Detecting Hierarchical Relationships and Roles from Online Interaction Networks

Mohammad Tareq Jaber

A Dissertation Presented for the Degree of

Doctor of Philosophy

Department of Computer Science & Information Systems

Birkbeck, University of London

UK

November 2015

To my beloved country

Syria. . .

Abstract

In social networks, analysing the explicit interactions among users can help in inferring hierarchical relationships and roles that may be implicit. In this thesis, we focus on two objectives: detecting hierarchical relationships between users and inferring the hierarchical roles of users interacting via the same online communication medium. In both cases, we show that considering the temporal dimension of interaction substantially improves the detection of relationships and roles.

The first focus of this thesis is on the problem of inferring implicit relationships from interactions between users. Based on promising results obtained by standard link-analysis methods such as PageRank and Rooted-PageRank (RPR), we introduce three novel time-based approaches, "Time-F" based on a defined time function, Filter and Refine (FiRe) which is a hybrid approach based on RPR and Time-F, and Time-sensitive Rooted-PageRank (T-RPR) which applies RPR in a way that takes into account the time-dimension of interactions in the process of detecting hierarchical ties.

We experiment on two datasets, the Enron email dataset to infer manager-subordinate relationships from email exchanges, and a scientific publication co-authorship dataset to detect PhD advisor-advisee relationships from paper co-authorships. Our experiments demonstrate that time-based methods perform better in terms of recall. In particular T-RPR turns out to be superior over most recent competitor methods as well as all other approaches we propose.

The second focus of this thesis is examining the online communication behaviour of users working on the same activity in order to identify the different hierarchical roles played by the users. We propose two approaches. In the first approach, supervised learning is used to train different classification algorithms. In the second approach, we address the problem as a sequence classification problem. A novel sequence classification framework is defined that generates time-dependent features

based on frequent patterns at multiple levels of time granularity. Our framework is a flexible technique for sequence classification to be applied in different domains.

We experiment on an educational dataset collected from an asynchronous communication tool used by students to accomplish an underlying group project. Our experimental findings show that the first supervised approach achieves the best mapping of students to their roles when the individual attributes of the students, information about the reply relationships among them as well as quantitative time-based features are considered. Similarly, our multi-granularity pattern-based framework shows competitive performance in detecting the students' roles. Both approaches are significantly better than the baselines considered.

Declaration

This thesis is the result of my own work, except where explicitly acknowledged in the text.

Acknowledgements

By the name of Allah, the most beneficent and the most merciful, to whom is all praises and thanks for the grants I have been blessed within my life.

I take immense pleasure in expressing my sincere and deep sense of gratitude to my outstanding supervisors, Prof. Peter Wood and Dr. Panagiotis Papapetrou. Thanks for your expertise, understanding and patience during all stages of my PhD. Your supervision is invaluable. I must also thank Prof. Sven Helmer for his guidance within the first year. Also, many thanks to Prof. Mark Levene for his insightful comments during the annual progress review meetings.

Special thanks go to my mom Najah, who has suffered to give me happiness and has supported me in her prayer, my dad Marwan, who is my support and my role model. Without their emotional and financial support throughout my studies I wouldn't bring my dream into reality. I will be always indebted to both of you.

Additionally, thanks to all those, at Birkbeck Computer Science department, and the London Knowledge Lab. In particular, Natraj Raman who was always inspiring me. Thanks to Khaled, Muawya and Godfried and all others who created a stimulating and very friendly working environment, making my experience extremely enjoyable.

My deep gratitude to my family for their endless love and encouragement. My sisters Rana and Rasha, my brother Abdulhadi, the sweatiest nephews and nieces Tarek, Ahmad, Aya, Sarah and Zenah. I also extend my thanks to my in-laws, Mohamad, Abdulrazak, Haya, Saleh, Samar, Layla, Omar, Abdulkarem, Baraa and Hanan.

Last but not least, a special thank to my beloved wife. Salma, thank you for your scarifies, patience and support. Your love inspires me in every way. The outcome of my PhD is not only a thesis and some papers but also an awesome little boy. Marwan, you are the joy of our life. I cannot be anything but happy and excited when you are around.

Publications

Publications relating to the thesis:

1. Mohammad Jaber, Peter T. Wood, Panagiotis Papapetrou, and Sven Helmer. "Inferring Offline Hierarchical Ties from Online Social Networks". In Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web (WWW 2014), pp. 1261–1266 (Chapters 3 and 4).

2. Mohammad Jaber, Panagiotis Papapetrou, Sven Helmer, and Peter T. Wood. "Using Time-Sensitive Rooted PageRank to Detect Hierarchical Social Relationships". In Proceedings of the 13th International Symposium on Intelligent Data Analysis (IDA 2014). Vol. 8819, pp. 143–154 (Chapter 5).

3. Mohammad Jaber, Panagiotis Papapetrou, Ana González Marcos, and Peter T. Wood. "Analysing Online Education-based Asynchronous Communication Tools to Detect Students' Roles". In Proceedings of the 7th International Conference on Computer Supported Education (CSEDU 2015), pp. 416–424 (Chapter 6).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A social network is defined as a graph structure that consists of a set of actors ("nodes") and a set of relationships ("ties" or "edges") between the actors [127]. With the rapid development of web-based applications, social networks have moved online. Different types of online social networks have emerged where an online community can be formed between a group of people who can interact with each other by sharing of information, experiences, and perspectives throughout community-oriented websites [130]. Examples of current online social networks include discussion forums, chat rooms, traditional social media, email networks and collaboration networks.

The number of people involved in online social activities is increasing day by day. For example, according to the Office of National Statistics[1], more than 38 million people in the UK (76% of the British population) use the Internet, and about 54% of those use some form of social networking website. This explosion of online life has attracted the attention of researchers over the past decade.

The scope of research done in this field extends over a wide range of areas. From a sociological perspective, some work such as [15], has studied the relationship be-

[1] `www.ons.gov.uk/ons/dcp171778_373584.pdf`

tween online and offline social capital, and analysed the differences between them. However, much research such as ours, has studied online social networks using computer science-based techniques. Mining the huge amount of social capital data can be beneficial in improving the effectiveness of online social websites as well as many other applications such as recommender systems, marketing systems, collaboration systems and search systems.

1.1 Problems addressed by this thesis

In this thesis, we address two problems in the field of social network mining. The first is detecting hierarchical social relationships (covered in chapters 3, 4 and 5), such as detecting manager-subordinate relationships from an email network. The second problem is detecting hierarchical roles (covered in chapters 6 and 7), such as classifying a user as a project manager or a project worker from interaction on a project discussion forum.

1.1.1 Detecting hierarchical social relationships

Interactions between groups of people and the patterns of these interactions are typically affected by the underlying social relationships between the people. In social networks, such social relationships are usually implicit. Nonetheless, analysing patterns of social interactions between the members of a social network can help to detect these implicit social relations. For example, consider a social network where the members declare explicitly some type of social relationship with others, such as x is a colleague of y. Now, suppose that we also have available the communication patterns between x and y, e.g., how often they exchange e-mails in a month. Using this information we may be able to infer additional relationships between these two members, such as x is the manager of y. As another example, in a co-authorship network, explicit relationships exist between co-authors, where authors are linked

to all their co-authors. Analysing these explicit relationships may help detect other interesting social ties such as which pairs of co-authors have a PhD advisor-advisee relationship between them. We call relationships, such as manager-subordinate and advisor-advisee, *hierarchical* relationships.

In the first half of this thesis, we address the problem of detecting implicit hierarchical relationships in a social network by exploiting the interactions between the members of the network. We mainly focus on two key features that play a central role in our problem: (a) the structure of the interaction network, and (b) the evolution of the interactions, or in other words, the "dynamics" of the interactions over time. Our intuition is that actors connected by a hierarchical tie will exhibit different temporal interaction patterns to those who are connected by some other type of tie. Our findings demonstrate that for our problem setting "time matters".

We formulate the problem as follows: given an input graph of an interaction network, where nodes represent actors and edges represent interactions between the actors, we analyse the network and infer an output graph representing a hierarchical relationship network over the same group of actors as in the input graph.

Detecting hierarchical social ties may be beneficial for many reasons and in different application domains. First, they can be used for classifying actors according to their role in an organisation or discovering communities and analysing inter- or intra-community relations. Also, hierarchical ties may be used for detecting influential actors within a social network and studying how they influence the whole network. Additionally, such ties can be used for validating the theory of influence attribution in social science, where it is assumed that information usually flows from "opinion leaders" to "ordinary users" [70]. In certain applications, such as analysing a company's performance, the detected management hierarchy from the interaction networks can be compared with the real organisational chart. This could help to assess and adapt the current structure in order to improve the overall work flow.

1.1.2 Detecting hierarchical roles

In the second half of the thesis, we move from detecting hierarchical ties to inferring hierarchical roles. Hierarchical roles are assigned to people in many situations. One example is in project management, where different roles are assigned to project members.

Promoting learning through collaborative activities is one recent orientation in learning strategies. Working on a project means working in a team, and a project team can be seen as a social group where team members are involved in social interactions with each other, share interests and have the common goal of completing the project. For example, the learning framework presented by Alba-Elías [4], assumed that a PRINCE2™ (Projects IN a Controlled Environment) [91] project has an explicit project management team structure consisting of defined and agreed roles (not jobs) and responsibilities for the people (such as students) involved in the project [91]. This project structure facilitates the students' learning process because it clarifies the differences between the different roles of persons who work together on the same project, but with very different responsibilities.

The overall objective of this research is to examine the relationships between students through their online asynchronous conversations such as discussion posts and blogs. More generally, this work addresses the problem of detecting user roles from their online interactions. In this context, we adopt two approaches:

- **Educational Data Mining (EDM)**: We analyse the capability of Educational Data Mining (EDM) to identify patterns that emerge from the online interactions between students according to their role in a project. Using this approach, we identify the set of most relevant features for building a model using supervised learning. In addition, we show that time-based features, which capture the temporal dimension of the interactions, improve the classification results.

- **Sequence Classification (SC)**: The involvement of each student in a project is captured as a sequence of communication events such as sending and/or reading messages. Since the temporal order of various communication patterns may play a major role in identifying the correct role for each student, we devise a framework for identifying frequent communication patterns at different levels of time granularity.

We focus our study on analysing asynchronous communications because they tend to be better structured and developed than synchronous conversations [44], and they provide project members time to examine and reflect on a topic before they formalize their contribution [49] or provide feedback related to a piece of preformed work. Furthermore, asynchronous communication tools are considered appropriate for virtual project teams, because this way students can participate in the project at their own pace.

The knowledge acquired by our proposed algorithms can help teachers understand how students' roles in a project relate to their communication behaviour. In addition, teachers may assess the performance of students and whether they fulfill the roles to which they were assigned. This can be done by comparing a student's role and his/her observed interaction patterns. Moreover, our sequence classification framework can be applied in different domains within the area of sequence classification. For example, we might use our framework to track a suspicious series of transactional data in a bank to detect money laundering activities.

1.2 Thesis Contribution

The contributions of this thesis can be summarised as follows:

Detecting hierarchical social relationships

- We address the problem of detecting hierarchical social relationships in the extreme case when none of these relationships are known beforehand. We exploit a personalised version of a link-analysis method, namely Rooted-PageRank (RPR) which gives promising results.
- We propose three novel time-based approaches, called (1) Time function (Time-F), (2) Filter-and-Refine (FiRe) and (3) time-sensitive Rooted-PageRank (T-RPR) which take into account how the structure of the interaction network changes over time. In T-RPR, we propose two approaches for aggregating scores from each of the time slots over which T-RPR is run, one based on a simple weighted average and the other based on voting.
- We investigate the performance of all the above methods under different experimental settings using directed, undirected, weighted and unweighted interaction networks. We use the Enron e-mail network between 155 employees to detect 147 manager-subordinate relationships between them. In addition, we evaluate the performance on a co-authorship network between more than 1 million authors to detect 2100 PhD advisor-advisee relationships.
- Our experiments show that our T-RPR, Time-F and FiRe methods achieve considerably better results than the competitor RPR baseline approach. This supports our claim that "time matters". For example, T-RPR, Time-F and FiRe score 0.68, 0.62, 0.6 respectively for recall in detecting advisor-advisee relationships compared to only 0.39 using RPR.

