time t :

$$\mathbf{A}_{(t)} = \mathbf{U}_{(t)} \mathbf{\Lambda}_{(t)} \mathbf{U}_{(t)}^T$$

To find out which latent dimension i at t_1 corresponds to latent dimension j at t_2 , we compute a mean of eigenvalues at t_1 , weighted by the similarity between the two latent dimensions at t_1 and t_2 . As a similarity measure, we use the dot product between the corresponding normalized eigenvectors³. Thus if $\lambda_{(2)j}$ is an eigenvalue at t_2 , its estimated previous value at t_1 is

$$\hat{\lambda}_{(1)j} = \left(\sum_i \mathbf{U}_{(1)\cdot i}^T \mathbf{U}_{(2)\cdot j} \right)^{-1} \sum_i \mathbf{U}_{(1)\cdot i}^T \mathbf{U}_{(2)\cdot j} \lambda_{(1)i}$$

In other words, the values $\hat{\lambda}_{(1)j}$ are an estimate of the diagonalization of $A_{(1)}$ by $U_{(2)}$. We can then perform linear extrapolation to predict an eigenvalue $\hat{\lambda}_{(3)j}$ in the future.

$$\hat{\lambda}_{(3)j} = 2\lambda_{(2)j} - \hat{\lambda}_{(1)j}$$

Using the matrix $\hat{\mathbf{A}}_{(3)} = (\hat{\lambda}_{(3)j})$, the predicted edge weights are then $\mathbf{U}_{(2)} \hat{\mathbf{A}}_{(3)} \mathbf{U}_{(2)}^T$. Note that the computed eigenvalues $\hat{\lambda}_{(3)j}$ are not necessarily ordered by absolute value, instead they retain the ordering of $\mathbf{U}_{(2)}$.

³We also experimented with various powers of the dot product, but results were mostly identical.

Table 1: Summary of link prediction methods.

	Function	Spectral transformation
P	Polynomial	$f(\lambda) = \sum_i \alpha_i \lambda^i$
NNP	Nonneg. polynomial	$f(\lambda) = \sum_i \alpha_i \lambda^i, \alpha_i \geq 0$
EXP	Matrix exponential	$f(\lambda) = e^{\alpha \lambda}$
RED	Rank reduction	$f(\lambda) = \lambda$ when $ \lambda \geq \lambda_k $, $f(\lambda) = 0$ otherwise
AA	Adamic/Adar	–
NB	Common neighbors	–
EXT	Spectral extrapolation	Irregular

5.3 Experimental Setup

We compare the performance of the spectral extrapolation algorithm to other link prediction methods on the task of predicting links in the collection of network datasets shown in Table 2. We split each network into training, validation and test sets of edges. The splits are chosen by timestamps. Each test set contains 30% of the total number of edges, and each validation set contains 30% of the number edges in each combined training and validation set. We then prune all nodes outside the giant connected component in the training set, in order to avoid predicting edges between unconnected parts of the network.

As a performance measure, we use the mean average precision (MAP) [23] in the following way: For each vertex adjacent to edges in the test set, we compute the average precision of predicted edges, and compute the mean over all these average precisions. For unweighted and positively weighted networks (such as communication networks), the task is to predict the location of edges, and the mean average precision is computed in relation to that goal, i.e. we count an edge as correctly predicted if it is present at least once in the test set. For other edge weights, we count an edge as correctly predicted when it has positive weight for signed networks, or if it has weight greater than the average for rating networks.

We compare the spectral extrapolation method with graph kernels and non-spectral link prediction methods. The graph kernels are learned using the method of [16]. The non-spectral link prediction methods are those described in [22]. A summary of all evaluated methods is given in Table 1.

5.4 Evaluation Results

Figure 7 shows the extrapolation method applied to the English Wikipedia hyperlink graph. The numerical results are in Table 2.

These results show that the spectral extrapolation method outperforms graph kernels in some cases, but not always. In several cases, graph kernels learned using the method of [16] predict links with higher accuracy than spectral extrapolation. This can be explained either by growth following graph kernels very closely, e.g. the Wikipedia votes in Figure 6, or by overfitting, as for the Swedish Wikipedia.

The Wikipedia edit graphs are available for all Wikipedia languages separately, and a cross comparison suggests that the spectral extrapolation method works better for large graphs. The English Wikipedia is not included as it was too large to process, with about 200 million edits.

