Mälardalen University Press Dissertations
No. 217

# PAGERANK IN EVOLVING NETWORKS AND APPLICATIONS OF GRAPHS IN NATURAL LANGUAGE PROCESSING AND BIOLOGY

**Christopher Engström**

**2016**


School of Education, Culture and Communication

ISBN 978-91-7485-298-1
ISSN 1651-4238
Printed by Arkitektkopia, Västerås, Sweden

Mälardalen University Press Dissertations
No. 217

# PAGERANK IN EVOLVING NETWORKS AND APPLICATIONS OF GRAPHS IN NATURAL LANGUAGE PROCESSING AND BIOLOGY

Christopher Engström

Akademisk avhandling

som för avläggande av filosofie doktorsexamen i matematik/tillämpad matematik
vid Akademin för utbildning, kultur och kommunikation kommer att offentligen
försvaras torsdagen den 8 december 2016, 13.15 i Kappa, Mälardalens högskola, Västerås.

Fakultetsopponent: Professor Raimondo Manca, Sapienza University of Rome


Akademin för utbildning, kultur och kommunikation

Abstract

This thesis is dedicated to the use of graph based methods applied to ranking problems on the Web-graph and applications in natural language processing and biology.

Chapter 2-4 of this thesis is about PageRank and its use in the ranking of home pages on the Internet for use in search engines. PageRank is based on the assumption that a web page should be high ranked if it is linked to by many other pages and/or by other important pages. This is modelled as the stationary distribution of a random walk on the Web-graph.

Due to the large size and quick growth of the Internet it is important to be able to calculate this ranking very efficiently. One of the main topics of this thesis is how this can be made more efficiently, mainly by considering specific types of subgraphs and how PageRank can be calculated or updated for those type of graph structures. In particular we will consider the graph partitioned into strongly connected components and how this partitioning can be utilized.

Chapter 5-7 is dedicated to graph based methods and their application to problems in Natural language processing. Specifically given a collection of texts (corpus) we will compare different clustering methods applied to Pharmacovigilance terms (5), graph based models for the identification of semantic relations between biomedical words (6) and modifications of CValue for the annotation of terms in a corpus.

In Chapter 8-9 we look at biological networks and the application of graph centrality measures for the identification of cancer genes. Specifically in (8) we give a review over different centrality measures and their application to finding cancer genes in biological networks and in (9) we look at how well the centrality of vertices in the true network is preserved in networks generated from experimental data.

ISBN 978-91-7485-298-1
ISSN 1651-4238

# Acknowledgements

I would like to start by thanking all my colleagues at Mälardalen University for their help and support during the work of this thesis.

In particular I would like to thank my main supervisor Professor Sergei Silvestrov for all your support and guidance throughout this work.

I would like to thank Karl Lundengård, Jonas Österberg and all the other current or former PhD students during these years for any suggestions, helpful comments and a nice overall atmosphere to work in. I would also like to thank Dr. Ying Ni and Professor Anatoliy Malyarenko for the nice collaboration and discussions related to data mining of energy consumption data as well as Professor Dmitrii Silvestrov for the many discussions and guidance on topics related to markov chains.

I acknowledge my co-authors, for introducing me to exciting applications and for many fruitful discussions. In particular I want to thank Thierry Hamon for introducing me to the field of Natural Language Processing and Holger Weishaupt for the many interesting discussions related to networks and cancer research.

Finally I want to thank some people close to me, my parents for always being supportive and Alexander Hedbrant for always being there, good luck with your new addition to the family, if you ever find yourself in need of a bedtime story then feel free to make use of this thesis.

*Västerås*, October, 2016
Christopher Engström

# List of Papers

This thesis contains the following papers:

Paper A. Engström, Christopher and Silvestrov, Sergei (2015), *Non-normalized PageRank and random walks on N-partite graphs*, 3rd Stochastic Modeling Techniques and Data Analysis International Conference (SMTDA2014), 193–202.

Paper B. Engström, Christopher and Silvestrov, Sergei (2016), *PageRank, a Look at Small Changes in a Line of Nodes and the Complete Graph*, in Silvestrov S., Rančić M., (eds.), Engineering Mathematics II: Algebraic, Stochastic and Analysis Structures for Networks, Data Classification and Optimization, Springer Proceedings in Mathematics and Statistics, Springer, 2194–1009; 179.

Paper C. Engström, Christopher and Silvestrov, Sergei (2016), *PageRank, Connecting a Line of Nodes with a Complete Graph*, in Silvestrov S., Rančić M., (eds.), Engineering Mathematics II: Algebraic, Stochastic and Analysis Structures for Networks, Data Classification and Optimization, Springer Proceedings in Mathematics and Statistics, Springer, 2194–1009; 179.

Paper D. Engström, Christopher and Silvestrov, Sergei (2015), *A componentwise PageRank algorithm*, Applied Stochastic Models and Data Analysis (ASMDA 2015). The 16th Conference of the ASMDA International Society, 186–199.

Paper E. Engström, Christopher and Silvestrov, Sergei (2016), *Graph partitioning and a component wise PageRank algorithm*, submitted to journal, arXiv:1609.09068 [cs.DS]

Paper F. Engström, Christopher and Silvestrov, Sergei (2016), *Using graph partitioning to calculate PageRank in a changing network*, to appear in 4th Stochastic Modeling Techniques and Data Analysis International Conference (SMTDA2016).

Paper G. Engström, Christopher and Silvestrov, Sergei (2016), *Calculating PageRank in a Changing NetworkWith Added or Removed Edges*, to appear in Proceedings of ICNPAA2016.

Paper H. Dupuch, Marie. Engström, Christopher. Silvestrov, Sergei. Hamon, Thierry. Grabar, Natalia (2013), *Comparison of Clustering Approaches through Their Application to Pharmacovigilance Terms* Artificial Intelligence in Medicine: 14th Conference on Artificial Intelligence in Medicine, AIME 2013 Proceedings, 58–67.

Paper I. Hamon, Thierry. Engström, Christopher. Manser, Mounira. Badji, Zina. Grabar, Natalia. Silvestrov, Sergei (2012), *Combining Compositionality and Pagerank for the Identification of Semantic Relations Between Biomedical Words*, Proceedings of the 2012 Workshop on Biomedical Natural Language Processing, BioNLP '12, 109–117.

Paper J. Hamon, Thierry. Engström, Christopher. Silvestrov, Sergei (2014), *Term Ranking Adaptation to the Domain: Genetic Algorithm-Based Optimisation of the C-Value*, Advances in Natural Language Processing: 9th International Conference on NLP, PolTAL 2014, 71–83.

Paper K. Weishaupt, Holger. Johansson, Patrik. Engström, Christopher. Nelander, Sven. Silvestrov, Sergei. Swartling, Fredrik J. (2016), *Graph Centrality Based Prediction of Cancer Genes*, in Silvestrov S., Rančić M., (eds.), Engineering Mathematics II: Algebraic, Stochastic and Analysis Structures for Networks, Data Classification and Optimization, Springer Proceedings in Mathematics and Statistics, Springer, 2194–1009; 179.

Paper L. Weishaupt, Holger. Johansson, Patrik. Engström, Christopher. Nelander, Sven. Silvestrov, Sergei. Swartling, Fredrik J. (2015), *Loss of conservation of graph centralities in reverse-engineered transcriptional regulatory networks*, Applied Stochastic Models and Data Analysis (ASMDA 2015). The 16th Conference of the ASMDA International Society, 1077–1091.

The following papers are *not* included in the thesis

- Engström, Christopher and Silvestrov, Sergei (2014) *An evaluation of centrality measures used in cluster analysis*, AIP Conference Proceedings (ICNPAA2014), 1637:1, 313–320.

- Engström, Christopher and Silvestrov, Sergei (2014) *Generalisation of the Damping Factor in PageRank for Weighted Networks*, Silvestrov, Dmitrii and Martin-Löf, Anders (eds.), Modern Problems in Insurance Mathematics, 313–333.

# Contents

# Chapter 1

# Introduction

Networks are no longer simply about physical networks such as road, pipe or electricity networks, nowadays networks are used in a variety of fields. Networks can be found in everything from models of the spread of diseases to the interaction between genes in our bodies to networks of linguistic relations between words to the Internet.