Detecting hierarchical roles

- In the first approach, we propose and build several supervised classification models using different data mining algorithms, to detect roles played by project members who are working in the same project and interacting online. In these models, we investigate several sets of quantities and network-based features to train the classification models. We show that the reply-based, time-based and reading-based features are discriminative in detecting users' roles.
- In the second approach, we address the problem as a sequence classification problem. We propose a novel sequence classification framework that generates features based on frequent patterns at multiple levels of time granularity. Feature selection techniques are applied to identify the most informative features that are then used to construct the classification model. Our framework is flexible, so can be applied in different domains to classify sequences of discrete events.
- We evaluate the performance of these approaches on real data of 194 students interacting online through a project communication tool and playing different roles. Our experiments show that, in the first approach, "reply-based" features and a subset of "time-based" features coinciding with the first weeks of the project, improve mapping students to their roles. This highlights the importance of initial interactions between project members. Moreover, our multi-granularity pattern-based sequence classification framework can achieve competitive performance in detecting the students' roles, scoring in the best case more than 0.9 for F-measure compared to only 0.57 using our baseline similarity-based model.

1.3 Thesis Outline

Chapter 2 reviews background research on all areas related to the problems we address in this thesis. This covers areas such as detecting and predicting social relationships from interaction networks, and analysing and modeling user online behaviour. We also review previous work done on the bibliographic networks, which is one of the applications we study. In addition, as we apply our approach in an educational domain, we highlight related work in that context. The chapter concludes with other related work that does not belong to any of the aforementioned areas.

Chapter 3 introduces the main definitions and formulates the problem of detecting hierarchical social ties from online interaction networks. We also describe the two real datasets we used in the experiments, i.e., the Enron (emails) dataset and the coauthor (bibliographic) dataset. In this chapter, we employ our three baseline methods (PageRank, Degree and Rooted-PageRank) on the explicit interactions between actors in order to infer the implicit hierarchical relationships.

Chapter 4 proposes two novel time-based approaches "Time-F" and "FiRe" which take into account the time-dimension of interactions in the process of detecting hierarchical ties. The chapter furnishes the experimental results obtained using these approaches on Enron and coauthor datasets. A real case study on the Enron dataset is explained at the end of this chapter.

Chapter 5 introduces another novel time-sensitive method, called T-RPR, that captures and exploits the dynamics and evolution of the interaction patterns in the network in order to identify the underlying hierarchical ties. The results using T-RPR in different experimental settings are also discussed. The chapter resumes with the case study covered in chapter 4, presenting the results using T-RPR.

Chapter 6 studies the application of Educational Data Mining to examine the online behaviour of students in order to detect their roles. First, several sets of features used to train the classification models are defined. Second, the process of selecting the relevant features is explained. Finally, the chapter concludes with the experimental results and main findings.

Chapter 7 defines a multi-granularity framework for classifying sequences of discrete events. In this chapter, we first introduce our baseline similarity-based model. Then, we explain the phases of our framework, namely feature generation, feature selection and model construction. The empirical results of applying our framework in the educational domain are presented at the end.

Chapter 8 completes the thesis by providing a summary of the contributions and the conclusions of each chapter. Several future directions are then discussed, including alternative approaches for investigation.

Chapter 2

Related Work

Although the history of social networks is relatively short, it is considered to be one of the most active research area these days [92]. Social networks have attracted researchers from different disciplines and backgrounds in computer science. Currently the interest in social networks covers different types of networks. These include traditional discussion forums, famous social media such as Facebook and Twitter, and collaboration networks such as email networks and bibliographic networks.

Given that this thesis aims to detect hierarchical social relationships and roles in different types of social networks, it is useful to understand the position of this thesis within the intensive efforts carried out in the literature. In this chapter, we highlight different areas that are related to our research. We also compare our work with similar approaches in the literature and explain the differences between our findings and others' output.

2.1 Background and Definitions

Data Mining (DM) is the process of extracting knowledge from data. In this process, the data available is scanned to infer useful, interpretable and hidden information [126]. Although many researchers use DM as a synonym for the ***knowledge***

discovery process (KDP) [25], DM is just one step of the KDP. The KDP describes the entire knowledge extraction process and consists of several steps which are executed in a sequence. These steps include understanding the application domain, data preparation, data analysis (i.e., data mining), evaluation and understanding the generated knowledge.

DM can be performed using a wide range of techniques. In this section, we highlight some of these techniques that are strongly connected to the related work and/or our approaches explained later in this thesis.

2.1.1 Machine Learning

Machine learning represents the ability to learn from the data available in order to extract interesting patterns, and make intelligent decisions based on the detected patterns. For example, we may learn from a set of labelled examples to detect the labels of new (unseen) objects. Machine learning can be applied in several ways as follows:

- ***Supervised Learning (Classification)***: The supervision comes from a training dataset which consists of a set of labelled objects. By learning from the training dataset, the unknown labels of objects in a given testing dataset can be detected. We use this technique to detect the roles of students working in the same project (Chapter 6 and 7). An example of a classification model is a ***Support Vector Machine (SVM)*** [29, 125]. This model represents the objects in the training dataset as points in the space of real vectors. Then, a hyperplane or a set of hyperplanes is used to separate the points into categories according to their labels. A good separation is one that keeps gaps as wide as possible between categories. A new unlabelled example is then mapped to the same space and assigned to the category it falls in.

- ***Unsupervised Learning (Clustering)***: In clustering, the input is not labelled. Based on certain features, clustering groups similar objects in one cluster. However, since the training data are not labelled, clustering cannot give semantic meaning to obtained clusters [50].

2.1.2 Statistics

A statistical model is a set of mathematical functions that describe the behaviour of the objects in a target class in terms of random variables and their probability distributions. In networks, a number of measures have been introduced to reflect the importance of nodes e.g., the most influential persons in social networks. These measures include:

- ***Degree Centrality***: The idea behind this measure is that "an important node is involved in a large number of interactions" [3]. It is defined as the total number of links the node has with other nodes. In the case of a directed network, this can be separated into two measures, namely ***out-degree*** and ***in-degree***, in which the former is the number of arcs from the node to others and the latter one is the number of arcs to the node from others. However, the limitation of this measure is that, only the local structure around the node is considered and not the global structure of the networks.

- ***Closeness Centrality***: The idea behind this measure is that "an important node is typically close to, and can communicate quickly with, the other nodes in the network" [86]. It reflects the reachability of a node within a network. It is defined as the number of other nodes divided by the sum of all distances between the node and all others. However, one limitation of this measure is that it is not applicable to networks with disconnected components.

- ***Betweenness Centrality***: The idea behind this measure is that "an important node will lie on a high proportion of paths between other nodes in the network" [3]. It reflects the central position of a node x in a network. It is defined as the proportion of shortest all-pairs paths in which the node x acts as a bridge. The limitation of this measure is that many nodes may not be located on a shortest path between two other nodes. This gives them a score of zero.

- ***Eigenvector Centrality***: The idea behind this measure is that "an important node is connected to important neighbors" [11]. It calculates the importance of a node in a network relative to all other nodes. In other words, let x be a node with connections to several nodes in the network. Connections to high-scoring nodes contribute more to the score of x, compared to connections to low-scoring nodes. A number of variations of this measure have been introduced into the literature, such as PageRank [14] and HITs [66].

  - ***PageRank (PR)*** [14] is a link analysis algorithm used by the Google search engine. The main idea behind PageRank is that it calculates a probability distribution that represents the likelihood that a person who is randomly surfing a graph through edges representing links between web pages, will arrive at any particular node. PageRank is an iterative algorithm, which starts by dividing the distribution among all nodes in the graph. After each pass, the approximate PageRank scores will be assigned to the nodes in the graph. These PageRank scores approximate the theoretical true value. Let PR be the vector of PageRank values. Using the recursive definition of PageRank [14], PR is computed as follows:

$$PR(v) = \frac{1-d}{|V|} + d \sum_{\forall u \to v} \frac{PR(u)}{|F(u)|}$$

    where $F(u)$ is the set of outgoing arcs from node u; hence $|F(u)|$ corresponds

to the out-degree of node u, and $u \to v$ means that there is an arc from u to v.

PageRank avoids the problem of "sink states" (i.e., nodes that have no outgoing edges), by assuming that from such nodes the surfer moves to some other random node in V. PageRank defines a *damping factor* of $1-d$ as the regulating residual probability of visiting a random node in a graph from a particular node. The appropriate value of $1-d$ depends on the application and graph. In the case of the web, a random user surfing the web will typically follow about 6 hyperlinks before becoming bored and choosing some other random web-page to surf. Thus, in this case, $1-d = 1/6 \approx 0.15$.

- ***HITs*** [66]: Kleinberg developed an algorithm that uses the link structure of the web to discover and rank pages (nodes) relevant for a given query. The algorithm finds the authoritative pages (nodes) that contain valuable answers to the query. However, to find "authority" pages, we need to find "hubs" pages. These are the pages that advertise or point to the "authority" pages.

  In a recursive way, each node is assigned two weights: an authority and a hub score. When a node v has a good hub score, this increases the authority scores of all nodes that v points to. Similarly, if v obtained a good authority score, this increases the hub scores of all nodes pointing to v. The authority and hub scores of v are defined as follows:

$$auth(v) = \sum_{\forall u \to v} hub(u) \qquad\qquad hub(v) = \sum_{\forall u \to v} auth(u)$$

- ***Clustering Coefficient***: This measure assesses the tendency of a group of nodes to form a cluster together. It is based on a theory that, in real world networks, nodes tend to be in clusters, in which each cluster has a high density of ties edges between its members. Two variations of clustering coefficient have been proposed:

- ***Global Clustering Coefficient***: This measure was first introduced by Luce and Perry (1949). It reflects the clustering in the whole network and can be applied on directed or undirected networks. Let a triad be a set of three nodes which are connected by two edges (open) or three edges (closed). The global clustering coefficient is defined as the number of closed triads divided by the number of closed and open triads in the network.
- ***Local Clustering Coefficient***: This measure is based on local density [113, 124, 128]. For each node, it is defined as the number of ties present between the node's neighbours divided by the number of possible ties between the neighbours. For example, if the neighbours of node x are all connected with each other, this gives x a local clustering coefficient of 1. Alternatively, if none of x's neighbours are connected to each other, this give x a score of zero.

2.2 Detecting and predicting social relationships

Similarly to our approach, Rowe et al. [110] approached the problem of inferring the Enron management hierarchy as a ranking problem. They employed information that reflects the flow of emails between users such as the number of sent or received emails. Additionally, they used information about the nature of connections in the email network. Examples of these include the number of cliques each user is involved in and the degree centrality scores. A "social" score for each user was computed using the aforementioned features by which the user was ranked. After that, users with similar scores were grouped together in order to determine different levels in the management hierarchy. Although we addressed the problem using a similar ranking methodology, our research studies the problem of inferring the direct hierarchical relationships between employees, not detecting a user's level in the hierarchy.

An earlier study using the Enron dataset by Klimt and Yang [67] was in the area of email classification into user-specific folders. In other words, given a user and a set of his/her defined email folders, the authors built a model to classify a new email sent to the user to one of these pre-defined folders. They used an SVM classifier in various ways. First, the individual parts of an email such as From, To, Subject and Body, were used as features. Then, they used all these features together with regression weights to train and test a classifier. Their results showed that the body section was the most discriminative feature among all the individual features in classifying emails. Furthermore, using ridge regression to combine all the features linearly proved to give better results.

Gupte et al. [47] premised their study on the assumption that the existence of a link in an interaction network such as Twitter indicates a social-rank recommendation. For example, a link showing that u is following v suggests that u recommends v. In other words, when there is no reverse link from v to u, this suggests that v has a higher social position in the hierarchy. In this study, the authors introduced the concept "social agony status". This status is caused when people connect to others (or follow others in the case of Twitter) who are lower in the hierarchy. Given a directed interaction graph, the nodes are ranked in a way that minimises the social agony. In the same study, a polynomial time algorithm was proposed to find the largest hierarchy in the directed graph. The results obtained over different online social networks showed how hierarchy emerges as the size of networks increase. Moreover, they proved that the degree of stratification (i.e. number of levels) increases slowly when the size of graph increases significantly. Compared to our study on detecting hierarchical relationships, this study focused on inferring a user's social stratification level, whereas our approach detects the superior of each user from an interaction network.

Agarwal et al. [2] extended the original Enron dataset which includes the direct manager-subordinate relationships between 155 employees. The new extended version included about 13724 dominance pairs between 1518 employees extracted manually by investigating the organisational charts of Enron. A dominance pair is a pair of employees who are connected by a series of manager-subordinate ties. However, in our experiments, we do not use this extended version since the email inboxes of those 1518 employees are not complete and the dominance relationships are not all immediate. Given that two employees are related in the hierarchy, the authors detected which person dominated the other. Comparing with our work, we focus on detecting the *immediate* dominance relationships from the email network without any previous knowledge whether the pairs of employees are connected in the hierarchy or not. Agarwal et al. proposed two approaches, (1) a predication model based on the degree centrality of the nodes, and (2) a model based on NLP techniques.