We can conclude from the experiments that the spectral evolution model results in a single link prediction algorithm

Table 2: Overview of datasets and results of our experiments. The link prediction accuracy results are reported in the mean average precision (MAP) measure. We highlight well-performing cases: The overall best performance numbers are in bold and if spectral extrapolation has better than average performance, its numbers are in italics. The list of prediction algorithms is given in Table 1.

Network	Nodes	Edges	P	NNP	EXP	RED	AA	NB	EXT	Ref.
Social networks (person-person)										
Advogato	6,518	51,726	0.758	0.878	0.881	0.672	0.933	0.931	<i>0.881</i>	[29]
Epinions	131,828	841,372	0.778	0.794	0.794	0.785	0.862	0.860	<i>0.844</i>	[24]
Facebook	63,891	876,993	0.744	0.743	0.742	0.560	0.835	0.835	<i>0.749</i>	[32]
Flickr	2,302,925	33,140,018	0.647	0.636	0.605	0.500	–	–	0.577	[26]
LfbmSeTi	220,970	17,359,346	0.619	0.619	0.619	0.441	0.467	0.466	0.619	[4]
Slashdot Zoo	79,120	515,581	0.826	0.824	0.820	0.728	0.798	0.797	<i>0.821</i>	[17]
YouTube	3,223,643	18,524,095	0.504	0.530	0.521	0.499	0.592	0.592	<i>0.551</i>	[26]
Wikipedia talk	2,394,385	5,021,410	0.640	0.662	0.655	0.501	0.640	0.640	0.673	[19]
Wikipedia votes	7,115	103,689	0.867	0.945	0.929	0.637	0.839	0.840	<i>0.917</i>	[19]
Collaboration networks (person-person)										
Astro-ph	18,772	396,160	0.870	0.869	0.867	0.579	0.982	0.982	<i>0.870</i>	[19]
DBLP	722,383	8,302,144	0.751	0.750	0.751	0.611	–	–	0.755	[21]
Communication networks (person-person)										
Enron emails	87,365	1,149,884	0.870	0.871	0.870	0.670	0.908	0.913	0.795	[14]
EU emails	265,214	420,045	0.852	0.968	0.961	0.759	0.921	0.921	0.974	[19]
Authorship networks (author-article)										
Dutch Wikipedia	98,343+1,360,344	16,135,919	0.731	0.693	0.728	0.611	–	–	<i>0.730</i>	[33]
French Wikipedia	236,982+3,479,808	38,015,550	0.681	0.744	0.747	0.655	–	–	0.772	[33]
German Wikipedia	368,375+2,846,674	48,924,295	0.654	0.685	0.685	0.649	–	–	0.722	[33]
Italian Wikipedia	115,948+1,973,838	21,635,876	0.752	0.759	0.759	0.629	–	–	<i>0.742</i>	[33]
Japanese Wikipedia	160,993+1,654,657	17,267,982	0.738	0.749	0.752	0.614	–	–	0.763	[33]
Polish Wikipedia	91,033+1,153,614	14,969,191	0.731	0.713	0.715	0.609	–	–	<i>0.722</i>	[33]
Portuguese Wikipedia	123,296+2,032,116	13,043,612	0.646	0.674	0.674	0.612	–	–	0.736	[33]
Russian Wikipedia	97,000+1,639,565	15,633,090	0.701	0.734	0.734	0.644	–	–	0.769	[33]
Spanish Wikipedia	278,739+2,132,836	21,634,713	0.668	0.743	0.743	0.649	–	–	0.792	[33]
Swedish Wikipedia	57,862+861,805	8,457,086	0.681	0.674	0.674	0.591	–	–	<i>0.678</i>	[33]
Rating networks (person-item)										
BookCrossing	105,283+340,532	1,149,780	0.840	0.841	0.841	0.833	–	–	0.836	[35]
Jester	24,938+100	616,912	0.763	0.757	0.769	0.671	–	–	<i>0.758</i>	[9]
MovieLens	71,567+65,133	10,000,054	0.755	0.755	0.727	0.642	–	–	0.759	[10]
Netflix	480,189+17,770	100,480,507	0.559	0.561	0.560	0.416	–	–	0.561	[2]
Citation/Link networks (document-document)										
arXiv Hep-ph	28,093	12,730,098	0.697	0.703	0.700	0.592	0.757	0.757	<i>0.748</i>	[19]
Citeseer	723,131	1,764,929	0.677	0.677	0.677	0.501	0.644	0.643	0.684	[3]
Patents	3,774,768	16,522,438	0.666	0.665	0.665	0.500	–	–	0.666	[11]
Wikipedia links	1,870,709	39,953,145	0.634	0.639	0.630	0.500	–	–	<i>0.624</i>	[25]
WWW	325,729	1,497,135	0.681	0.685	0.685	0.501	0.687	0.687	<i>0.684</i>	[1]
Folksonomies (document-tag)										
BibSonomy	4,969+624,020	1,935,143	0.609	0.710	0.707	0.553	–	–	0.711	[12]
CiteULike	731,769+153,277	2,411,819	0.632	0.644	0.644	0.503	–	–	<i>0.642</i>	[7]