Many real world networks have a lot in common, for example they are often very large, from hundreds or thousands of vertices in linguistic networks, to tens of thousands in gene-interaction networks to hundreds of thousands in some road or electricity networks to an estimate of more than 40 billion web pages on the indexed Internet [9, 8]. Fortunately they are also typically extremely sparse, most words are only synonyms to a couple other networks in a network over synonym relations, most web pages only link to a couple other web pages, most road intersections are between just a couple of roads and so on.

This thesis is dedicated to the study of networks in three different applications

1. PageRank [10] for ranking of home pages used mainly in search engines and approaches towards improving the calculation speed of PageRank for changing networks.

2. Improving upon current methods used in text annotation and the creation of networks describing linguistic relations between words in specialized domains.

3. The identification of cancer genes using centrality based algorithms on biological networks.

**This thesis is organised as follows:** In this chapter we will give a short summary of some of the mathematical background to the main topics of this thesis. In particular we will start by giving a summary of some necessary and well known definitions and results from graph theory, random walks and matrix theory. We will also give a short introduction to the three main topics of the thesis: 1) PageRank, definitions and methods of calculation 2) Applications of graphs in natural language processing and 3) Applications to biological networks in cancer research.

In Chapter 2 which is based on Papers **A**, **B** and **C** we will take a deeper look at how PageRank can be calculated analytically for a couple of common types of graphs. In particular two different methods will be used, one based purely on a probabilistic perspective and another from a linear algebra point of view. We also give some example of how these results can be applied on graphs composed of multiple graphs of certain types as well as showing some of the limitations of the methods.

In Chapter 3 which is based on Papers **D** and **E** we will look at how graph partitioning can be used to calculate PageRank more efficiently. In particular we will consider the well know partition of a graph into strongly connected components as well as one other type of partitioning of the graph and how this can be used to divide the problem into multiple smaller subproblems. We will also take a look at how different types of graph components can be handled differently to speed up calculations. To finish this chapter we also give some numerical results.

In Chapter 4 which is based on Papers **F** and **G** we will consider a variation of the problem considered in Chapter 3, namely how to update PageRank efficiently after parts of the graph change. In particular we will look at specific types of changes to the graph or graph components and how the old PageRank values can be used when calculating the new updated values. We will also give a framework for how these methods could in theory be applied using the framework introduced in Chapter 3.

The remaining part of this thesis is dedicated to ranking and clustering problems in Natural language processing (NLP) (Papers **H**, **I** and **J**) and of gene-interaction networks in biology (Papers **K** and **L**).

Specifically in Chapter 5 we will compare different clustering methods

together with different distance or similarity measures and how well they compare to manually created clusters of Pharmacovigilance terms (terms related to adverse drug effects). In order to calculate the distance or similarity between different words we make use of the network of semantic relations between terms supplied by the semantic resource ontoEIM [1], which is then later used for clustering and evaluation. Chapter 5 is based on Paper **H**.

Chapter 6 which is based on Paper **I** concerns the use of a PageRank derived method for the filtering and identification of synonymy relations between words. Specifically we apply PageRank to cluster the initial network of proposed relation between words into clusters of synonyms. Using our filtering method we show that it gives a roughly $0.1 - 0.15$ increase in precision compared to the unfiltered relations on the Gene Ontology (GO) corpus.

Chapter 7, the last chapter related to applications in NLP takes a step back and considers the problem of annotation of important terms in a text corpus, part of the creation of the type of terminologies exploited in Chapter 5-6. The chapter concerns the filtering of candidate terms given by a term extractor such as the one implemented in YATEA using a parametrization of the C-Value. By using different levels of parametrization optimised using a genetic algorithm we show that it is possible to improve R-precision and average precision on two of three corpora related to biology by roughly 10%. Chapter 7 is based on Paper **J**.

The last two chapters concerns the exploitation of centrality measures for the identification of cancer genes in biological networks. Chapter 8, based on Paper **K** gives a review over different types of graph centrality and how they can be applied to biological networks.

In Chapter 9 which is based on Paper **L** we look at how well centrality of vertices in the true network is preserved in networks generated from experimental data. Specifically we consider gene-interaction networks created by microarray data and how well the methods that create networks from such data preserve the centrality of the real network.

## 1.1 Preliminaries

We start this thesis with a short introduction to some of the elementary definitions and concepts in graph theory, matrix theory and their relation to random walks and Markov chains which will be used extensively throughout this work.

### 1.1.1 Definitions from graph theory

This section is partly based on [36], although it would be reasonable to assume that almost any textbook on graph theory would cover these topics. This section mainly serve to highlight what material the reader is assumed to already be somewhat familiar with and to resolve any questions about notation or definitions.

Many problems can be formulated using graphs, everything from actual real world networks such as road networks, pipe networks or the network of links between home pages to graph based models such as gene interaction networks, networks of synonym relations between words or modeling of social networks. But graphs are also used in many situations of which there is no intuitive "real" network, for example using a breadth first search in the Cuthil-Mckee algorithm [12] to transform a sparse matrix to a band matrix in order to reduce storage space required, or using a depth first search in Tarjan's algorithm [32] to find all "strongly connected components" (see Definition 4) of corresponding graph in order to transform a matrix into block triangular form.

**Definition 1** (Basic definitions of graphs)**.** A graph is defined as a set of elements $V$ called *vertices* and a set of pairs of vertices $E$ called *edges* representing relations between vertices. A single edge between $v_a$ and $v_b$ we write as $(v_a, v_b)$.

- *Directed graph*: A graph is called a *directed graph* if the edge set is an ordered set such that $(v_a, v_b)$ represents an edge starting in vertex $v_a$ and ending in vertex $v_b$.
- *Loop*: A *loop* is an edge with the same source and target vertex $(v_a, v_a)$.
- *Simple graph*: A graph is called *simple* if there exist no pair of edges with the same source and target vertices and neither does it contain any loops.
- *Weighted graph*: A graph is *weighted* if for every edge we assign a positive value called a weight.

Often graph and network will sometimes be used to mean the same thing, similarly we may sometimes talk about nodes instead of vertices or links instead of edges. We are mainly interested in simple directed graphs (sometimes allowing loops sometimes not). Unless it is stated otherwise all graphs

can be assumed to be simple in that they contain no pair of edges with the same source and target vertices. A graph is typically drawn by representing vertices as dots or circles and edges as lines or in the case of directed graphs arrows between pairs of vertices. Any weights on edges are written close to corresponding edge in the image. An example of a weighted directed graph with six vertices and seven edges can be seen in Figure 1 below.

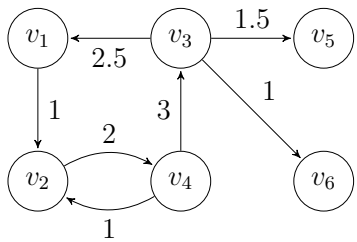


**Figure 1:** An Example of weighted graph with 6 vertices and 7 edges.

One of the most common ways to represent a graph is using an Adjacency matrix, this is also the representation that will be used throughout this whole thesis.

**Definition 2.** Given a directed graph $G := (V, E)$ with $|V|$ vertices and $|E|$ edges, the *adjacency matrix* $\mathbf{A}$ of $G$ is the $|V| \times |V|$ matrix with elements:

$$A_{ij} = \begin{cases} 1 \text{ if } (v_i, v_j) \in E \ , \\ 0 \text{ otherwise } . \end{cases}$$

To get the adjacency matrix of an undirected graph consider a directed graph where every edge in the undirected graph is represented by an edge in both directions in the directed graph. If the graph is weighted it is useful to replace the ones in the adjacency matrix with corresponding weights on the edges in this case we instead call it the *distance matrix* or *modified adjacency matrix* if the weights are decided by some rules depending on the graph rather than arbitrarily.

The adjacency matrix is especially useful when working with weighted graphs such as transition matrices in Markov chain theory. An example of

the distance matrix **D** (adjacency matrix modified for edge weights) of the graph in Figure 1 can be seen below.

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 2.5 & 0 & 0 & 0 & 1.5 & 1 \\ 0 & 1 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

One of the most useful thing about the adjacency matrix is that if **A** tells if two vertices have a vertex between them (path of length 1), then the elements of successive powers of **A** describes whether there is a path between corresponding vertices of successive longer lengths. For example $A^4_{ij} \neq 0$ if and only if there is a path of length four starting in $v_i$ and ending in $v_j$. While this can be used for example to determine if the graph is strongly connected (for each pair $i, j$ is there a positive integer $k$ such that $A^k_{ij} > 0$?), find the length of shortest paths between vertices (the smallest $k$ for which $\mathbf{A}^k_{ij} \neq 0$) and so on, search based algorithms such as depth first search (DFS) are usually much more computationally effective.