Other studies [61,85,95,96] analysed a company structure in general from the online interactions (emails) between employees. In these works, the authors proposed 6 popular metrics (in-degree centrality, out-degree centrality, betweenness centrality, closeness centrality, clustering coefficient and eigenvector centrality) and used them as a means to understand the relationship between the formal positions of the employees in a company hierarchy and the real but informal roles in a social network. The hierarchical network extracted from email communications were matched with the company's organisational chart. The main aim was to understand this relationship between the inferred hierarchical network and the real organisational chart which can play a key role in the process of redesigning the company structure. The authors ran experiments on two real datasets: Enron and a manufacturing company. Their results showed that the number of incoming emails (in-degree) was most helpful to distinguish between high level managers and others.

Tang et al. [118] developed a framework for detecting different types of relationships by learning across heterogeneous networks. In other words, as we did in our study, they inferred missing social relationships by analysing other types of relationships, namely interaction relationships. For example, using the log files of mobile calls between a group of users, they detected friendship relationships. However, their study premised on the assumption that a limited number of the relationships we want to infer, should be known beforehand. This is different from our context, where we focused on the extreme case when none of the social relationships are known. The features used in building their predictive model were based on four social psychological bases. The first base is "Social Balance" [31] which is a theory that suggests that people in a social network tend to form a balanced network structure. The theory states that in a triad, where the three nodes are users and the edges are the relationships between them, either all three users are friends or only one pair of them are friends. The second base is "Structural Hole" [16] which represents a person who is linked to people in parts of the network that are not otherwise well connected to one another. For example, consider three users a, b, and c, where both a and b are connected to c but there is no direct connection between a and b. In this case, c is a structural hole. The authors found that on average a structural hole (e.g., c) tends to have the same type of relationship with the other users (e.g., a and b). The third base used is "Social Status" [32,46,74] which is a theory based on directed relationships in a network. This theory supposes that each directed relationship is labelled by a positive "+" or negative "-" sign, where +/- denotes that the target node has a higher/lower status than the source node. The theory states that, in a triad, if each negative edge reverses its direction and converts it sign to positive, then the new triad should be acyclic. The fourth base they used is two-step flow theory "Opinion Leaders" [70] which states that ideas usually flow first to opinion leaders then to other people in the network. To validate the "Opinion Leaders"

theory they applied the PageRank algorithm on the interaction network and considered the top 1% ranked users as the opinion leaders. In our study, we reuse the "Opinion Leaders" theory by studying more deeply the appropriate settings for the PageRank algorithm to detect opinion leaders such as managers and PhD advisers. We also showed that Rooted-PageRank gives a significant improvement over the results obtained by the original PageRank algorithm.

In another study [8], the authors addressed the problem of recognising a user's romantic partner from the network connections between his/her friends. They developed a new measure of tie strength called "dispersion" which reflects the extent to which two people's mutual friends are well-connected. Their experiments were conducted over two large Facebook datasets. Results showed that the dispersion-based classifier is twice as good in identifying partners as an embeddedness-based classifier, where the embeddedness of a tie is the number of mutual friends shared between its two endpoints [82]. Further, they found that using machine learning models which were trained by dispersion-based features as well as some other interaction features (such as messaging, commenting, profile-viewing and co-presence at events and photos) produced even higher accuracy. However, dispersion does not seem relevant to the problem of detecting hierarchical relationships.

Burke et al. [15] identified differences in how parents communicate to their children on Facebook (giving advice, affection, and reminders to call) compared to their friends. In addition, they investigated the dominant topic in parents' discussion with their adult sons and daughters (discussing health issues, talking about their grandchildren, ...) compared to that with their adolescent children. The approach they proposed was based on quantifying user behaviour (who becomes friends with whom and when), mutual friends and how communications varies with children's ages and with the geographic distance to their parents. In the same study, a predictive language model of parent-child relationships was built, in which text features

were used in a technique known as elastic-net logistic regression. This model was used to classify the conversation target as a family member or general friend. The main findings in this study were, as previously found in offline research, that mothers tend to express their affection by reminding their children to call. Moreover, fathers often talk about shared subjects, such as sports, with their children. The study also found a linguistic shift when parents talk to their adult children compared to those who talk with their young children. In the former case, parents treat them as adult friends. However, unlike what has been observed in offline communications, online interactions between parents and their children do not decrease with geographic distance.

2.3 Analysing and modelling user behaviour

Classifying users by analysing their on-line activities is one of the most active areas in the field of social networks. One example is the study carried out by Kumar et al. on Flicker and Yahoo! data [69]. The users were classified as one of "passive members" of networks, "inviters" who motivate other offline friends to migrate online, or "linkers" who are fully active users. In another study [81], Maia et al. considered five individual features as well as four interaction features to group YouTube users into categories such as "small community member","content producer", "content consumer", "producer and consumer" and "other". Choudhury et al. [23] analysed users' interaction behaviour on the MySpace website to match each user to one of three defined roles: "generators", "mediators" and "receptors". In newsgroup applications, Golder and Donath [45] analysed the communications between users to classify the users' roles as "celebrities", "ranters", "lurkers" or "newbies".

Wu and Chen [135] carried out their research on a Chinese social network website called "17salsa-net" which is similar to Facebook. They analysed activities such as writing posts, sharing photos, commenting and sending emotions. They used the K-

means method to classify users. The best result was obtained for $k = 4$ where users were classified in four groups, "posting-pictures users", "literary users", "quickly-responding users" (responding to photos and text by sending a quick emotional click) and "commenting users".

The **ego-centric network** of a node is the network that includes the node, all its neighbours, and all edges/arcs between these nodes. Welser et al. [131] extended ego-centric network analysis to classify roles such as technical editor and substantive expert in Wikipedia. However, their approach included a high level of manual analysis which is not easily performed in large-scale forums. Himeloim et al. [52] addressed the problem of detecting social leaders in political forums using features such as the number of replies and the number of new threads initiated by users. Another study [72] used a "regular equivalence" approach [127] in which two users, who play the same role, must have something in common with respect to the relations they have with other users. However, this approach seems to be difficult to apply when the features used are more complex (not binary).

Chan et al. [19, 20] proposed techniques to analyse and classify the behaviour of the users in discussion forums. They carried out their experiment on a medium-sized forums (boards.ie). They defined nine features including user's popularity, reciprocity, length of interaction, initialisation, neighbours' roles, and the volume of communication measures. Then, a two-stage clustering method was used to group the users into 15 groups and eight roles. Analysing the forums using users' roles showed differences in the composition of the forums according to the subject.

Two main differences between the aforementioned studies and our study can be noted. First, these studies grouped users into categories of roles that are not known in the data. In the approach we propose in Chapter 6, the roles of users are pre-known in the dataset. Knowing these roles we can evaluate the effectiveness of our approach. Second, all these efforts addressed the problem in static settings. In

other words, the temporal aspect was not considered when the quantitative features of the activities were analysed. However, our approach investigates the time effects on the classification results.

Angeletou et al. [6] studied the health of discussion forums and the association between community evolution and the behaviour of its users. In this study, they built a semantic model and rules for representing users' behaviour over time. First, a method based on semantic rules was employed to match each user to his/her role. In this method, the dynamic features of the interactions were taken into account. Also an ontology was built to capture the behavioural features of users in a common machine-readable format. Using this approach, they analysed the data of three communities over three years. The experimental results showed that a greater proportion of "popular participants" leads to a higher level of activities. On the other hand, having more "ignored" users in the community decreases the activities in general. It was also found that a stable forum's leads to an increase in activities over time, whereas a forum, in which roles fluctuate, negatively affects the online community's health in the long term.

Rowe at el. [109] analysed the differences in the behaviour of different types of communities. The dataset they used belonged to, an IBM Connections enterprise social software system. Their study focused on the three types of communities: (1) "Communities of Practice (CoP)": a group of people with a common interest or practice share information, (2) "Teams": who work on a shared goal for a particular client, project or business function, and (3) "Technical Support Group (Tech)": who provide support for a particular technology. In order to identify the behavioural differences between communities, micro (user-level) and macro (community-level) features were analysed. A sliding window was used to capture the temporal aspect of the features. Micro-level features included the total number of items created by the user and the total replies sent/received by the user. On the other hand, the aim

of the macro features was to describe the attributes of each community type through the use of statistical measures. These features included the total number of users participating in the community as well as the total number of items of content which received replies from users. In the same study, user questionnaires were conducted to identify the link between the user's objectives and the characteristics obtained from analysing each community type. The empirical results showed differences between the three analysed types in both the micro and macro levels. For example, in "CoP" types, users were dispersed in their activities compared with "Teams" and "Tech" types. However, "Teams" users were more active in creating new content compared with other types. This is consistent with the questionnaire responses received by "Teams" users where they gave the ability to create new content the highest priority. Similarly "CoP" users believed that the ability to reply to existing content was the most important. This is consistent with experimental results which revealed that commenting on blogs/post was the highest in the "CoP" type.

Tang at el. [120] studied the problem of detecting influential users in social networks by considering dynamic interactions over time. Based on a temporal network model [119], they extended the static centrality metrics by introducing the definitions of temporal closeness and temporal betweenness to identify the key nodes in online social networks. Temporal closeness reflects how fast a member can spread a piece of information. On the other hand, temporal betweenness detects members who act as bridges in the most active communication paths over time. The evaluation on the Enron email dataset proved that temporal centrality metrics provide better understanding of dynamic processes and are more accurate in detecting key players who are able to speed up or block the process of information dissemination in online social networks compared to traditional static metrics. However, the authors do not consider the problem of inferring manager-subordinate relationships from the Enron email dataset.

2.4 Research in bibliographic networks

Bibliographic networks are defined as collaboration networks, where the nodes generally consist of one type of object such as authors or papers. When the nodes represent the set of authors, an edge (undirected link) between authors u and v exists, if they produced a joint work (paper, book, report, ... etc). Its weight $w(u, v)$ is equal to the number of works to which u and v both contributed [10]. Research on this type of network has focused on analysing the co-authorship network to rank authors according to their impact in the field [34]. Some other studies, attempted to predict future collaboration between authors [116]. In our research, we study this type of network to detect PhD advisee-advisor relationships between authors.

Another type of bibliographic network is a citation network. In this type, nodes represent the set of papers, and an arc (directed link) from paper p to paper p' exists if paper p cited paper p'. The aim of studies carried out on citation networks, is to rank the papers according to their importance.

In their original versions, most graph-search algorithms (such as PageRank and HITs) do not consider "the temporal dimension of data". As a result, several studies brought time into play in their approaches in order to retrieve higher quality results in the field of publication search. They followed the intuition that papers considered to be high quality in the past may not necessarily be considered high quality in the present or the future. Moreover, applying the original PageRank or HITs algorithms in bibliographic networks may assign higher scores to older papers since older publications tend to accumulate more in-links (citations) due to their longer existence, while recent publications of high quality may not be ranked highly.

Yu at el. [138] proposed several approaches based on the history of publications' citations as well as the source (author and journal) of each publication. In all their approaches, the content of the paper was not considered. In order to rank non-recent

papers, they depended mainly on the content of the publication and the dates when the publication was cited by other papers. The current importance of an old publication is determined from recent citations, which are considered more important than citations which occurred further in the past. To consider these differences in the importance of citations, a modification to the original PageRank formula was introduced ("TimedPageRank") where smaller weights were given to earlier citations. Also they predicted the importance of a particular paper in the future by analysing its citation behaviour within the last year. However, this technique does not seem relevant for ranking very recent papers with few or no citations. For such cases, they defined reputation-based features which use the reputation of both a paper's authors and its venue. The author's reputation was calculated by averaging the time-weighted PageRank scores of all his/her papers in the past. Similarly the journal's reputation was calculated by considering all papers published in the journal previously. The empirical results on a dataset of publications from 1992-1999 showed that "TimedPageRank" was significantly better than the original PageRank in ranking papers. Moreover, using "TimedPageRank" in combination with author and journal reputation returned the best results. When predicting the importance of very new papers, the reputation-based approach was promising with more than half of new high quality papers predicted from the ideal rank of high quality papers of the next year.