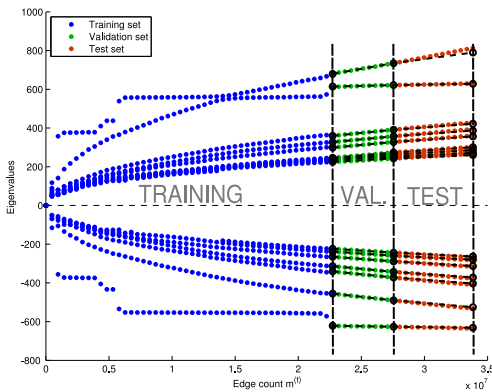


Figure 7: Applying spectral extrapolation to the Wikipedia hyperlink graph. Spectral extrapolation is computed using only the decompositions at the split points between the training and validation and test sets.

suitable for networks that grow spectrally, and that makes the choice of a specific graph kernel obsolete. As we showed in Section 3, networks from all application areas are observed to grow spectrally. Therefore, the spectral extrapolation model can be applied universally to link prediction problems of any kind, replacing individual graph kernels in all kinds of link prediction applications.

6. RELATED WORK

6.1 Avoided Crossings

As observed in several examples, a latent dimension can overtake another. In these cases, our model predicts *crossings* in the spectral plot. Looking more closely at actual spectra, we however observe something different: The smaller, growing eigenvalue slows down growth before it reaches the larger eigenvalue, and the larger eigenvalue starts growing at about the same rate. We observe this behavior in several datasets, as shown in Figure 8.

This phenomenon is called an *avoided crossing* and can be explained as follows [18, 8.5]. Let \mathbf{A} and \mathbf{B} be two symmetric rank-1 matrices of the same size. Then the eigenvalues of $\mathbf{A} + t\mathbf{B}$ for a real parameter t display the avoided crossing

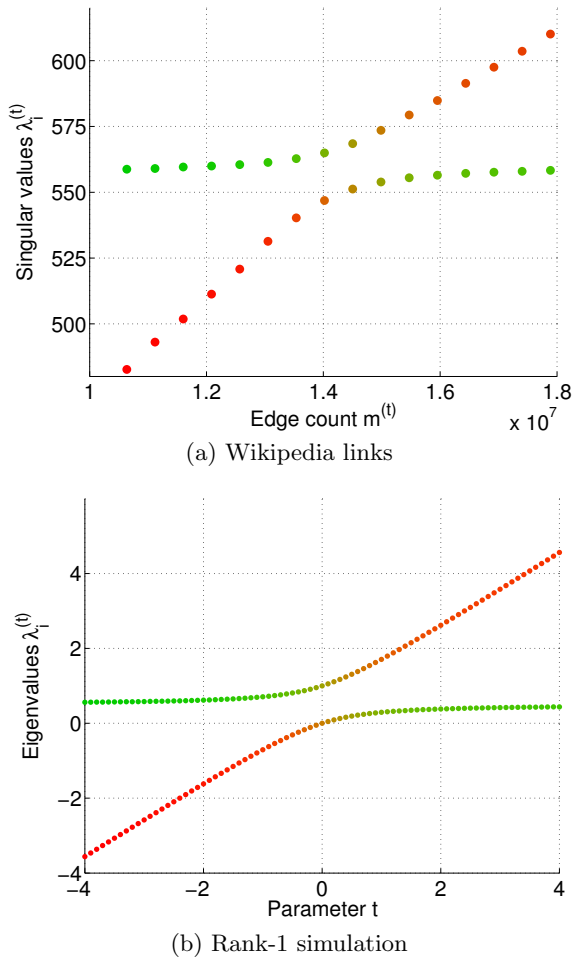


Figure 8: An avoided crossing as observed in the growth of the Wikipedia link network’s two dominant eigenvalues and eigenvectors in (a), and in a rank-1 model in (b). The red and green color components denote the cosine distance of eigenvectors to initial eigenvectors.