Most real world networks being extremely sparse, for example most intersections in a road network are only between $2-3$ roads, most home pages only link to a couple of other home pages, most genes don't interact with each other directly, and so on. Because of this a sparse representation of the adjacency matrix is often required but this usually does not in any way change how we think about it. It does however change what we can do with it, for example actually calculating powers of the adjacency matrix of a large network is usually not possible since this would quickly destroy the sparsity of the matrix, similarly inverting the matrix or calculating most matrix factorizations is not feasible. Instead most methods need to be based on search based algorithms (such as depth or breadth first search) or matrix times vector multiplications which does not change the matrix itself we are working with.

It is easy to envision a person walking between the vertices of a graph by traversing the edges of the graph. This leads to the concept of paths which is essential both if we want to simulate a randomised such kind of walk on the graph (see Subsection 1.1.3), something which is very useful in a huge number of applications such as iteration of a Markov chain where each state in the Markov chain is represented by one vertex in the random

walk and edges represent possible transitions with positive probabilities to move between the different states. Determining if there is any way for such a person to move from one vertex to another (using any number of steps) is also of importance. For example if each intersection on a map is represented by a vertex and edges represents roads between intersections. Then finding that there is no way to traverse the edges to get from one vertex $v_a$ to another vertex $v_b$ would mean that if we wanted to take a trip from the intersection represented by $v_a$ to the intersection represented by $v_b$ we would not be able to take the car the whole way but would need to use some alternative method of transportation such as taking the ferry part of the way.

**Definition 3** (Paths, cycles and connectivity)**.** Given a simple graph $G := (V, E)$, a *path* is a sequence of vertices $v_1, v_2, v_3, \ldots, v_n$ such that between any consecutive vertices $v_i, v_{i+1}$ in the sequence there is a $(v_i, v_{i+1})$ edge. A path in a directed graph where the order of the vertices in the $(v_i, v_{i+1})$ edge is important is also called a *directed path*.

- *Cycle*: A *cycle* is a path where the first and last vertex in the sequence is the same $v_1 = v_n$. If it is a directed path we also call it a *directed cycle*.

- *Strongly connected graph*: A directed graph is called *strongly connected* if for any pair of vertices $v_a, v_b$ there exist a path starting in $v_a$ and ending in $v_b$ and a path starting in $v_b$ and ending in $v_a$, in other words there exist a cycle containing both $v_a$ and $v_b$. Two vertices for which there is no path in either direction between them is called *disconnected*, similarly if for two subgraphs $G_a, G_b$ there exist no path between any vertex $v_a \in G_a$ to any vertex $v_b \in G_b$ the subgraphs are called *disconnected*.

- *Directed acyclic graph* (DAG): A directed graph $G$ is called a *directed acyclic graph* if $G$ contains no *directed cycles*.

The connectivity of the adjacency or related distance matrix is often very important, this is the same as saying that the adjacency matrix is irreducible. This is important if you for example want to want to simulate a Markov chain using Markov chain Monte Carlo, want to know which vertices can effect the rank of other vertices in PageRank [10, 6] or knowing which centrality measures you can or cannot use. A directed acyclic graph is interesting since they are usually much easier to work with than a generic graph, by

ordering the vertices appropriately the adjacency matrix of these graphs can be written using a triangular matrix which significantly improves the speed at which you can for example solve a linear system of equations involving $\mathbf{A}$.

Even if the graph itself is neither strongly connected nor acyclic, we can still talk about parts of the graph that are strongly connected when considered by itself, this leads to the definition of strongly connected components.

**Definition 4.** Given a directed graph $G := (V, E)$ and a subgraph $S \subseteq G$, then $S$ is called a *strongly connected component* (SCC) of $G$ if for any pair of vertices $v_a, v_b \in S$ there is a path from $v_a$ to $v_b$ and from $v_b$ to $v_a$. In addition the subgraph $S$ is maximal in the sense that adding any other set of vertices or edges from $G$ to $S$ would break this property.

Due to the maximal property that no other vertex or edge can be added to a SCC it is easy to show that each vertex $v_a$ can only be part of a single SCC since if it was part of two different SCCs then these would not be proper SCCs since the union of the two would itself be s SCC (since all vertices in both have a path to/from $v_a$). Using a similar argument it is also easy to see that the partition of a graph into strongly connected components is unique and forms a partial order (there can be paths from one SCC $S_a$ to another SCC $S_b$ but if there is there cannot be any in the other direction since then $S_a \cup S_B$ would be a SCC and both $S_a$ and $S_b$ would break the maximal property.

Finding a SCC partitioning and corresponding partial order is equivalent to finding a relabeling of the vertices such that the adjacency matrix is written in Block triangular form where each diagonal block represents a SCC and non-diagonal blocks represents edges between components. This can sometimes be useful in order to divide a problem into smaller parts by considering each SCC in sequence rather than the graph as a whole, we will look more at this in Chapter 3.

Two types of vertices will prove to be of special interest in this work namely vertices with either no incoming edges or no outgoing edges. These vertices are obviously not part of any cycle and therefor form their own SCC, but in addition to that they often need to be handled separately from other vertices. For example vertices with no outgoing edges need to be handled separately when calculating PageRank for ranking of home pages or the constructed matrix will not be stochastic. Similarly, C-Value for finding annotated terms in a text in Natural Language Processing needs to be defined

separately for vertices with no incoming edges (see Chapter 7). Because of this it is useful to give these types of vertices a name of their own.

**Definition 5.** Consider a directed graph $G := (V, E)$. A vertex $v$ is called a *dangling vertex* if there is no edge from $v$ to any other vertex. Similarly if there is no incoming edges, $v$ is called a *root vertex*.

The term "dangling vertex" comes from the term "dangling page" which is a reference to a home page with no outgoing links on the web while "root vertex" is a reference to the root of a tree graph.

### 1.1.2 Perron–Frobenius theorem for nonnegative matrices

Before looking at Markov chains and random walks on graphs we will take a short look at an important important theorem from linear algebra related to this subject, namely Perron-Frobenius theorem for nonnegative irreducible matrices. This section is not intended to give a comprehensive overview of the subject, merely introduce some of the definitions and most important results relevant for this thesis. Parts of this section is based on [26], but there are of course many other textbooks covering these subjects [4, 5, 22, 31].

A matrix is nonnegative if it has all non-negative elements. The class of nonnegative matrices plays an important role in many applications, the stochastic matrices are probably the most obvious example, but it is also worth to note that all adjacency matrices as well as the distance matrix of many weighted graphs are also nonnegative for example if edge weights represent a distance, capacity or some (positive) cost.

Let us start by defining what we mean by a irreducible matrix.

**Definition 6.** A square nonnegative matrix $\mathbf{A}$ is said to be *irreducible* if for every element $A_{ij}$ of $\mathbf{A}$ there exist a positive integer $k \equiv k(i, j)$ such that $(\mathbf{A}^k)_{ij} > 0$. A matrix that is not *irreducible* is said to be *reducible*.

Irreducibility or matrices is closely related to strongly connected graphs. If we replace every non-zero element of $\mathbf{A}$ with a 1, then $\mathbf{A}$ will be an adjacency matrix of some graph. For this graph to be strongly connected we require the existence of a path between all pairs of vertices $v_i, v_j$, this is the same as requiring that $A^k_{ij} > 0$ for some $k$.

Another common equivalent definition of irreducibility valid for all square matrices is that a square matrix is called irreducible if there exist no permutation matrix $\mathbf{P}$ such that

$$\mathbf{P}^\top \mathbf{A}\mathbf{P} = \begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{0} & \mathbf{Z} \end{bmatrix},$$

where $\mathbf{X}$ and $\mathbf{Z}$ are both square matrices and $\mathbf{0}$ is a matrix with all zeros.

Much is know about nonnegative irreducible matrices, much of which can be credited to Perron–Frobenius theorem for nonnegative irreducible matrices which says a lot about the eigenvalues and eigenvectors of the matrix..