Driven by the similar motivations, and by studying the same problem, Li at el. [77] proposed a new method in which they used Markov chains and a random reader to formulate the problem. In all the modifications introduced, they aim to keep the requirement that the probabilities of going from one page to other pages sum to one. This is the main difference between this study and the aforementioned study [138], in which this requirement was violated. Li at el.'s approach was based on the stationary probability distribution which means that after a series of tran-

sitions, the probability of the random reader arriving at each state will converge to a steady state, regardless of the choice of the initial probability at each state. Also, the transition probability matrix A should be irreducible and aperiodic. The irreducibility of A means that the citation graph should be strongly connected. In other words, for any pair of nodes u and v, there is a directed path from u to v. As there is no directed path from older papers to newer ones, they added an artificial link (citation) from each paper to every paper and each link was given a small transition probability controlled by a time function $f(t)$ $(0 \leq f(t) \geq 1)$, where t is the difference between the dates of two publications. The reader will follow an actual link (citation) with a probability given by $f(t)$. This reader will jump to a random paper with probability $1 - f(t)$. If the paper was published a long time ago, then the value of $1 - f(t)$ of this paper should be large, which means that the reader will jump to a random paper with high probability. On the other hand, if the paper is new, the chance to follow a reference (citation) of the paper is high, and the probability to jump to a random paper is modest. The results obtained showed that their approach worked considerably better in ranking papers than the orginal PageRank and even better than the method proposed in [138].

In the same context, Fiala at el. [39, 40] studied the problem of ranking researchers (authors) instead of publications. In their earlier study [40], they used the PageRank algorithm to rank the authors of papers. The citation graph among authors was used, in which the weights assigned to the edges were based on information from the co-authorship graph as follows: a citation from a colleague (co-author) should contribute less to the prestige of the cited author than a citation coming from a foreign researcher, i.e. an author who has never co-authored a paper with the cited author. However, they consider some exceptions to this rule, such as where the number of shared papers of the two authors is relatively small compared to the total number of papers written by the author. They concluded that running PageRank

on a citation graph weighted in this way returns better results than those obtained when an unweighted citation graph is used.

Fiala developed the approach further in [39]. By considering also the date of the publication and date of citation, he aimed to weigh citations more discriminately. For instance, a citation of one author by another may be made before any shared papers were published. In such cases, this citation should not be considered as a friendly citation from a co-author. Fiala defined several "time-aware" modifications to calculate the weights of edges depending on several factors, such as the number of common publications and whether or not they were published before or after the citation was made. The modifications he introduced made the citation weights reflect the author's influence in a better way. For each "time-aware" modification introduced, a ranking of authors was generated and compared with the ranking obtained by the time-unaware counterpart method. Moreover, a number of common ranking methods (such as citation, in-degree, HITs, and PageRank) were tested, in order to evaluate the approach. All the methods proposed were tested on the "Web of Science" data for computer science journal articles from 1996-2005. The rankings obtained were compared with the two famous lists of award winners (ACM. A. M. Turing and ACM SIGMOD E. F. Codd). The results showed that generally most award winners were ranked highly by all time-aware rankings. However, no method was best overall since each individual ranking brought an improvement in some aspect. In all cases, each time-aware variant was always better than its time-unaware counterpart.

All the aforementioned research addressed problems that are different from our studied problem. In Chapter 3, 4, and 5, we consider the problem of detecting PhD advisee-advisor relationships by analysing a co-authorship network.

2.5 Educational Data Mining

There are many educational environments available, such as traditional classroom, e-learning [17], learning management systems (LMS) [79], intelligent tutoring systems [62], concept maps [90], social networks, forums, educational game environments [98], virtual environments [123] and ubiquitous computing environments [54].

The data provided by each of the aforementioned educational environments is different, making it possible to analyse various types of problems using Educational Data Mining (EDM) techniques. The EDM process converts raw data from educational systems into useful information that could have a significant impact on educational research and practice. This process does not differ much from other areas of application of Data Mining (DM), because it follows the same steps as the general DM process: preprocessing, DM techniques (such as classification, clustering, association-rule mining, sequential mining, and text mining, as well as other approaches, such as regression, correlation and visualisation that are more connected to statistical analysis), and post-processing.

According to Romero and Ventura [108], the most commonly applied DM tasks in education are regression, clustering, classification and association-rule mining. The most commonly used DM techniques/methods in educational research are decision trees, neural networks and Bayesian networks. Some authors suggest several EDM subjects as being relevant [18]:

1. Assessing and predicting students' learning performance from their online behaviour/interaction.
2. Developments for the detection of atypical student learning behaviours.
3. Grouping students according to some criteria such as personal behaviour. These groups of students can be used by the instructor to provide better teaching using adaptive personalised systems.

4. Applying social network analysis to study the relationships between students instead of individual attributes.
5. Evaluating learning material and educational web-based courses.
6. Constructing courseware in an automatic way.
7. Providing course adaptation and learning recommendations based on the student's learning behavior.
8. Providing feedback to teachers and students in e-learning courses to improve the learning process.
9. Visualising the educational data to highlight useful information and support decision making.

In this section, we only cover the first four subjects since they are related to the problem we study in Chapters 5 and 6. In these chapters, we address the problem of detecting the roles of students working on the same project from their online behaviour.

2.5.1 Predicting students' learning performance

Much research in the literature has focused on studying the problem of predicting students' performance, scores, or marks from their on-line interactions, some using *quantitative features*, others using *qualitative features*. As an example of the former, Palmer at el. [94] used stepwise multivariate linear regression to predict students' marks by analysing message frequencies in online discussion forums, including the number of posts and replies, the number of messages read, the length of threads, and students' average response time to other messages. Cheng et al. [22] used regression analysis, employing only the number of posts and page views to infer course performance.

In the same context, other studies considered *qualitative features* in their models. In other words, they analysed the discussion content, which helped in understanding the explicit semantic information in the transcript from the discussion forums. As an example of this, the study by Yoo and Kim [137] used stepwise regression analysis to analyse the question-answer dialogues between students as well as emotional features covered by LIWC (Linguistic Inquiry and Word Count) to infer the performance of participants in a project .

Using a combination of features has proved to be more effective in addressing certain problems in a number of applications. Lopez and Romero et al. [80, 107] used three types of features (quantitative, qualitative and SNA-based features) to predict students' pass/fail results. In their studies, they examined two sets of algorithms. The results obtained using "classification-via clustering" algorithms gave better results than those obtained using "supervised-learning" algorithms. In the same study, they filtered out irrelevant features by applying a set of feature-selection algorithms. In our approach, proposed in Chapter 6, we use a similar approach for selecting the most relevant and discriminative features from the data of students' online communication in order to detect the roles of students working on a project. We also used two types of features (quantitative and SNA-based) in building the classifier. However, Lopez and Romero built a *binary* classifier to predict students' results (pass/fail), whereas we detect *three* possible classes of students' roles.

2.5.2 Analysing and modelling students' behaviour

Research carried out in this area analysed the online behaviour of the students to detect any undesirable or unusual attitude, such as improper actions, cheating, wasting a lot of time, etc. Dekker et al. [33] demonstrate the usefulness of decision tree, Bayesian classifier, logistic models, the rule-based classifiers and random forest in predicting first-year student drop out. Agapito and Ortigosa [1] proposed an

approach based on the C4.5 decision tree [101] to detect potential symptoms of low performance in e-Learning courses. Cocea and Weibelzahl [27] estimate the motivational level of the student by analysing his/her log file data. The approach they built was also based on a decision tree classifier. The results showed that both the performance in tests and the time spent in reading are important factors in predicting students' motivation.

Some other research in this area focused on finding students' groups based on their observed online actions. The aim of these studies is to help the instructors in building better and personalised educational system according to each group requirements. In this task, researchers used two data mining techniques, clustering (unsupervised learning) and classification (supervised learning).

Various clustering algorithms were used in these studies. For example, Cobo at el. [26] used agglomerative hierarchical clustering to model students' behaviour. In this study, the total number of messages written and read by the student were used as features. In another study, Khan et al. [65] analysed the frequency of access as well as the duration of sessions (i.e. the duration the student is logged in) in order to cluster students into different categories according to their participation. They built their model using hierarchical cluster analysis (Ward's method). Zakrzewska [139] used a hierarchical clustering algorithm as well to divide students into groups according to their individual learning styles. Chen et al. [21] used K-means clustering in order to group students who experienced similar learning portfolios such as exam scores, assignment scores and on-line behaviour.

On the other hand, classification algorithms were widely used to group similar learners. For example, Superby et al. [117] built a model to classify university students as early in the academic year as possible in one of three groups, low-risk, medium-risk or high-risk. In this study, the authors used several classification algorithms such as discriminant analysis, neural networks, random forests and decision

trees. In another study [30], decision trees were used to classify students according to their accumulated knowledge in e-learning systems. Fok and Chen [41] proposed a hidden Markov-based model to classify students according to their navigation or content access patterns.

2.5.3 Social network analysis in education

More recently, *Social Network Analysis (SNA)* was used to analyse students' online behaviour in discussion forums. In these studies, the discussion forums are represented as networks, in which students are the nodes, and ties are the relationships between students (such as reply relationships). SNA analyses the network structure and gives insight about structural characteristics, such as the nodes with the most incoming or outgoing ties [84]. Premised on that, many SNA-based tools have been developed to improve the learning process through analysing data from forums. For example, Meerkat-ED [102] analyses the network structure and visualizes snapshots of connections between students. Similarly, SNAPP [9] gives the course instructor a better view of the evolution of students' relationships within discussion forums using SNA techniques. These techniques evaluate the correlation between students' participation and the learning objectives.

In addition, SNA techniques have been applied in several education-based applications. Reyes and Tchounikine [106] introduced SNA-based models for mining data collected from forum-like tools. Based on the theory stating that cohesion plays a central role in collaborative learning, the aim of this study was to help tutors in tracking the groups activities and to allow the tutors to pay attention to groups with a low level of social cohesion. In the same context, Reffat and Chanier [105] measured the cohesion of small groups in collaborative distance learning that used discussion forums. The proposed approach helped in detecting isolated people and active sub-groups. Rallo et al. [103] used data mining on social networks to analyse the structure and content of educational online communities. In other work, the

framework they proposed analysed the structure and the relationships that were established between all the members of education-based community. Nankani et al. [87] studied the educational collaborative network. They used SNA to support information fusion through collaborative channels and key participants or groups in the network. The aim of this study was to aid decision makers in an educational organisation to take appropriate action depending on the patterns detected.

2.6 Sequence Classification

Chapter 7 employs sequence mining techniques in order to detect the roles of users from their online interaction behaviour. In this section, we give a brief background of that field and present its main techniques.

With more and more data from various domains being produced in the form of event sequences, sequence mining has become an important and much-researched area. Depending on the types of events they contain, sequences can be discrete (e.g., symbolic sequences such as DNA, proteins, or text) or continuous (e.g., time series, such as sensor measurements, ECGs, or stocks). An important task within the area of sequence mining is sequence classification. For example, in health-informatics, an ECG can be used to classify an individual as healthy or sick [129]. Moreover, in genomics, sequence classification is employed to build models of known protein sequences in order to detect the function of a new protein [35]. More recently, research has focused on tracking suspicious series of financial transactions in a bank to detect money laundering or other fraudulent activities [78].

Sequence classification methods in the literature can be divided into three categories: distance-based methods such as the baseline approach we describe in Chapter 7, feature-based methods, such as our multi-granularity framework introduced in Chapter 7, and model-based classifiers such as Hidden Markov Models (HMM) which are not considered in this thesis.

In feature-based classification, a sequence is transformed into a single vector of features. Such vectors are typically constructed by viewing the sequence as a bag-of-words (i.e., a word is a symbol), and they provide a summarisation of the sequence. For example, a vector can be the histogram of the mean frequency of each event symbol included the sequence. However, this approach will ignore the sequential order of the events. A modified method, called k-grams, was proposed in [37], where each sequence of k consecutive events is treated as a single feature. Using k-grams, each sequence can be represented as a binary vector indicating the presence and the absence of each k-gram in the vector. Given that each sequence has been converted to a feature vector, a supervised learning algorithm, such as an SVM [75, 76] or a decision tree [24] can be used to train the classifier. Usually if the number of k-grams is large, some feature-selection technique is required to retrieve the most relevant features. For example, Chuzhomova et al. [24] employ a genetic algorithm to find the best subset of features.