behavior. This behavior is explained by the fact that the two eigenvectors of \mathbf{A} and \mathbf{B} are not orthogonal in the general case and by considering the dimension of matrices with multiple eigenvalues.

In the rank-1 case, let $\mathbf{A} = \alpha\alpha^T$ and $\mathbf{B} = \beta\beta^T$, then α and $t\beta$ are only eigenvalues of $\mathbf{A} + t\mathbf{B}$ if α and β are collinear or orthogonal. Otherwise, an avoided crossing is observed as in Figure 8, where the color and brightness of points encodes the dot product of the corresponding eigenvectors with the initial eigenvectors, using green for the first and red for the second eigenvector. The plots show the two dominant singular values of the Wikipedia link network, and a rank-1 simulation.

Thus, an avoided crossing indicates that a decomposition into outer products of orthogonal vectors is not the natural representation for some networks. If the true decomposition using a and b is used, the crossing would be *unavoided*. Alternatively, an avoided crossing may result from a perturbation of orthogonal latent dimensions in the same manner.

6.2 Joint Diagonalization

A related but different link prediction method consists of computing a *joint diagonalization* of a network’s adjacency matrices at different times [31]. By construction, the resulting network evolution is spectral, and the approximation at each timepoint is less precise than the eigenvalue or singular value decomposition as described in this paper.

In fact, our spectral evolution model provides a justification for these advanced methods, since joint diagonalization implies constant or near-constant eigenvectors.

6.3 Spectral Kernel Learning

The spectral transformation of kernel matrices has been used before in semi-supervised learning settings [5, 34]. These previous works however do not observe which spectral transformation applies to the used datasets, and are restricted to basic spectral transformations such as rank reduction and linear kernels. These works also do not take into account the temporal evolution of the data.

6.4 Laplacian Matrix

Other graph kernels than those presented here can be defined as the spectral transformation of either the combinatorial or normalized Laplacian matrix $D - A$, where D is the diagonal degree matrix. In our experiments, we found the Laplacian eigenvalues to not grow in a regular fashion, but instead “jump” at unpredictable times. In the literature, these kernels are mostly used for spectral clustering rather than link prediction [27].

7. DISCUSSION

The spectral evolution model states that in many real-world networks, growth can be described by a change of the spectrum, while the corresponding eigenvectors remain constant. This assumption is true to a large extent in most networks we analysed in many different types of networks. A limitation however exist: The observed phenomenon of avoided crossings hints at non-orthogonal latent dimensions.

We then derived a link prediction algorithm based on the spectral evolution model. This method generalizes several spectral link prediction functions. In actual networks, this method provides more accurate link prediction in many cases, in particular in networks with irregular but still spectral growth. For networks with very regular growth, regular graph kernels perform better however. Across all datasets, the performance of our method was better than any single link prediction method. Our new extrapolation method is also parameter free: Not only are there no parameters, as in various graph kernels, but our method makes the choice of a specific spectral growth model unnecessary.

Comparing different types of networks, the spectral evolution model seems to be true for bipartite user-item networks and true to a lesser extent for user-user networks. We conjecture an application of spectral evolution to non-orthogonal matrix decompositions, based on the observations of avoided crossings. We conjecture that a step towards this goal may be to consider the non-diagonal entries of $\mathbf{U}_{(1)}^T \mathbf{A}_{(2)} \mathbf{U}_{(1)}$.

Finally, the spectral evolution model provides the justification for more complex link predictions methods such as those based on tensor decomposition, by interpreting the eigenvalue decomposition over time as a joint diagonalization problem.

Acknowledgment. This research has been co-funded by the EU in FP7 in the WeKnowIt project (215453).