**Theorem 1.1.1** (Perron–Frobenius theorem for nonnegative irreducible matrices [26])**.** *If $\mathbf{A}$ is a square non-negative irreducible matrix with spectral radius $r = \rho(\mathbf{A})$, then each of the following is true.*

1. *$r = \rho(\mathbf{A})$ is an eigenvalue of $\mathbf{A}$.*
2. *The algebraic multiplicity of $r$ is equal to one (alg $mult_{\mathbf{A}}(r) = 1$).*
3. *There exist an eigenvector $\vec{x}$ such that $\mathbf{A}\vec{x} = r\vec{x}$.*
4. *The unique vector defined by*

   $$\mathbf{A}\vec{p} = r\vec{p}, \quad \vec{p} > 0, \text{ and } ||\vec{p}||_1 = 1$$

   *is called the* Perron vector *of $\mathbf{A}$ There are no nonnegative eigenvectors for $\mathbf{A}$ except for positive multiples of $\vec{p}$, regardless of the eigenvalue.*
5. *$r = \max_{\vec{x} \in N} f(\vec{x})$ (the Collatz-Wielandt formula)*

   $$\text{where } f(\vec{x}) = \min_{\substack{1 \le i \le n \\ x_i \neq 0}} \frac{[\mathbf{A}\vec{x}]_i}{x_i} \text{ and } N = \{\vec{x} | \vec{x} \geq 0 \text{ with } x \neq 0\}$$

For a proof of this beautiful theorem we refer to [26], instead we will consider some of the implications of the theorem for when $\mathbf{A}$ is a distance or adjacency matrix of a graph or the transition matrix of a Markov chain. For example if $\mathbf{T}$ is an irreducible stochastic matrix (nonnegative with all row sums equal to 1) and $\vec{e}$ is the vector of all 1's, then $\mathbf{T}\vec{e} = \vec{e}$ so $(1, \vec{e})$ is an eigenpair of $\mathbf{T}$. However by the Perron-Frobenius theorem there is only one

nonnegative eigenvector, so $\rho(\mathbf{A}) = 1$ and $\vec{e}$ is the Perron vector. Similarly if we instead consider $\mathbf{T}^\top$ (reminding ourselves that a matrix and its transpose have the same eigenvalues), then the Perron vector of $\mathbf{T}^\top$ is the stationary distribution vector of this Markov chain. We will look more on this in the next section.

Initially the theorem was stated for positive matrices only and then later generalised for irreducible nonnegative matrices. For positive matrices we have the additional property that there exist only one eigenvalue on the spectral radius. This is used to divide the nonnegative irreducible matrices into two different classes.

**Definition 7.** A nonnegative irreducible matrix $\mathbf{A}$ having only one eigenvalue, $r = \rho(\mathbf{A})$, on its spectral radius is said to be a *primitive matrix*.

The existence of one or several eigenvalues on the spectral radius is important for the existence and structure of limits of powers of the matrix.

**Theorem 1.1.2** ([26])**.** *A nonnegative irreducible matrix* $\mathbf{A}$ *with* $r = \rho(\mathbf{A})$ *is primitive if and only if* $\lim_{k\to\infty} (\mathbf{A}/r)^k$ *exists, in which case*

$$\lim_{k\to\infty} \left(\frac{\mathbf{A}}{r}\right)^k = \frac{\vec{p}\vec{q}^\top}{\vec{q}^\top\vec{p}} > \mathbf{0}$$

*where* $\vec{p}$ *and* $\vec{q}$ *are the respective Perron vectors for* $\mathbf{A}$ *and* $\mathbf{A}^\top$ *respectively.*

For a complete proof we refer to [26]. The idea behind the proof is quite simple however: from the Perron-Frobenius theorem it is obvious that $1 = \rho(\mathbf{A}/r)$ is a simple eigenvalue for $\mathbf{A}/\mathbf{r}$ and $\mathbf{A}$ is primitive if and only if $\mathbf{A}/r$ is primitive as well. In order to show the existence of the limit it is therefor enough to show that the limit $\lim_{k\to\infty} (\mathbf{M})^k$ exist if $\rho(\mathbf{M}) = 1$ and $\lambda = 1$ is the only eigenvalue on the unit circle and $\lambda = 1$ is simple.

### 1.1.3 Markov chains and random walks on graphs

The theory of random walks on graphs and the very related topic of Markov chains is a widely studied subject. The goal of this section is not to give a comprehensive overview of the subject, but merely giving some elementary definitions and results related to random walks on finite graphs which will be relevant for this thesis. For a more in depth overview over random walks on

graphs I can recommend the work by Peter G. Doyle, J. Laurie Snell [14], for an introductory book in Markov chains, or more generally probability theory, there is a large amount of alternatives [15, 28, 31]. We begin by defining what we mean by a random walk.

A random walk on a graph can be described as a process which starts in one vertex and then successively moves from one vertex to another by randomly following the edges in the graph. In our work we are interested in two types of random walks on graphs, first the standard random walk on a graph where in every step of the random walk we move to a new vertex following the edges of the graph by picking any of the outgoing edges from the current vertex with equal probability, which is the reciprocal of the number of such edges. Alternatively if the graph is weighted with positive weights such that for every vertex the sum of the weights of all outgoing edges from that vertex equal 1, then these weights can be used directly, i.e. the weights could serve as transition probabilities of a Markov chain.

The other type of random walk we are interested in is one where the random walk can stop at a random step, either by reaching a dangling vertex or with some probability for non-dangling vertices.

**Definition 8.** Given a directed graph $G := (V, E)$ with $|V|$ vertices and $|E|$ edges then a random walk on $G$ is defined as a process which starts in some vertex $v_0$ and then moves on the graph by moving to a new vertex in every step following the edges of the graph. More specifically: if we are at vertex $v_t$ at step $t$, in the next step $t + 1$ we move to any of the vertices linked to by any of the edges starting in $v_t$ with equal probability $1/n(v_t)$ where $n(v_t)$ is the number of outgoing edges from $v_t$.

- If the graph is weighted with positive weights where $w(v_i, v_j)$ denote the weight on the $(v_i, v_j)$ edge. For a vertex $v_i$ with at least one outgoing edge we calculate the probability to move from one vertex to another as follows
$$\mathsf{P}(v_{t+1} = v_j | v_t = v_i) = \frac{w(v_i, v_j)}{\sum_{k=1}^{|V|} w(v_i, v_k)} , \tag{1.1}$$
where $w(v_i, v_k) = 0$ if there exist no $(v_i, v_k)$ edge. If vertex $v_i$ is a dangling vertex then $\mathsf{P}(v_{t+1} = v_j | v_t = v_i) = 0$ for all $j$. Alternatively the weights can be used directly as the step-probability when appropriate.

- If the graph contains any dangling vertices, then the random walk is assumed to stop as soon as it tries to move away from said dangling

vertex. Similarly if the weights are used directly but the sum of probability when moving from a vertex is less then one, remaining probability represent a probability to stop the random walk.

The first type of random walk which contain no dangling vertices or other probability to stop the random walk can also be described as an ordinary homogeneous Markov chain where the elements $T_{ij}$ of the transition matrix $\mathbf{T}$ are determined by $T_{ij} = \mathsf{P}(v_{t+1} = v_j | v_t = v_i)$ calculated as the probability in eq. (1.1). Obviously any transition matrix can be used as long as it is a stochastic matrix.

All Markov chains are assumed to be homogeneous (can be represented by a single stochastic matrix) and we classify them depending on the type of transition matrix. For example this means that if the transition matrix is *irreducible* or *primitive*, we also say that the Markov chain is *irreducible* or *primitive* respectively.

While the alternative random walk with some probability to stop the random walk is not a Markov chain, it can easily be modeled using a Markov chain by adding one vertex to the graph with only a single loop. By adding a link to this new vertex from any dangling vertices with weight one as well as from any vertices with some probability to stop the random walk with weight equal to this probability we ensure that the resulting matrix is stochastic and hence describes a Markov chain. Obviously these two random walks are the same until we hit the new vertex, hence we can use this Markov chain with one more absorbing state to model the random walk on a graph with some probability to stop the random walk in some or all states.

Given a random walk on a graph, there are many natural questions to ask such as: If I let the random walk go on for a long time, does the probability to be in some vertex converge to some value? If I start a random walk in vertex $v_i$ what is the probability that I ever end up in state $v_j$? What is the expected number of times I will visit state $v_j$ if I let the random walk go on forever?

We remember from our discussion of the adjacency matrix that $\mathbf{A}^k_{ij} \neq 0$ if there is a path from $v_i$ to $v_j$ of length $k$, similarly it is easy to see that an element of some power of the transition matrix $\mathbf{T}^k_{ij}$ is the probability to be in state $v_j$ after $k$ steps if starting in state $v_i$ in a random walk described by transition matrix $\mathbf{T}$. This leads us to the concept of stationary distributions.