One can also use a pattern-based approach to build a feature-based classifier [68, 73]. In this approach, sub-sequence patterns are considered as features. These sub-sequences must satisfy some pre-defined criteria, such as being frequent and distinctive in at least one class, and not redundant. In our study, we adopt this technique, and employ an existing sequential pattern mining algorithm, SPAM [7], to mine frequent sequences, which are then used as features for training our classifier.

On the other hand, distance-based methods, also known as lazy-learners, use a similarity function that measures to what extent two sequences are similar. Euclidean distance [63, 129] is a similarity measure commonly used in time-series classification when the compared sequences are of the same length and phase, while Dynamic Time Warping [64] is used when more flexible matching is desired. Under the same category, alignment-based methods have been used in several applications in which the sequences consist of symbols [60]. Two types of functions have been

proposed: (1) global-alignment functions, such as the Edit Distance, which compute an optimum global alignment score through dynamic programing [89], and (2) local-alignment functions, such as Smith-Waterman [115] and BLAST [5], which calculate scores between two sequences based on most similar sub-regions. Once the similarity scores have been calculated, an existing classification algorithm, such as k-nearest neighbour or SVM with a local alignment kernel [112], can be applied. A thorough overview of sequence classification algorithms is outside the scope of this thesis but can be found in [136].

2.7 Computational tools

We now review some of the computational tools used in this thesis.

2.7.1 Weka

Weka [134] is open source software written in Java and developed by the University of Waikato under the GNU General Public License. It includes a collection of machine learning algorithms which can be applied to data mining tasks. These tasks include data pre-processing, clustering, classification, regression, visualisation and feature selection. All Weka's algorithms can be accessed directly either through a graphical user interface or from one's own Java code [58].

The input data has to be available in a single flat file or relation where each point is represented by a fixed number of attributes (numeric or nominal values). Weka also supports SQL databases using Java Database Connectivity and can process the results returned by a database query.

We used Weka when building models for detecting the roles of users working on a shared activity and interacting online (Chapters 6 and 7).

2.7.2 Eclipse

Eclipse is a generic Integrated Development Environment (IDE). It is a free Java-based open source platform which is released under the Eclipse Public License, and allows software developers to build Java applications. However, users can extend its abilities by installing plug-ins written for the Eclipse platform, such as development toolkits for C/C++, R, Perl and Python. Users can also write and contribute their own plug-in modules.

We used Eclipse when implementing and evaluating all approaches discussed in this thesis.

2.7.3 JUNG

JUNG (Java Universal Network/Graph Framework)[1] is a software library that provides a common and extendible language for the modelling, analysis, and visualisation of data that can be represented as a graph or network. It is written in Java, which allows JUNG-based applications to make use of the extensive built-in capabilities of the Java API, as well as those of other existing third-party Java libraries.

The JUNG architecture is designed to support a variety of representations of entities and their relations, such as directed and undirected graphs, multi-modal graphs, graphs with parallel edges, and hypergraphs. It provides a mechanism for annotating graphs, entities, and relations with metadata. This facilitates the creation of analytic tools for complex data sets that can examine the relations between entities as well as the metadata attached to each entity and relation. The current distribution of JUNG includes implementations of a number of algorithms from graph theory, data mining, and social network analysis, such as routines for clustering, decomposition, optimisation, random graph generation, statistical analysis,

[1] `http://jung.sourceforge.net`

and calculation of network distances, flows, and importance measures (centrality, PageRank, HITS, etc.).

JUNG also provides a visualisation framework that makes it easy to construct tools for the interactive exploration of network data. Users can use one of the layout algorithms provided, or use the framework to create their own custom layouts. In addition, filtering mechanisms are provided which allow users to focus their attention, or their algorithms, on specific portions of the graph.

We used this library for implementing the PageRank and Rooted-PageRank algorithms according to our proposed approach (Chapter 3).

2.8 Summary

In this chapter, we reviewed work in social network analysis that is related to our research. We also described briefly the computational tools used in our experiments.

Although many efforts focused on studying the problem of detecting hierarchical relationships, none of the previous work has studied the extreme case when none of the social relationships between actors are known beforehand. In this thesis, we address this extreme case and investigate whether the use of timestamps related to online interactions between actors can improve the detection of both hierarchical relationships between actors and hierarchical roles played by the actors. We also conduct further investigation to examine the ability of heterogeneous models, based on both temporal and structural features of interactions, to improve the results obtained using homogeneous models.

Furthermore, in this thesis we develop a novel sequence classification technique that has not been proposed previously. This technique is based on capturing frequent sequences at multiple levels of temporal granularity. We show that our sequence classification technique is competitive in detecting the roles of actors where the interactions of each actor are captured as a sequence of communication events.

Chapter 3

Detecting Hierarchical Social Relationships using Structure-based Algorithms

3.1 Overview

As mentioned in the introductory chapter, social relationships between actors are not always explicit. The on-line interactions between a group of people can be analysed to detect these implicit social relationships.

In this chapter, we address the following problem: given a set of actors, who are connected via a social network, we want to infer potential hierarchical relationships that may exist between these actors. For example, given a set of employees in a company, we are interested in relationships such as that between a manager and subordinate, or, given a set of co-authors, we want to infer relationships such as that between a PhD advisor and advisee. Our proposed approach takes into account the interaction between the actors in the network, and infers these hierarchical relationships by exploiting any interaction patterns that may occur during their communication.

The main contributions of this chapter are summarised as follows:

- We study the problem of inferring hierarchical relationships in a social network and approach it as a ranking problem.
- We develop a model to detect the hierarchical relationships using three link-analysis ranking methods: Vertex-degree Centrality, PageRank, and Rooted-PageRank. We use the first two methods as a baseline, then use the third method to solve the problem based on the opinion leader theory.
- We demonstrate the performance of these methods on two large real social networks: the Enron e-mail network and a co-authorship network. In our experimental findings, we observe that Rooted-PageRank performs well and can achieve competitive recall values.

The results presented in this chapter have been published previously in [57].

3.2 Background

In this section, we provide the appropriate definitions and formulate the problem studied in this chapter.

3.2.1 Problem Setting

Let V denote the set of *actors* (members) of a social network. We consider two types of graphs defined over V: the *interaction graph* and the *hierarchy graph*.

Definition 1 (Interaction Graph) An *interaction graph* is defined as $G_I = (V, E^c, W)$, where E^c is the set of edges (directed or undirected) representing the interactions between the actors in V and W is a vector of edge weights, where $w_e \in W$ corresponds to the weight of the edge e connecting nodes u and v.

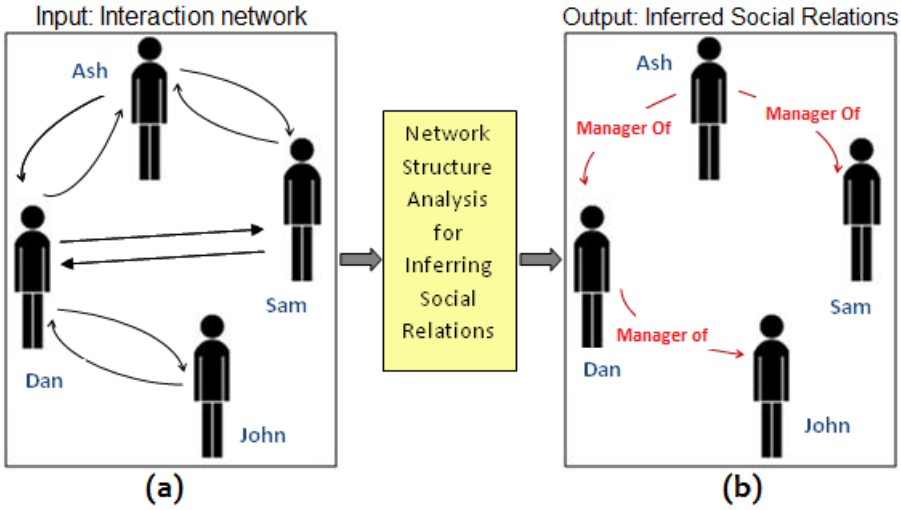

Figure 3.1: Inferring implicit social relationships from interaction networks.

We note that G_I can be modelled both as a directed or undirected graph, as well as weighted or unweighted, depending on the nature of the interactions and the application domain at hand.

Definition 2 (Hierarchy Graph) A *hierarchy graph* is a directed graph defined as $G_H = (V, E^s)$, where $E^s \subseteq V \times V$ is a set of of edges representing the hierarchical relationship between the actors in V. Each edge $(u, v) \in E^s$ indicates that actor $u \in V$ is the direct superior of actor $v \in V$ in the hierarchy.

3.2.2 Problem Formulation

Given a set of actors V and their corresponding interaction graph G_I, the problem is to infer the hierarchy graph G_H of V.

For example, given a set of e-mails exchanged among employees, or papers co-authored by authors, we want to infer manager-subordinate relationships in a company, or advisor-advisee relationships in academia, respectively.

In Figure 3.1 we can see an example of the problem we want to solve. Given the interaction graph (a) of the four actors (employees), we want to infer their corresponding hierarchy graph (b).

3.3 Methods

We use three structure-based methods, namely **degree centrality (DC)**, **PageRank (PR)** and **Rooted-PageRank (RPR)**. The first two measures were explained in chapter(2).

The **Rooted-PageRank (RPR)** idea was first introduced by White and Smyth [133], who based their model on [51] and [59] which investigated extending the PageRank algorithm to generate "personalized" rank. The algorithm starts with a predefined root node set $R \subseteq V$. This root set is the set of nodes relative to which all other nodes are ranked. The difference between general PageRank and rooted PageRank is that in the latter, the random walker will jump back to a node in R with a "back probability" β, where $0 \leq \beta \leq 1$. As β become larger, the probability of seeing the random walker at any node in R increases and as a result, the influence of the root nodes in R grows. However, even when $\beta = 0$, the influence of root nodes is still present since the random walk is launched from a node in R.

The algorithm defines a "prior probability" p_v for each node v, with $\sum_{\forall v \in V} p_v = 1$. This measure reflects the importance of root node v relative to other root nodes. Usually equal prior probabilities are given to all nodes in R, so that $p_v = 1/|R|$ if $v \in R$ and $p_v = 0$ otherwise. The formula of Rooted-PageRank can be defined as:

$$RPR(v) = \beta p_v + (1 - \beta) \sum_{\forall u \to v} \frac{RPR(u)}{|F(u)|}$$

where $F(u)$ is defined as before.

3.4 Using a Structure-based Model

In this section, we first propose approaches based on degree centrality (DC) and on PageRank (PR), before discussing the approach based on Rooted PageRank (RPR).

The key idea of these approaches is based on the "opinion leader" theory. This theory has been studied for a long time in communication science, being first introduced by Lazarsfeld et al. [70]. The theory assumes that ideas/innovation usually flow from opinion leaders to all other actors in a network. In other words, actors who are classified as opinion leaders have more influence on others than ordinary actors. Accordingly, opinion leaders are more likely to have higher social position (such as managers or advisors) than ordinary actors (such as subordinates and advisees).

3.4.1 Degree/PageRank based Approaches

As a starting point, we classify actors in the network into two classes: (a) opinion leaders (OL) and (b) ordinary users (OU). To do this, we follow three steps (see Figure 3.2):

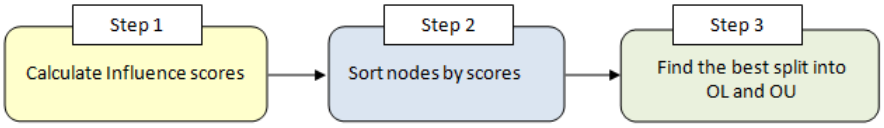

Figure 3.2: Main steps in the Degree/PageRank based approach.

1. First we assign a score to each actor. This score should reflect the importance (influence) of the actor in the whole network by analysing the interaction network structure. We employ the aforementioned algorithms, PageRank (PR) and Degree Centrality (DC), to assign a score to each node, as defined by the following two functions, respectively:

$$Score_{RP} : V \rightarrow \mathbb{R} \text{ and } Score_{DC} : V \rightarrow \mathbb{R}$$

2. We sort the actors according to their scores, with actors having higher scores (those in OL) appearing earlier in the list, compared to those with lower scores (in OU).

3. Finally, assuming that the set of hierarchical relationships E^s is known, we find the best position to split the sorted list of actors into two sub-lists, one representing OL and the other OU. The best position is the one that maximises the percentage of hierarchical relationships $(a, b) \in E^s$ (a is the superior of b) for which it is the case that $a \in OL$ and $b \in OU$.