8. REFERENCES

- [1] R. Albert, H. Jeong, and A.-L. Barabási. The diameter of the World Wide Web. *Nature*, 401:130, 1999.
- [2] J. Bennett and S. Lanning. The Netflix prize. In *Proc. KDD Cup*, pages 3–6, 2007.
- [3] K. Bollacker, S. Lawrence, and C. L. Giles. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proc. Int. Conf. on Autonomous Agents*, pages 116–123, 1998.
- [4] L. Brožovský and V. Petříček. Recommender system for online dating service. In *Proc. Znalosti*, pages 29–40, 2007.
- [5] O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 585–592, 2003.
- [6] J. Ding and A. Zhou. Eigenvalues of rank-one updated matrices with some applications. *Applied Mathematics Letters*, 20(12):1223–1226, 2007.
- [7] K. Emamy and R. Cameron. CiteULike: A researcher’s social bookmarking service. *Ariadne*, (51), 2007.
- [8] F. Fouss, L. Yen, A. Pirotte, and M. Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *Proc. Int. Conf. on Data Mining*, pages 863–868, 2006.
- [9] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
- [10] GroupLens Research. MovieLens data sets. <http://www.grouplens.org/node/73>, October 2006.
- [11] B. H. Hall, A. B. Jaffe, and M. Trajtenberg. The NBER patent citations data file: Lessons, insights and methodological tools. In *NBER Working Papers 8498*, National Bureau of Economic Research, Inc, 2001.
- [12] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. BibSonomy: A social bookmark and publication sharing system. In *Proc. Workshop on Conceptual Structure Tool Interoperability*, pages 87–102, 2006.
- [13] J. Kandola, J. Shawe-Taylor, and N. Cristianini. Learning semantic similarity. In *Advances in Neural Information Processing Systems*, pages 657–664, 2002.
- [14] B. Klimt and Y. Yang. The Enron corpus: A new dataset for email classification research. In *Proc. Eur. Conf. on Machine Learning*, pages 217–226, 2004.
- [15] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proc. Int. Conf. on Machine Learning*, pages 315–322, 2002.
- [16] J. Kunegis and A. Lommatzsch. Learning spectral graph transformations for link prediction. In *Proc. Int. Conf. on Machine Learning*, pages 561–568, 2009.
- [17] J. Kunegis, A. Lommatzsch, and C. Bauchhage. The Slashdot Zoo: Mining a social network with negative edges. In *Proc. Int. World Wide Web Conf.*, pages 741–750, 2009.
- [18] P. D. Lax. *Linear Algebra and Its Applications*. John Wiley & Sons, 1984.
- [19] J. Leskovec. Stanford network analysis project. <http://snap.stanford.edu/>, March 2010.
- [20] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 462–470, 2008.
- [21] M. Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *Proc. Int. Symposium on String Processing and Information Retrieval*, pages 1–10, 2002.
- [22] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. Int. Conf. on Information and Knowledge Management*, pages 556–559, 2003.
- [23] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [24] P. Massa and P. Avesani. Controversial users demand local trust metrics: an experimental study on epinions.com community. In *Proc. American Association for Artificial Intelligence Conf.*, pages 121–126, 2005.
- [25] A. Mislove. *Online Social Networks: Measurement, Analysis, and Applications to Distributed Information Systems*. PhD thesis, Rice University, 2009.
- [26] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Growth of the Flickr social network. In *Proc. Workshop on Online Social Networks*, pages 25–30, 2008.
- [27] A. Radl, U. v. Luxburg, and M. Hein. The resistance distance is meaningless for large random geometric graphs. In *Proc. Workshop on Analyzing Networks and Learning with Graphs*, 2009.
- [28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems—a case study. In *Proc. ACM WebKDD Workshop*, 2000.
- [29] D. Stewart. Social status in an open-source community. *American Sociological Review*, 70(5):823–842, 2005.
- [30] G. W. Stewart. Perturbation theory for the singular value decomposition. Technical report, Univ. of Maryland, College Park, 1990.
- [31] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 374–383, 2006.
- [32] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in Facebook. In *Proc. Workshop on Online Social Networks*, pages 37–42, 2009.
- [33] Wikimedia Foundation. Wikimedia downloads. <http://download.wikimedia.org/>, January 2010.
- [34] X. Zhu, J. Kandola, J. Lafferty, and Z. Ghahramani. *Semi-supervised Learning*, chapter Graph Kernels by Spectral Transforms. MIT Press, 2006.
- [35] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proc. Int. World Wide Web Conf.*, pages 22–32, 2005.