**Definition 9.** Let $\mathbf{T}$ be the transition matrix of an Markov chain with finite number of states, then a nonnegative vector $\vec{\pi}$ with sum 1 (a probability

distribution vector) is said to be a *stationary distribution* of this Markov chain if $\vec{\pi} = \mathbf{T}^\top \vec{\pi}$.

Using Perron-Frobenius theorem (Theorem 1.1.1) it is easy to see that the Perron vector belonging to a transposed irreducible transition matrix $\mathbf{T}^\top$ is a stationary distribution of corresponding Markov chain. The same theorem also proves that for Markov chains with an irreducible transition matrix this distribution is unique.

Markov chains described by primitive transition matrices are especially useful since they make the calculation of the stationary distribution easy. It is useful to consider the stationary distribution as the average time spent in each state after iterating the Markov chain for a long time.

**Theorem 1.1.3** (Ergodic theorem for primitive Markov chains [31])**.** *As $k \to \infty$, for a primitive Markov chain*

$$\mathbf{T}^k \to \vec{e}\vec{\pi}^\top$$

*element wise where $\vec{\pi}$ is the unique stationary distribution of the Markov chain and $\vec{e}$ is the one vector (each element equal 1). The rate of approach to the limit is geometric.*

For a complete proof of the theorem we refer to [31], however it is worth to point out that most of the proof apart from the convergence rate follows immediately from Perron-Frobenius theorem for nonnegative matrices (Theorem 1.1.1) and Theorem 1.1.2.

Computationally it is usually more convenient to instead calculate the stationary distribution using repeated matrix times vector multiplications instead. Transposing and multiplying both sides with $\vec{\pi}$ from the right gives $\lim_{k\to\infty} (\mathbf{T}^\top)^k \vec{\pi} = \vec{\pi}\vec{e}^\top\vec{\pi} = \vec{\pi}1 = \vec{\pi}$. Hence for a primitive transition matrix we can calculate the stationary distribution numerically by repeatedly calculating $\vec{x}_k = \mathbf{T}^\top \vec{x}_{k-1}$ given some initial probability distribution $\vec{x}_0$.

There is one other topic related to random walks that is central to this thesis, namely the concept of hitting probability or expected number of visits of a vertex in a random walk with some stopping chance.

**Definition 10.** Given a directed graph $G := (V, E)$ with $|V|$ vertices and $|E|$ edges and a random walk on $G$ as defined in Definition 8 with some probability to stop the random walk in some or all vertices then:

- $P_{ab}$ is the probability that the random walk starting in vertex $v_a$ visits $v_b$ at least once. We do not count the starting vertex as a visit, hence $P_{aa}$ is the probability that the random walk starting in vertex $v_a$ returns to $v_a$ at least once.
- $P_{ab}$ is called the *hitting probability* of $v_b$ given that we start in $v_a$.

Obviously given a strongly connected graph with $|V| < \infty$ vertices and no stopping probability, or an irreducible Markov chain with a finite state space then $P_{ab} = 1$ for all $a, b$ so this is mainly interesting if we have either a stopping probability in the random walk or if it is not irreducible (meaning we should eventually get trapped in some state or group of states).

Note that given $P_{ab}$ and $P_{bb}$ it is also easy to calculate the expected number of visits to $v_b$ simply as the probability of at least one visit times the geometric sum over the return probability:

$$E(\#\text{visits to } v_b | \text{starting in } v_a) = P_{ab} \sum_{k=0}^{\infty} (P_{bb})^k \ .$$

## 1.2 PageRank

The PageRank algorithm was initially developed by S. Brinn and L. Page to rank home pages on the Internet [10] for the creation of the search giant Google. The thought behind the method is that home pages that are linked to by many other home pages are in some way better or more important and should therefor be further up in the list of home pages displayed after doing a web search. While this could be achieved by simply counting the number of incoming links to a page, this leads to problems with link farms or other types of spam, empty or low quality pages whose main purpose is to link to another page to promote that. In order to avoid some of these problems they choose to not only rely on the number of incoming links, but also the quality of the pages that give these links, for example it is obvious that a link from a unknown private blog should count for less than say a link from one of the governments home pages.

The number of home pages on the Internet is huge, for example searching for the word *the* in Google gives today 2016-10-02 over 25 billion hits and this doesn't even include all of the indexed part used by Google.

In addition to the dataset being very large, to make matters worse we need the method to return very accurate results. Anything that isn't ranked

among the first couple of pages of results will most likely be ignored by the user [10] with the user instead opting to change their search query if they can't find what they are looking for.

Fortunately the original method itself is already very efficient, using a sparse matrix representation it is linear in the number of links (edges), however there is still is a demand for even faster methods if it is to be able to cope with the problem in the future. PageRank is traditionally computed using some variation of the Power Method or Jacobi method (see Subsection 1.2.2) and using these methods many of the properties of PageRank is well known. We will consider some ways to improve the calculation speed of PageRank in Chapter 3 and further how to update PageRank as the network changes in Chapter 4.

For example it can be shown that the convergence speed of the method depends on a scalar parameter $c$ which is used in the definition [19]. In particular the closer this parameter is to 1 the slower the method converges. The same parameter is also important for the problem to be well conditioned, again a $c$ value close to 1 is problematic since that makes the problem ill conditioned [20].

The mathematics behind PageRank itself is built on the theory non-negative matrices and Perron-Frobenius theorem for non-negative irreducible matrices in particular [5, 16, 22] and the theory of Markov chains [28, 30]. However determining how PageRank changes when changing either some off the parameters or the graph is not as well known.

While PageRank is mainly used in search engines, similar methods have been applied in a number of other applications such as the EigenTrust algorithm for reputation management in peer-to-peer networks such as the torrent protocol where it is used to decrease the distribution of unauthentic files [21]. Another similar method to PageRank is DebtRank which have been used for the evaluating of default risk in financial networks[3]. Some work have also been made using PageRank to learn more about the Web itself for example by looking at the distribution of PageRank theoretically as well as experimentally [13].

For a good overview of over the PageRank algorithm and the application to Web search engines I can recommend the book by Langville and Meyer [23].

### 1.2.1 Definitions and notation

PageRank can be defined in a large number of equivalent ways, in fact we will give three different definitions of PageRank which all give the same ranking. The first two are PageRank as originally defined either as the eigenvector to the dominant eigenvalue of a modified version of the adjacency matrix or as the stationary distribution of a Markov chain. The third definition, which is the one we will mainly use is slightly different since the rank vector is not normalized, is defined similarly using random walks on a graph but now as the expected number of visits to a vertex from multiple random walks with a stopping probability. We will end this section by showing that the three different definitions do in fact give the same ranking.

S. Brin and L. Page originally defined PageRank as the eigenvector to the dominant eigenvalue of a modified version of the adjacency matrix of a graph [10] as defined in Definition 11.

**Definition 11.** PageRank $\vec{R}$ for the vertices in a graph $G := (V; E)$ is defined as the (right) eigenvector with eigenvalue one to the matrix:

$$\mathbf{M} = c(\mathbf{A} + \vec{g}\vec{w}^\top)^\top + (1-c)\vec{w}\vec{e}^\top \ , \tag{1.2}$$

where $\mathbf{A}$ is the adjacency matrix weighted such that the sum over every non-zero row is equal to one, $\vec{g}$ is a vector with zeros for vertices with outgoing edges and 1 for all vertices with no outgoing edges, $\vec{w}$ is a non-negative vector with norm $||\vec{w}||_1 = 1$, $\vec{e}$ is a one-vector and $0 < c < 1$ is a scalar. All vectors are of length $n$ and all matrices of size $n \times n$.

By weighting the edges of the adjacency matrix such that the sum over every non-zero row is equal to one $\mathbf{A}$ will be a (sub)stochastic matrix. The part $\vec{g}\vec{w}^\top$ can be seen as giving all dangling vertices a set of outgoing edges depending on $\vec{w}$ weighted such that corresponding rows have sum 1. Adding $\vec{g}\vec{w}^\top$ ensures that the matrix $(\mathbf{A} + \vec{g}\vec{w}^\top)^\top$ will be a stochastic matrix and since $\vec{w}\vec{e}^\top$ is also a stochastic matrix this ensures that $\mathbf{M}$ itself will also be stochastic.