   To find the best position, we first calculate the following probabilities for each possible split position i:

$$P_i(OL \rightarrow OU) = P(a \in OL \wedge b \in OU) \tag{3.4.1}$$

   Note that if the sorted list of actors is given by $(v_1, v_2, \ldots, v_i, \ldots, v_N)$, then OL is the set of opinion leaders $\{v_1, v_2, \ldots, v_i\}$, OU is the set of ordinary users $\{v_{i+1}, v_{i+2}, \ldots, v_N\}$, and a is a superior of b.

   The best split position is that position k where we get the maximum probability for $P_i(OL \rightarrow OU)$, i.e.,

$$k = \underset{i}{argmax} \;\; P_i(OL \rightarrow OU) \tag{3.4.2}$$

In order to compare this approach with the RPR-based approach explained in the next section, we also evaluate the results in an alternative way. For each split position, we calculate the proportion of actors whose superiors appear within the list of opinion leaders.

First, we define a function $minOL$ on actors. For $b \in V$, this function returns the length of smallest opinion leaders list that includes the direct superior a of the actor b as follows:

For each $b \in V$, such that $a \in V$ is the superior of b

$$minOL(b) = \begin{cases} |OL| & \text{if } b \notin OL \wedge a \in OL \\ |OL| - 1 & \text{if } b \in OL \wedge a \in OL \end{cases} \tag{3.4.3}$$

where OL is the smallest opinion leaders list which includes the superior a.

Example Consider the ranked list of actors given by (x, y, z, k, l, m). Assume that y is the superior of z, and k is the superior of x. Then

$minOL(z) = 2$ since (x, y) is the smallest OL that contains y, and $z \notin OL$.

$minOL(x) = 3$ since (x, y, z, k) is the smallest OL that contains k and $x \in OL$.

Next we define the following recall function on the split position i:

$$\rho(i) = \frac{|U|}{|E^s|} \times 100 \tag{3.4.4}$$

where

- E^s is the set of pairs of actors, representing the hierarchical relationship between the actors.
- $U = \{x | x \in V \quad \wedge \quad \exists y \in V \quad \wedge \quad (y, x) \in E^s \quad \wedge \quad minOL(x) \leq i\}$.

This function finds the percentages of relationships in which the superiors appear within the top i actors of the ranked list.

3.4.2 Rooted PageRank-based Approach

In this approach, we address the problem in a different way. Instead of calculating the global importance of each node, we evaluate the importance of nodes from each actor's perspective separately (local view). In other words, a node x may have a big influence on node y. However the same node x may be ineffective to another node z. Accordingly, finding the influential users is a relative issue. We may end up with completely different results when finding the most influential actors for two different nodes.

To determine relative influence, we employ the aforementioned Rooted-PageRank (RPR) algorithm. This algorithm takes as input a set of root nodes. By analysing the network structure, RPR calculates the importance scores of all nodes relative to the root nodes. In our application, we choose a single root node representing the person whose direct superior we are trying to detect. For example, in the emails dataset, a manager x should be important to his/her subordinates. As a result, x should have high rank in the list of most influential people for his/her subordinates. Similarly, in coauthor networks, an advisor should appear in a high position within the important actors of his/her advisees.

As in the previous approach, this approach passes through three stages:

1. For each node $x, x^* \in V$ such that $\exists(x^*, x) \in E^s$, we run RPR with root node x to evaluate the importance of nodes relative to x. Each node $v \in V$, $v \neq x$, in the network will have a score returned by the function:

$$Score_x(v) = RPR_x(v) \tag{3.4.5}$$

2. Then we produce a list $L(x)$ sorted according to the importance scores, where $L(x) = [v_1, v_2, ..., v_i, ..v_{|V-1|}]$, such that:

   $Score_x(v_i) \geq Score_x(v_j)$ and $1 \leq i < j \leq |V| - 1$

3. Finally, assuming the superior of x is x^*, the rank of x^* in $L(x)$ is returned by the function $rank(x, x^*)$. This function returns the total number of actors in $L(x)$ who have an RPR score greater than or equal to the score of x^*.

   We define $rank(x, x^*)$ as follows:

$$rank(x, x^*) = |\{v : v \in L(x) \wedge score_x(v) \geq score_x(x^*)\}| \tag{3.4.6}$$

In order to evaluate our approach we use the recall function ρ defined in Equation (3.4.4). This function represents the percentage of relationships in which the superiors appear within the top i actors of their inferiors' ranked lists.

$$\rho(i) = \frac{|\{x : rank(x, x^*) \leq i\}|}{|E^s|} \times 100$$

where E^s is the set of pairs of actors representing the hierarchical relationship between the actors.

3.5 Datasets

3.5.1 Enron email dataset

The Enron dataset is one of the richest email datasets in the field. It includes the internal emails exchanged between employees working in a real corporation. It also includes the external emails between the customers and employees. All these emails were exchanged in the period 1998–2002. There are several versions of this dataset which have been made accessible to the public. Starting from mid-2002, when financial problems hit the company, the Federal Energy Regulatory Commission (FERC) made a dataset of 619,449 emails from 158 Enron employees available, after all attachments had been removed. The first revised version was released by Cohen in 2004, in which he placed each email in a separate text file using the mbox format. Later on, efforts were made to convert the dataset into a variety of formats. At the same time, much work addressed inconsistencies and integrity issues within the dataset.

In our research, we adopted the version which was provided with the ground truth of manager-subordinate relationships by Diehl et al. [36]. They also pre-processed the data by deleting extraneous, unneeded emails and fixing some anomalies in

the collection, such as empty or illegal user email addresses. Also all duplicated and blank emails were dropped. The version we used consists of 155 employees. However, some employees used more than one email address. In such cases, we chose one of them randomly and replaced the others with the chosen one. Additionally, in our experiments, we removed all email addresses belonging to people not employed by Enron, since they would only add noise to our analysis. As a result, only *internal* emails exchanged between Enron employees were considered.

We ended up with 155 employees, each with a unique email address, and 147 manager-subordinate relationships, formalising the management hierarchy in which 8 employees are top-level managers with no superior. Apart from these top-level managers, each employee has only one manager, while a manager can have many subordinates. The statistics of the full dataset can be found in Table 3.1.

	Enron	Co-author
$\|V\|$	155	1036990
E^c type	Emails	coauthoring
$\|E^c\|$	47738	1632442
E^s type	manager-subordinate	advisor-advisee
$\|E^s\|$	147	2098
From	1998	1967
To	2002	2011

Table 3.1: Statistics of Enron and coauthor datasets.

3.5.2 Co-author dataset

The co-author dataset represents a network of authors. It is built from data crawled by Tang et al. [121] from Arnetminer.org, which is an intelligent system for extracting and mining academic social networks. It includes more than one million authors who contributed to about 1.46 million papers in total between 1967 and 2011. The total number of co-author relationships is about 3.8 million. Each paper has a title, date, conference where it was published and a list of authors. Many efforts, such

as by Tang et al. [122], have been made to infer the advisor-advisee relationships between the authors. In this study, they created ground truth data by collecting the advisor-advisee information from the Mathematics Genealogy project[1] and the AI Genealogy project[2]. Also they crawled the researchers' home pages to detect advisor-advisee ties. In our research, we used an extended version of their dataset. This contains 2099 advisor-advisee relationships. The statistics of the dataset are presented in Table 3.1.

3.6 Experimental Results

3.6.1 Classifying Opinion Leaders and Ordinary Users

As mentioned earlier, we want to find the position at which to split the list of actors into sub-lists OL (opinion leaders) and OU (ordinary users) such that the maximum percentage of hierarchical relationships (u, v) has $u \in OL$ and $v \in OU$. Figure 3.3(a) shows the percentage results we obtained using Equation (3.4.1) for the Enron dataset. For PageRank (PR), the best split position is when we consider OL as the top 20% of the sorted list of employees. At that position, for 72% of manager-subordinate relationships, the manager appears in OL and the subordinate in OU. Degree Centrality (DC) reveals the same pattern but with worse results: the best split position is after the top 35% of the sorted list, with only 56% of manager-subordinate relationships correctly classified.

On the other hand, in the co-author dataset, the best split position is roughly the same for both PR and DC, namely after the top 3% (see Figure 3.3(b)), with 65% and 56% of the advisor-advisee relationships being correctly classified, respectively.

[1] http://www.genealogy.math.ndsu.nodak.edu
[2] http://aigp.eecs.umich.edu

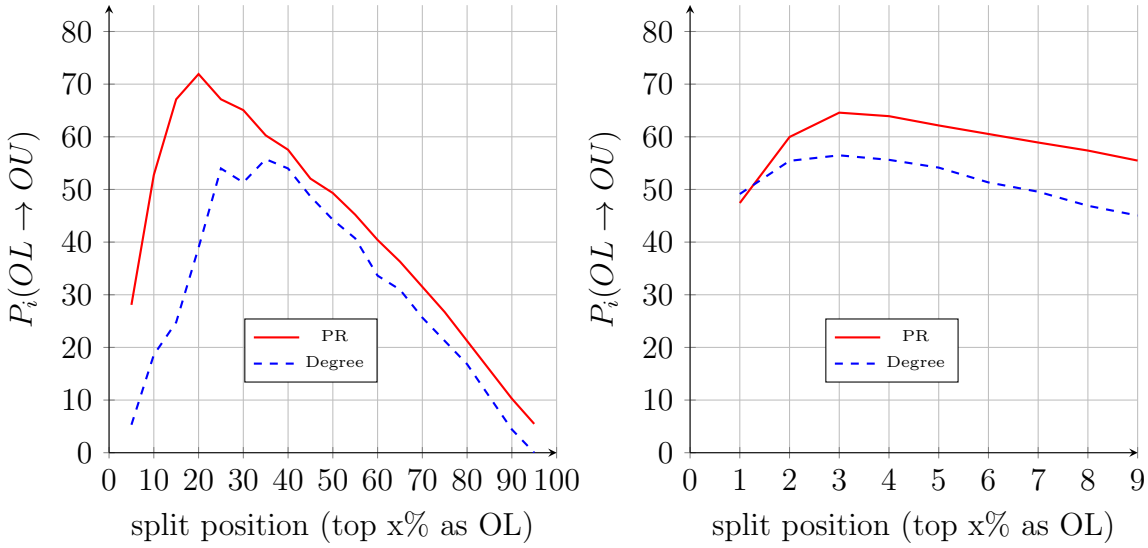

Figure 3.3: Results of classifying actors into opinion leaders (OL) and ordinary users (OU) using PageRank (PR) and Degree Centrality (DC) on (a) the Enron dataset and (b) the co-author dataset.

3.6.2 Detecting hierarchical ties

For this experiment, we ran Rooted-PageRank (RPR) n times, where n is the number of relationships in the network, with a different actor (inferior) as the root node each time. Also, we re-evaluated the results obtained by PageRank (PR) and Degree Centrality (DC) using Equation (3.4.4). Figures 3.4 and 3.5 show the results on the Enron and co-author datasets, respectively.

Split-Position	Degree Centrality	PageRank	Rooted-PageRank
1	4.79	6.84	29.45
3	16.43	15.06	59.58
6	32.19	24.65	73.97
10	48.63	48.63	86.30
15	48.63	59.58	88.35
20	62.32	69.86	89.72
30	76.02	85.61	94.52
40	77.39	86.98	95.89

Table 3.2: Percentage results for the Enron dataset.

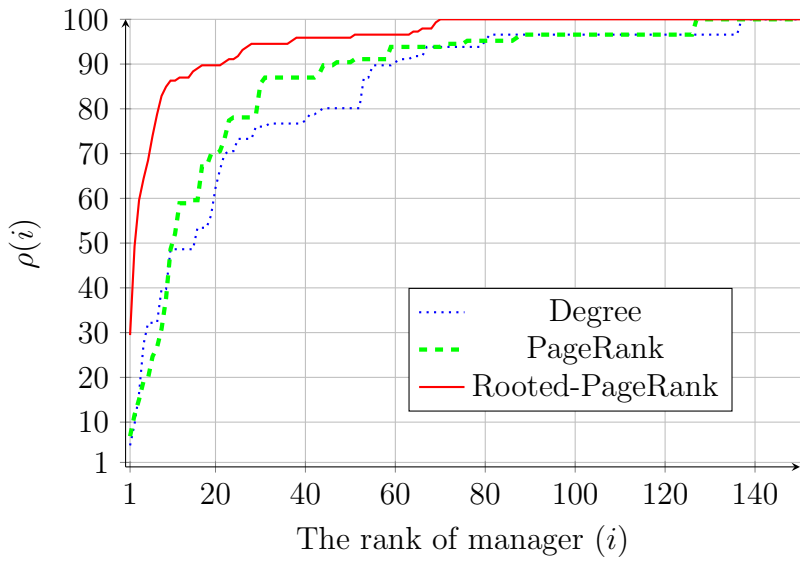

Figure 3.4: Results of PR, DC and RPR for the Enron dataset. This represents the percentages (y axis) of manager-subordinate relationships in which the managers have rank less or equal to i (x axis).

As shown in Table 3.2, using PR and DC respectively on the Enron dataset, we detect only 6.84% and 4.79% of the relationships in which the managers appear at the top of the sorted lists of their subordinates. RPR gives a significant improvement over these results by returning the employee's manager as the top result for almost 30% of the cases. Moreover, in RPR, for about 88% of subordinates, the managers are within the top 15 employees of the sorted lists of subordinates. The percentage increases to about 95% when we consider the top 30 employees of each subordinate's sorted list. On the other hand, PR can detect the manager of only 59.5% of relationship when we consider the top 15 employees in the sorted list. Degree Centrality is even worse with 48.6% of detected relationships appearing within the top 15 employees.

As shown in Table 3.3, when using RPR on the co-author dataset, 39% of advisors (821 out of 2099) appear as the top result within the sorted list of authors related to their advisees. In addition to that, for about 88% of advisees, the advisor appears in the top 20 results. This percentage rises to 95% when we look at the top 40 results. On the other hand, DC places up to only 5.4% of advisors in the top 300 authors within the sorted list. PR places only 4.5% of advisors within the top 300 authors.

Split-Position	Degree Centrality	PageRank	Rooted-PageRank
1	0	0	39.27
5	0	0	69.06
10	0	0	78.45
20	0	0	87.08
50	0.42	0	96.04
70	0.42	0.42	97.18
100	1.33	0.42	97.90
300	5.48	4.52	98.95

Table 3.3: Percentage results for the co-author dataset.

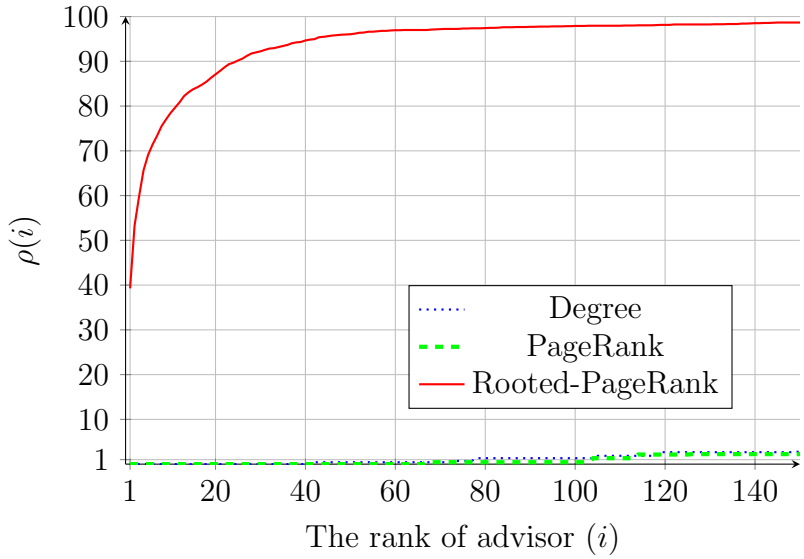

Figure 3.5: Results of PR, DC and RPR for the co-author dataset. This represents the percentages (y axis) of advisor-advisee relationships in which the advisors have rank less or equal to i (x axis).

The reason for obtaining higher percentages in the Enron dataset is that the total number of nodes is much larger in the co-author dataset (more than 1 million nodes) compared to the Enron dataset (only 155 nodes).

The aforementioned results confirm that addressing the problem of detecting hierarchical relationships using local view-based methods (such as RPR) is more effective than using global view-based methods (such as PR and Degree). In other words, in our problem the importance of actors is a *relative* rather than absolute measure.

3.7 Summary

In this chapter, we presented our initial investigation in the field of inferring implicit hierarchical ties between actors from their interaction networks. We introduced the main definitions and formulated the problem to be addressed. In this context, three methods were applied, PageRank (PR), Degree Centrality (DC) and Rooted-PageRank (RPR). We considered as examples detecting manager-subordinate relations between employees from their exchanged emails and discovering supervision relationships in academia by analysing the network of co-authored papers. Our experiments showed that RPR performs considerably better than both the PR and DC algorithms in terms of the percentage of correctly detected hierarchical relationships in our two datasets. We compare the computational costs of RPR with alternative approaches later in Section 5.3.5.

In the following chapters, we investigate whether considering the patterns of interactions over time can improve the inference of hierarchical relationships between actors.

Chapter 4

Detecting Hierarchical Relations using Time-based Methods

4.1 Overview

In the previous chapter, we approached the problem of detecting hierarchical social relationships in on-line social networks as a ranking problem. Three well-known algorithms PageRank, Degree Centrality and Rooted-PageRank were used to investigate this problem. The results showed that Rooted-PageRank significantly outperformed both PageRank and Degree Centrality in detecting manager-subordinate and advisor-advisee relationships in our Enron and Co-author datasets.

In this chapter, we continue our study and propose approaches that take into account the interaction patterns that occur during the communication between the actors. In other words, we investigate the temporal dimension of user interactions in addressing the problem of inferring the hierarchical relations. Our intuition is that actors connected by a hierarchical relationship will exhibit different temporal interaction patterns to those who are not. Our findings demonstrate that for our problem setting "time matters".

The main contributions of this chapter are as follows:

- As in the previous chapter, we study the problem of inferring hierarchical ties in a social network, approaching it as a ranking problem.
- We consider the link-analysis ranking method proposed previously, Rooted-PageRank(RPR), as a baseline approach.
- We propose two novel time-based approaches, called Time Function (Time-F) and Filter-and-Refine (FiRe), which take into account the time dimension of the interactions.
- We study the performance of these methods and compare them with Rooted-PageRank in terms of recall on our large real social networks: the Enron e-mail network and a co-authorship network. Our experiments show that the time-based methods achieve considerably better results than Rooted-PageRank (R-PR). In the Enron network, up to 44% of manager-subordinate ties were detected by time-based methods, compared to only 29% by RPR. Similarly in the co-author network, about 62% of advisor-advisee ties were detected by time-based methods, a significant improvement over the 39% achieved by R-PR.

The results presented in this chapter have been published previously in [57].

4.2 Time-based Methods

In this section, we propose methods which consider the time dimension of actor interactions, in an attempt to improve on the results of Rooted-PageRank. Our objective is, for each actor x, to improve the rank of y with respect to x should he/she exhibit a temporal communication pattern that suggests a strong hierarchical tie with x. We carry out our study based on the timestamps associated with the

interactions between actors. For each pair of actors x and y, we define a time-series $T_{x,y} = [i_{t_1}, i_{t_2}, ..i_{t_n}]$ representing their interactions over time. Analysing the time-series patterns of such interactions could improve the detection of hierarchical ties, in that two actors who are related by a hierarchical tie may interact over time in a different way from how they interact with other actors.

4.2.1 Time Function Model (Time-F)

As in Rooted-PageRank, for each actor x, we rank all other actors according to their calculated scores with respect to x. However, in this model, the scores employed to rank actors are calculated as follows:

$$TimeScore_x(y) = \sum_{t=1}^{n} f_{xy}(t) \tag{4.2.1}$$

where:

- x is the target actor (whose superior we wish to discover).
- $TimeScore_x(y)$ is the score of actor y with respect to x
- t is a time slot.
- n is the total number of time slots.
- $f_{xy}(t)$ is the score between x and y over time slot t.

The definition of $f_{xy}(t)$ varies according to the meaning of the hierarchical ties we are trying to detect. For example, if we are trying to detect a hierarchical tie which might be indicated by frequent and regular interactions, we define $f_{xy}(t)$ as follows:

$$f_{xy}(t) = \begin{cases} \frac{n_t}{N_t} & \text{if } N_t > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2.2}$$

where:

- n_t is the total number of interactions between x and y within t.
- N_t is the total number of interactions between actor x and all other actors within t.

On the other hand, if we expect a hierarchical tie to be indicated by early interactions, we define $f_{xy}(t)$ as follows:

$$f_{xy}(t) = \begin{cases} (1 - \frac{n_s}{N_s}) \times \frac{n_t}{N_t} & \text{if } N_t, N_s > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2.3}$$

where:

- n_s is the number of time slots between t and the first interaction of x.
- N_s is the number of slots between the first and last interactions of x (so n_s ranges from 0 to $N_s - 1$).
- n_t and N_t are as above.

In this function, y will have a high score in x's ranked list if x and y interacted in the early stages of x's activities. This is captured by the term $(1-\frac{n_s}{N_s})$ in the formula above. Moreover, in those early stages, the proportion of x's interactions with his/her superior y is expected to be greater than the proportion of x's interactions with others. This is captured by the term $\frac{n_t}{N_t}$ in Equation (4.2.3).

We use Definition (4.2.2) for $f_{xy}(t)$ when detecting manager-subordinate ties where interactions are email exchanges, and Definition (4.2.3) for $f_{xy}(t)$ when detecting advisor-advisee ties where interactions are paper co-authorships which are generally more intensive in the early stages of the advisee's activities. In other applications, alternative weightings could be used for the terms in $f_{xy}(t)$.

For each actor x, we generate the list of actors $LT_x = [y_1, y_2, ..y_{|V|-1}]$ which is ranked according to the scores calculated by Equation (4.2.1). The rank of his/her superior x^* is returned by the function $rank(x, x^*)$. Hence, the rank of the superior x^* of x is the number of actors in $LT(x)$ who have scores greater than or equal to the score of x^*.

The formal definition of $rank(x, x^*)$ is as follows:

$$rank(x, x^*) = |\{y : y \in LT(x) \wedge TimeScore_x(y) \geq TimeScore_x(x^*)\}| \quad (4.2.4)$$

We evaluate the ability of Time-F to detect hierarchical ties using the same function used to evaluate RPR. This function is based on calculating the recall, corresponding to using a position i, as follows:

$$\rho(i) = \frac{|\{x : rank(x, x^*) \leq i\}|}{|E^s|} \quad (4.2.5)$$

where E^s is the set of pairs of actors representing the hierarchical relationships between actors. For example, if $(u, v) \in E^s$, this means that u is the superior of v.

4.2.2 Filter-and-Refine Model (FiRe)

In the following approach, we *filter* using Time-F and then *refine* using RPR. The process of detecting hierarchical ties for an actor x consists of four steps as shown in Figure 4.1:

1. For each actor x, order the list of other actors by the time-series function scores ($TimeScore_x(y)$) and Rooted-PageRank scores ($RPRScore_x(y)$), generating the ranked lists
   $LT_x = [a_1, a_2, \ldots, a_k, \ldots, a_{|V|-1}]$ and
   $LR_x = [b_1, b_2, \ldots, b_k, \ldots, b_{|V|-1}]$, respectively, where:
   $TimeScore_x(a_i) \geq TimeScore_x(a_{i+1})$, $i = 1, 2, \ldots, |V| - 2$,
   and $RPRScore_x(b_i) \geq RPRScore_x(b_{i+1})$, $i = 1, 2, \ldots, |V| - 2$.

2. "Filtering step": Given parameter k, truncate list LT_x at position k to generate list $LT_x(k)$. If the truncation position k is within a group of actors who have equal scores, we include all these actors in the filtered list $LT_x(k)$. Hence, we have:

$$LT_x(k) = \begin{cases} [a_1, \ldots, a_k] & \text{if} \quad TimeScore_x(a_k) > TimeScore_x(a_{k+1}) \\ [a_1, \ldots, a_{k+s}] & \text{if} \quad TimeScore_x(a_i) = TimeScore_x(a_{i+1}), \\ & \text{for all} \quad i = k, \ldots, k+s-1 \\ & \text{and} \quad TimeScore_x(a_{k+s}) > TimeScore_x(a_{k+s+1}) \\ [a_1, .., a_{|V|-1}] & \text{if} \quad TimeScore_x(a_i) = TimeScore_x(a_{i+1}), \\ & \text{for all} \quad i = k, \ldots, |V|-2 \end{cases} \tag{4.2.6}$$

3. "Refine step": Re-order $LT_x(k)$ (obtained in step 2) according to the Rooted-PageRank scores (obtained in step 1). For simplicity in the various equations in (4.2.6), we assume that the truncation is actually at position k. The result of the refine step is the list $LTR_x = [a_{\pi(1)}, \ldots, a_{\pi(k)}]$, where π defines a permutation of $1, \ldots, k$ and
$RPRScore_x(a_{\pi(i)}) \geq RPRScore_x(a_{\pi(i+1)})$, $i = 1, \ldots, k-1$.