Often $\vec{w}$ is chosen to be a positive vector such as the equally weighted vector with elements $w_i = 1/n$ this also ensures that after adding $(1-c)\vec{w}\vec{e}^\top$ then $\mathbf{M}$ will not only be a stochastic matrix but it will also be a positive matrix and therefor an irreducible primitive matrix. This ensures that this eigenvector exists and is unique from Perron-Frobenius theorem for non-negative irreducible matrices 1.1.1.

Since $\mathbf{M}$ is a stochastic matrix it is also useful to look at PageRank as the stationary distribution of a random walk on the graph described as a Markov chain.

**Definition 12.** PageRank $\vec{R}$ for the vertices in a graph $G := (V; E)$ is defined as the stationary distribution of the Markov chain described by the following random walk on the graph. Consider a random walk on $G$, a probability distribution vector $\vec{w}$ and a constant $0 < c < 1$. If we are at vertex $v_i$ after $t$ steps, in step $t + 1$ we move to a new vertex as follows.

- If $v_i$ has at least one outgoing edge, with probability $c$ we move to a new vertex by traversing any of the outgoing edges from $v_i$ with equal probability, which is the reciprocal of the number of such edges.
- With probability $1 - c$ or if the $v_i$ have no outgoing edges we move to a new vertex according to probability distribution $\vec{w}$ (ignoring the graph).

It is useful to think of PageRank as the probability of a random web-surfer to be at a page after a long time of web-surfing. In every step with probability $c$ the web surfer clicks on any of the links on the current page to get to a new page or with probability $1 - c$ it ignores the current page and instead goes to a new page by writing the address directly.

While the original definition is very elegant and have a nice interpretation, the normalization can be problematic in particular if you want to compare PageRank between different graphs or calculate PageRank for different parts of a graph and then combine the results. In order to avoid these problems we will give a third slightly different definition of PageRank by instead considering multiple random walks on the graph.

**Definition 13.** Consider a random walk on a graph described by $\mathbf{A}$, which is the adjacency matrix weighted such that the sum over every non-zero row is equal to one. In each step with probability $0 < c < 1$, move to a new vertex from the current vertex by traversing a random outgoing edge from the current vertex with probability equal to the weights on corresponding edge weight. With probability $1 - c$ or if the current vertex have no outgoing edges we stop the random walk. Then PageRank $\vec{R}$ for a single vertex $v_j$ can be written as

$$R_j = \left( w_j + \sum_{v_i \in V, v_i \neq v_j} w_i P_{ij} \right) \left( \sum_{k=0}^{\infty} (P_{jj})^k \right) , \tag{1.3}$$

where $P_{ij}$ is the probability to hit vertex $v_j$ in a random walk starting in vertex $v_i$. This can be seen as the expected number of visits to $v_j$ if we do multiple random walks, starting in each vertex once and weighting each of these random walks by $\vec{w}$.

Note that although this definition makes no special treatment of dangling vertices the normalized and non-normalized definitions of PageRank are still proportional to each other and hence give the same rankings. Why this is true will be made clear after looking at how PageRank is calculated.

### 1.2.2 Standard methods to calculate PageRank

PageRank is often calculated on very large sparse networks, it is essential that this sparsity is exploited in order to be able to calculate PageRank effectively. In particular this means that $\mathbf{M}$ cannot be used directly since this is probably a full matrix (few zero elements) even if $\mathbf{A}$ is sparse, it also makes it hard to use most exact methods such as methods based on matrix factorizations.

The most common approach is by using a variant of the Power Method or rewriting the problem as a linear system and solving this using for example the Jacobi method. Both of which can be formulated in such a way that the most expensive step in both is repeated matrix times vector multiplications with the original matrix $\mathbf{A}$. We will also take a short look at using a power series, how this method relates to PageRank as defined in Definition 13 and how it also relates to the ordinary Jacobi method.

We will start by considering ordinary (normalized) PageRank, if we wanted to naively apply the Power method we would use the following iteration formula given some initial guess $\vec{x}$:

$$\vec{x}^{n+1} = \frac{\mathbf{M}\vec{x}^n}{||\mathbf{M}\vec{x}^n||_1} \ .$$

Since $\mathbf{M}$ is a stochastic matrix $||\mathbf{M}\vec{x}^n||_1 = 1$ and can therefor be removed, to get rid of the multiplication with $\mathbf{M}$ and instead use $\mathbf{A}$ we make use of the normalization as follows:

$$\vec{x}^{n+1} = \mathbf{M}\vec{x}^n = \left(c(\mathbf{A} + \vec{g}\vec{w}^\top)^\top + (1-c)\vec{w}\vec{e}^\top\right)\vec{x}^n =$$

$$c\mathbf{A}^\top\vec{x}^n + \left(c\vec{w}\vec{g}^\top + (1-c)\vec{w}\vec{e}^\top\right)\vec{x}^n = c\mathbf{A}^\top\vec{x}^n + d\vec{w} \ ,$$

where $d$ is some unknown constant, the last equality is found by noticing that the right hand side in the previous expression is proportional to $\vec{w}$.

Since we know that $||\mathbf{M}\vec{x}^n||_1 = 1$ then we get $d = 1 - ||c\mathbf{A}\vec{x}^n||_1$ which results in the following iteration formula for the Power method instead:

$$\vec{x}^{n+1} = c\mathbf{A}^\top \vec{x}^n + d\vec{w} \ ,$$

$$d = 1 - ||c\mathbf{A}\vec{x}^n||_1 \ .$$

This is the formulation initially used by S. Brin and L. Page to calculate PageRank [10] using only matrix times vector multiplications with the sparse matrix $\mathbf{A}$ instead of multiplication with the full matrix $\mathbf{M}$. The convergence of the power method can be shown to depend on the difference between the largest and second largest (in absolute value) eigenvalue of $\mathbf{M}$ [19], the large the difference in absolute value the faster the convergence. Under the reasonable assumption that $\mathbf{M}$ is both irreducible and primitive this difference is guaranteed to be non-zero from Perron-Frobenius for non-negative matrices (see Theorem 1.1.1).

Another common method to calculate PageRank is by rewriting the eigenvector problem as a linear system and solving this using the Jacobi method or another iterative method for solving linear systems. Since we know that 1 is an eigenvalue of $\mathbf{M}$ from Perron-Frobenius 1.1.1 we can rewrite PageRank $\vec{R}$ as follows.

$$\begin{aligned}
\vec{R} = \mathbf{M}\vec{R} &= \left(c(\mathbf{A} + \vec{g}\vec{w}^\top)^\top + (1-c)\vec{w}\vec{e}^\top\right)\vec{R} \\
&= c\mathbf{A}^\top\vec{R} + \left((c\vec{g}\vec{w}^\top)^\top + (1-c)\vec{w}\vec{e}^\top\right)\vec{R} \\
\Leftrightarrow (I - c\mathbf{A}^\top)\vec{R} &= \left((c\vec{g}\vec{w}^\top)^\top + (1-c)\vec{w}\vec{e}^\top\right)\vec{R} \propto \vec{w} \ ,
\end{aligned}$$

where $\propto$ denotes proportionality ($\vec{x} \propto \vec{y}$ if $\vec{x} = k\vec{y}$ for some constant $k$). As seen, in order to find $\vec{R}$ we need to solve the linear system $(\mathbf{I} - c\mathbf{A}^\top)\vec{R} = k\vec{w}$, where $k$ is any non-zero scalar. We can make use of the proportionality since we are typically only interested in the ranking rather than the value of the elements in the rank vector themselves. Obviously it is easy to get the normalized PageRank as the stationary distribution of a Markov chain simply by dividing the resulting PageRank vector by the 1-norm of the vector itself.

The Jacobi method is only guaranteed to converge on matrices which are diagonally dominant (absolute value of each diagonal element is larger than

sum of absolute value of all other elements on the same row or column), obviously this is the case since the sum of every column of $c\mathbf{A}^\top$ is either $c$ or zero in the case if a dangling vertex which is both less than the one on the diagonal of the identity matrix. Using the Jacobi method to calculate PageRank gives the following iteration procedure:

$$\vec{x}^{n+1} = \mathbf{D}^{-1}\left(k\vec{w} + c(\mathbf{L}+\mathbf{U})^\top \vec{x}^n\right) \ ,$$

where $\mathbf{D}+\mathbf{L}+\mathbf{U} = (\mathbf{I} - c\mathbf{A}^\top)$ such that $\mathbf{D}$ is the diagonal of $(\mathbf{I} - c\mathbf{A}^\top)$, $\mathbf{L}$ contains all elements below the main diagonal and $\mathbf{U}$ contains all elements above the main diagonal. If we do not allow self loops (non-zero diagonal elements of $\mathbf{A}$) this reduce to:

$$\vec{x}^{n+1} = k\vec{w} + c\mathbf{A}^\top \vec{x}^n \ . \tag{1.4}$$

This last expression should be recognized from before, with $k = 1 - ||c\mathbf{A}\vec{x}^n||_1$ this is exactly the Power method discussed earlier.

If we want to calculate the non-normalized PageRank defined in Definition 13 we will start by considering PageRank as a power series, and later show that in this case it can also be shown to be just a special case of the Jacobi method. To calculate non-normalized rank we want to be able to calculate the expected number of visits to some vertex $v_a$ in a random walk starting in $v_b$ where the random walk have a probability $1-c$ to stop in every step or stops right away if we end up in a dangling vertex.

Let us ignore the probability $1-c$ and instead consider a random walk taking $n$ steps, then the expected number of visits is equal to the sum of the probability that we visit the vertex in each of the $n$ steps. If we let $\mathbf{A}$ be the transition matrix of the Markov chain describing such a random walk, then the probability to be in a vertex after $k$ steps can be described by $(\mathbf{A}^\top)^k\vec{v_b}$ where $v_b$ is the probability distribution with a probability 1 to be in $v_b$ and zero elsewhere. Dangling vertices which represent vertices with no outgoing edges obviously is a problem since the presence of any such vertices means that $\mathbf{A}$ would no longer be stochastic and hence not a transition matrix. Fortunately this can easily be solved by adding a new absorbing state to the Markov chain to which all dangling vertices link.The expectation of the total number of visits is calculated as the sum over all $n$ steps:

$$E(\{\#\text{visits}|\text{n steps}\}) = \sum_{k=0}^{n} (\mathbf{A}^\top)^k \vec{v_b} \ .$$

In our case we want the number of steps to be described by the probability $c$ in such a way that we only take another step with probability $c$. Then the probability to take $k$ steps is the product of these probabilities $c^k$. Replacing the number of iterations $n$ with infinity and instead multiply the probability in each step $k$ with the probability that we take $k$ steps we get

$$E(\{\#\text{visits}|\text{prob. } c^k \text{ to take } k \text{ steps}\}) = \sum_{k=0}^{\infty} c^k (\mathbf{A}^\top)^k \vec{v_b} \ . \tag{1.5}$$

We note that although we added an absorbing state to $\mathbf{A}$ in order to handle dangling vertices, this is actually not required since there obviously is no path from the absorbing state to any other states, i.e. $\mathbf{A}$ may be the original (sub)stochastic matrix with some zero rows.

In order to get the PageRank we need to do one such random walk from each vertex $v_b$ and weight each such random walk by corresponding element in $\vec{w}$, this is easily achieved simply by replacing $v_b$ in eq. (1.5) with $\vec{w}$:

$$\vec{R} = \sum_{k=0}^{\infty} c^k (\mathbf{A}^\top)^k \vec{w} = \sum_{k=0}^{\infty} (c\mathbf{A}^\top)^k \vec{w} \ . \tag{1.6}$$

Such a series converge only if the spectral radius $\rho(c\mathbf{A}) < 1$, something we can be sure of since the sum of each row of $c\mathbf{A}$ is less than $c < 1$ We now have yet another way to calculate PageRank using a power series by using as many terms in the series as we need. Next step is to show that PageRank as defined in Definition 13 which we can calculate using a power series gives the same ranking as PageRank defined in Definition 11.

We already know that this series converge, this means we can consider it as the Neumann series of $(\mathbf{I} - c\mathbf{A}^\top)^{-1}$, this gives

$$\vec{R} = \sum_{k=0}^{\infty} (c\mathbf{A}^\top)^k \vec{w} = (\mathbf{I} - c\mathbf{A}^\top)^{-1} \vec{w} \Leftrightarrow (\mathbf{I} - c\mathbf{A}^\top)\vec{R} = \vec{w} \ . \tag{1.7}$$

This is exactly the equation system we solved using the Jacobi method to calculate PageRank, hence the two definitions are equal.

The main thing to take with us from this is that although Definition 11 and Definition 12 both require $\vec{w}$ to be a non-negative vector with norm $||\vec{w}||_1 = 1$, Definition 13 carry no such requirement, and still give the same ranking. This have two consequences, first of all we are no longer required

to redo our normalization of $\vec{w}$ every time we want to add or remove vertices or change the weight of a singe vertex. Secondly it makes PageRank independent on the overall size of the graph, for example comparing the PageRank between two disconnected graphs using normalized PageRank can be problematic, especially if there are dangling vertices. However comparing PageRank between two disconnected graphs using non-normalized PageRank is straightforward since there obviously is no path between the two graphs they also have zero effect on each others rank and can therefor be compared directly without issue.

We will end this chapter by showing that Calculating PageRank using the power series in eq. (1.6) is actually just a special case of using the Jacobi method. If we write out the first $n$ steps of the power series we end up with:

$$\begin{aligned}\vec{R}^n &= \vec{w} + c\mathbf{A}^\top\vec{w} + (c\mathbf{A}^\top)^2\vec{w} + \ldots + (c\mathbf{A}^\top)^n\vec{w} \\ &= \vec{w} + (c\mathbf{A}^\top)(\vec{w} + c\mathbf{A}^\top\vec{w} + (c\mathbf{A}^\top)^2\vec{w} + \ldots + (c\mathbf{A}^\top)^{n-1}\vec{w}) \\ &= \vec{w} + (c\mathbf{A}^\top)\vec{R}^{n-1} \ .\end{aligned}$$

This should be recognized from before, it is precisely the same expression as the one we got using the Jacobi method for a graph with no dangling vertices in eq. (1.4) with $k = 1$ and initial guess $\vec{x}^0 = \vec{w}$.

To summarize we have shown three different definitions of PageRank which all give the same ranking but with different interpretations. We will mainly work with Definition 13 since it gets rid of the normalization and the interpretation of PageRank as the expected number of visits during a number of random walks is useful for writing proofs. We have also shown three conventional methods for calculating PageRank, 1) using the power method, 2) using the Jacobi method and 3) using a power series. We also showed that at least if there are no self-loops all three methods are essentially the same.

## 1.3 Problems in Natural language processing

Natural language processing (NLP) in short can be described as the study of human language through the use of computers. Some examples include automatic text translation from one language to another, speech recognition or what is usually an earlier step, tagging the words in a text depending on their syntactic role, so called part-of-speech tagging.

A text corpus is simply a collection of texts, for example a collection of abstracts from articles in some research area or a collection of emails. Such a text corpus can be useful to learn a number of things about the texts in the corpus. For example consider a theoretical online version of this thesis, it could be desirable to have an interactive index such that you can write a term and then get to corresponding pages in the thesis. In such a index it would be desirable to give results not only for the exact term searched for but also for other semantically close terms, for example if I search for NLP, we would like my index to show results for Natural language processing as well.

This creates two problems, first in actually extracting these terms, mainly noun phrase, which refer to the specialized knowledge of the field and thus are likely search terms of a user, in addition we are also interested in determining the semantic relation between such terms, for example the terms *vertex*, *node* and *lump* are all sometimes used as synonyms, but obviously there is no such relation between *node* and *lump* in the context of graph theory.

While these kind of terms representing the knowledge in a field is usually recorded in terminologies, such terminologies suffer from a low coverage [7, 25]. This calls for the use of automatic approaches which first extract potential terms from the corpus [11] and then tries to find the true terms from such a list of initial candidates.

In Chapter 7 we consider the problem of filtering out the most likely true terms in a large list of such candidate terms using a combination of statistics such as the frequency of a term in combination with the nestedness of terms. With nestedness we mean terms which are part of another term, for example the term *linear algebra* contains two other terms *linear* and *algebra* all three of which may be important terms in their own right. Obviously all three terms could also be part of other terms of their own, combining all of these terms together creates a directed acyclic graph which can be used together with other statistics to attempt to filter out the "best" candidate terms.

For the detection of similar terms (synonymy, inclusion, etc.) many different approaches have been proposed [24, 18, 27, 17]. However even if there is a semantic relation between words, this relation could be either strong or weak. For example the term *matrix theory* is included as a part of both *algebra* and *linear algebra*, but obviously the relatedness is much stronger between *matrix theory* and *linear algebra*.

In Chapter 5 we use a network of such similarities between terms to cluster terms into groups of related terms in a corpus related to the detection and

prevention of adverse drug reactions. We compare different approaches of computing distance between terms using such a network of relations, for example by exploiting shortest paths [29] together with different clustering methods. The goal is to find which combination of distance and clustering method is most likely to give a good clustering of terms when compared to clusters manually created by experts.

When automatically extracting semantic relations between terms, it happens that two terms might be related in one context and unrelated in another or that the relatedness is different in different contexts. This can depend both on the method used and the documents analysed, for example two terms might be considered synonyms using one method but not another or they might be considered synonyms using one method while another detects an inclusion relation instead.

In Chapter 6 we consider a network of unfiltered relations where every pair of terms (vertices) might have one or several relations such as synonymy, inclusion or is-a relations between them with weights depending on how many times corresponding relation was identified. Using such a network we are interested in finding the true relation between terms while filtering out the incorrectly tagged relations.

## 1.4 Biological networks

How does one gene interact with another? Which proteins bind to each other? Which enzymes interact with each other? What happens to the function of one gene if I knock out another gene? All these are questions related to different kinds of relations between biological entities, something which could be modeled using a network. These kinds of networks are one of the most important tools in systems biology. For example if you know the underlying genes responsible for the development of a cancer (driver genes), you may also be able to develop medication targeting these genes specifically to be used instead or in combination with surgery to treat the cancer.

Given recent technological advancements in high-throughput simultaneous measurements of biological entities such as through the use of microarrays the amount of data that is available to facilitate the creation of such networks is now greater than ever. One example is gene-interaction networks which depict the dependency of genomic variations in causing certain phenotypes by modeling genes as nodes and linking them by an edge if a simultaneous

alteration of both is required to obtain a given biological outcome, such as lethality [33, 34].

By creating a network of gene-interactions for a single disease using patient data and comparing this to the network created from healthy samples, the idea is that it is possible to look at the difference between the two networks in order to locate one or several genes which are important for the development of the disease. Unfortunately these inferred networks are not of high enough quality that these differences are clear because of a large amount of noise.

These networks are often on the order of tens of thousands of genes while the dataset itself might be a couple of hundred patients for a somewhat uncommon disease. A combination of noisy measurement sand so many unknowns and comparatively few observations means that computational methods can usually only hope to find good candidates for genes to then be verified experimentally. A number of such graph based methods have been suggested for the prediction of disease genes from a variety of biological networks [2, 35, 37].

In Chapter 8 we give a review over different types of biological networks and how different types of graph centrality measures have been used to try and predict cancer genes.

In Chapter 9 we take a look at how well such centrality measures are preserved when generating a network from experimental data. In particular we compare a number of different network generation methods and calculate the rank-correlation between the rankings of the true and generated networks. The motivation behind this is that if the centrality in the generated network does not agree well with the centrality in the "true" network, then even if a method can be used to locate driver genes in a true network it might not be useful when considering the generated network.

# References

[1] Alecu, I., Bousquet, C., Jaulent, M.: A case report: using snomed ct for grouping adverse drug reactions terms. BMC Med Inform Decis Mak **8**(1), 4–4 (2008).

[2] Barabasi, A.L., Gulbahce, N., Loscalzo, J.: Network medicine: a network-based approach to human disease. Nat Rev Genet **12**, 56–68

(2011).

[3] Battiston, S., Puliga, M., Kaushik, R., Tasca, P., Calderelli, G.: Deptrank: Too central to fail? financial networks, the fed and systemic risk. Tech. rep. (2012).

[4] Bellman, R.: Introduction to Matrix Analysis. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (1970).

[5] Berman, A., Plemmons, R.: Nonnegative Matrices in the Mathematical Sciences. Academic Press (1979).

[6] Bianchini, M., Gori, M., Scarselli, F.: Inside pagerank. ACM Trans. Internet Technol. **5**(1), 92–128 (2005).

[7] Bodenreider, O., Rindflesch, T.C., Burgun, A.: Unsupervised, corpus-based method for extending a biomedical terminology. In: Workshop on Natural Language Processing in the Biomedical Domain, pp. 53–60 (2002).

[8] van den Bosch, A.: The size of the world wide web (the internet). `http://www.worldwidewebsize.com/`. Accessed 2016-10-21.

[9] van den Bosch, A., Bogers, T., de Kunder, M.: Estimating search engine index size variability: a 9-year longitudinal study. Scientometrics **107**(2), 839–856 (2016).

[10] Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Proceedings of the Seventh International Conference on World Wide Web 7, WWW7, pp. 107–117. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands (1998).

[11] CabrÃ©, M.T., EstopÃ , R., Vivaldi, J.: Automatic term detection: a review of current systems. In: Recent Advances in Computational Terminology. John Benjamins, Amsterdam, Philadelphia (2001).

[12] Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: Proceedings of the 1969 24th National Conference, ACM '69, pp. 157–172. ACM, New York, NY, USA (1969).

[13] Dhyani, D., Bhowmick, S.S., Ng, W.K.: Deriving and verifying statistical distribution of a hyperlink-based web page quality metric. Data Knowl. Eng. **46**(3), 291–315 (2003).

[14] Doyle, P.G., Snell, J.L.: Random Walks and Electric Networks, vol. 22, 1 edn. Mathematical Association of America (1984).

[15] Feller, W.: An Introduction to Probability Theory and its Applications, Vol. 1, 3rd Edition (1968).

[16] Gantmacher, F.: The Theory of Matrices. Gantmacher. Chelsea (1959).

[17] Grabar, N., Zweigenbaum, P.: Lexically-based terminology structuring. In: Terminology, vol. 10, pp. 23–54 (2004).

[18] Hahn, U., Honeck, M., Piotrowsky, M., Schulz, S.: Subword segmentation - leveling out morphological variations for medical document retrieval. In: AMIA (2001).

[19] Haveliwala, T., Kamvar, S.: The second eigenvalue of the google matrix. Technical Report 2003-20, Stanford InfoLab (2003).

[20] Kamvar, S., Haveliwala, T.: The condition number of the pagerank problem. Technical Report 2003-36, Stanford InfoLab (2003).

[21] Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on World Wide Web, WWW '03, pp. 640–651. ACM, New York, NY, USA (2003).

[22] Lancaster, P.: Theory of Matrices. Academic Press (1969).

[23] Langville, A.N., Meyer, C.D.: Google's PageRank and Beyond: The Schience of Search Engine Rankings. Princeton University Press (2006).

[24] Max, A., Bouamor, H., Vilnat, A.: Generalizing sub-sentential paraphrase acquisition across original signal type of text pairs. In: EMNLP, pp. 721–31 (2012).

[25] McCray, A.T., Browne, A.C., Bodenreider, O.: The lexical properties of the gene ontology (GO). In: Proceedings of AMIA 2002, pp. 504–508 (2002).

[26] Meyer, C.D. (ed.): Matrix Analysis and Applied Linear Algebra. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2000).

[27] Nguyen, H., Al-Mubaid, H.: New ontology-based semantic similarity measure for the biomedical domain. In: IEEE Eng Med Biol Proc, pp. 623–8 (2006).

[28] Norris, J.R.: Markov chains. Cambridge University Press, New York (2009).

[29] Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Transactions on systems, man and cybernetics **19**(1), 17–30 (1989).

[30] Rydén, T., Lindgren, G.: Markovprocesser. Univ., Lund (2000).

[31] Seneta, E.: Non-negative Matrices and Markov Chains. Springer Series in Statistics. springer (2006).

[32] Tarjan, R.: Depth first search and linear graph algorithms. SIAM Journal on Computing **1**(2), 146–160 (1972).

[33] Tong, A., Evangelista, M., Parsons, A., Xu, H., Bader, G., Page, N., et al: Systematic genetic analysis with ordered arrays of yeast deletion mutants. Science **294**, 2364–2368 (2001).

[34] Tong, A., Lesage, G., Bader, G., Ding, H., Xu, H., Xin, X., et al: Global mapping of the yeast genetic interaction network. Science **303**, 808–813 (2004).

[35] Wang, X., Gulbahce, N., Yu, H.: Network-based methods for human disease gene prediction. Brief Funct Genomics **10**, 280–293 (2011).

[36] West, D.B.: Introduction to Graph Theory, 2 edn. Prentice Hall (2001).

[37] Wu, X., Li, S.: Cancer gene prediction using a network approach. Ch Crc Math Comp Bio pp. 191–212 (2010).

# Index