4. "Detecting the Relations": Given that the ranked list LTR_x obtained in step 3 is $LTR_x = [a_{\pi(1)}, \ldots, a_{\pi(i)}, \ldots, a_{\pi(k)}]$, we detect the rank of actor x^*, who is related to x by a hierarchical tie as follows:

$$rank(x, x^*) = \begin{cases} |\{y : y \in LTR_x \wedge score_x(y) \geq score_x(x^*)\}| & \text{if} \quad x^* \in LTR_x \\ |V| - 1 & \text{if} \quad x^* \notin LTR_x \end{cases} \tag{4.2.7}$$

Intuitively, $rank(x, x^*)$ represents the number of actors who are ranked at least as high as the correct superior of x. In the ideal case, x^* should be ranked first in LTR_x (i.e. $rank(x, x^*) = 1$). In the worst case, x^* may be filtered out from the list during the filtering step. In such a case, we have to consider the full set of actors as candidates for the direct superior.

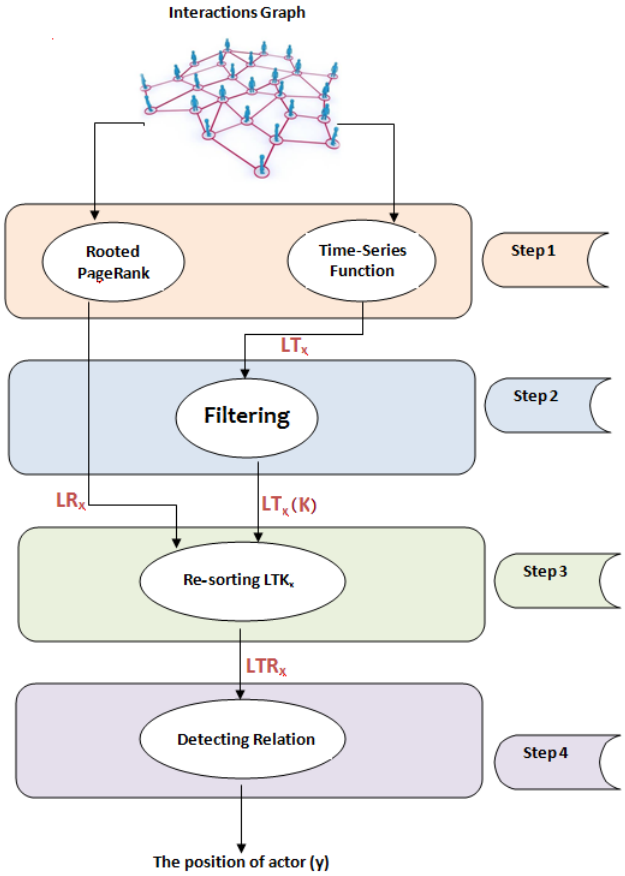

Figure 4.1: Filter and Refine (FiRe) approach.

In order to evaluate the FiRe approach and compare it with both Time-F and RPR, we use the same evaluation function (4.2.5) defined earlier.

Note: In the aforementioned methodology, we use Time-F as a "filter step" and RPR as a "refine step". However, we also evaluated FiRe when using RPR as a "filter step" and Time-F as a "refine step". The results in the latter case were worse than the former. The reason behind the difference in these results is that Time-F performs better than RPR in detecting the superiors. In other words, Time-F ranks superiors higher compared to how RPR ranks them. Using Time-F as a filter step decreases the chance to filter out the superiors when we consider the top k actors in the sorted lists. The full results can be found in Appendix A.

4.3 Results and Analysis

In this section, we first describe the experimental setup for the prediction process. Then we present the significant improvement we obtained over Rooted-PageRank by using Time-F and FiRe.

4.3.1 Experimental Setup

Before running the experiments, we had to decide exactly which features to include in the graph for each dataset.

Enron dataset

As we did in the previous chapter, we excluded all nodes (email-addresses) belonging to people not employed by Enron, since they would only add noise to our analysis. As a result, only emails exchanged between Enron employees were considered. Moreover, we ran the experiments on the unweighted directed graph. In other words, if person u sent m emails to person v, this is reflected in the graph as a single arc $e = (u, v) \in E^c$ where E^c is the set of arcs (pairs) representing the interactions between the actors. For the results of Time-F and FiRe presented in this chapter, each time-slot represents one month. However, we obtained similar patterns of results when we considered weeks instead of months as time slots. The full results can be found in Appendix A.

Co-author dataset

Since the co-author relationship is symmetric, the graph representing this dataset is undirected. Like the Enron dataset, the graph is unweighted: there is only one edge between a pair of authors when they coauthor at least one paper. For Time-F and FiRe, each time-slot represents one year. Due to the large size of the co-author

dataset, we used *htcondor*[1] with 150 nodes in order to run the experiments, which still took 3.5 hours to complete in the worst case.

4.3.2 Time-Function (Time-F) Results

As described earlier, for each actor x, we rank the other actors by the time-scores calculated using Equation 4.2.1. Figure 4.2 and Table 4.1 reveal the results obtained by Time-F in detecting manager-subordinate ties in the Enron dataset. About 35% of manager-subordinate pairs can be detected precisely, that is, where the manager appears first within the subordinate's ranked list. This shows an improvement over RPR which detected only 29.45% correctly. In general, Time-F performs better than RPR in detecting manager-subordinate ties when we look at the top i employees in the subordinate's ranked list where $i \leq 8$ (except for $i = 2$ when RPR is less than 1% better than Time-F).

In the co-author dataset, as shown in Figure 4.3 and Table 4.2, Time-F is better than RPR in detecting advisor-advisee relationships with a significant improvement of 20-30 percentage points in most cases. In 62% of cases, advisors are ranked first in the list of their advisees. This compares to only 39% in the case of RPR.

4.3.3 Filter-and-refine (FiRe) Results

In order to find the best cut-off position k, we tested all possible values of k from 2 to 20. Tables 4.3 and 4.4 list the most interesting FiRe results using different cut-off values k on the Enron and co-author datasets respectively.

In the Enron dataset, about 44% of manager-subordinate pairs can be detected with cut-off $k = 4$. This percentage decreases slightly to 40% when $k = 6$ but with better results for those relations where managers appear in the top 3 or 4 of their subordinate's ranked lists. Figure 4.2 compares between RPR, Time-F and

[1] http://research.cs.wisc.edu/htcondor/

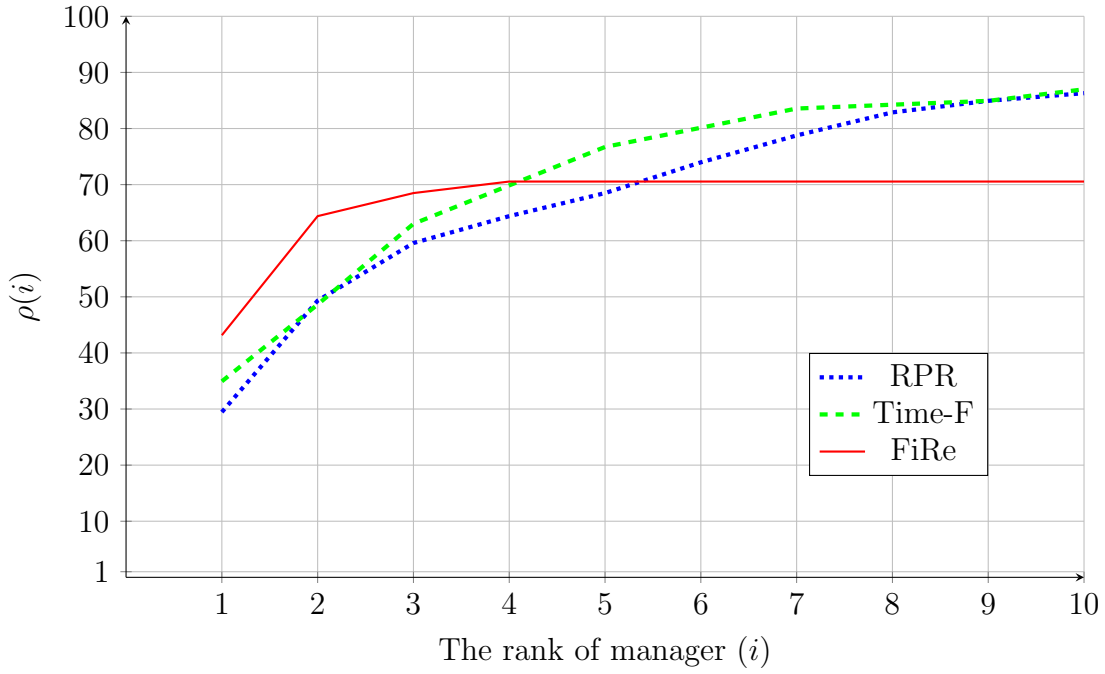

Figure 4.2: Results of RPR, Time-F and FiRe on the Enron dataset. This represents the percentages (y axis) of manager-subordinate relationships in which the managers have rank less or equal to i (x axis).

	$\rho(i)$		
i	Rooted-PageRank	Time-F	FiRe($k = 4$)
1	29.45	34.93	43.15
2	49.31	48.63	64.38
3	59.58	63.01	68.49
4	64.38	69.86	70.54
5	68.49	76.71	70.54
6	73.97	80.13	70.54
7	78.76	83.56	70.54
8	82.87	84.24	70.54
9	84.93	84.93	70.54
10	86.30	86.98	70.54

Table 4.1: Percentage Results of Rooted-PageRank, Time-F and FiRe on Enron dataset.

FiRe. Clearly, FiRe is the best approach with significant improvement in detecting managers who are ranked in the top three of their subordinates' lists. For example, in about 44% of manager-subordinate relations, managers come first in the ranked lists, compared to 34.9% and 29.4% detected by Time-F and RPR respectively.

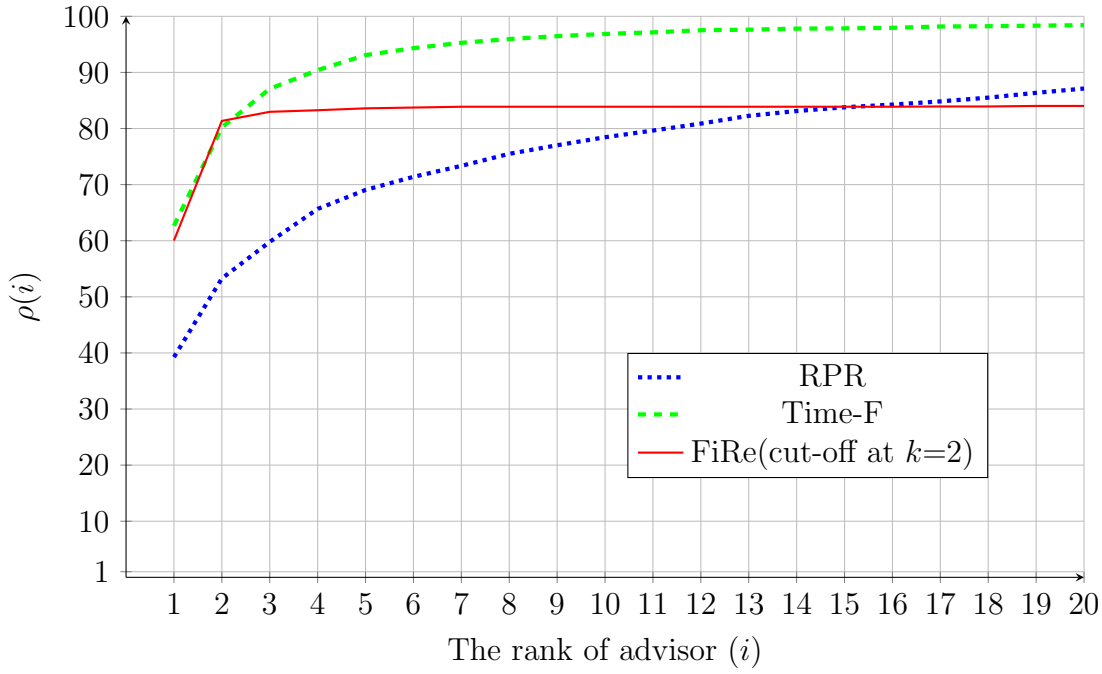

Figure 4.3: Results of R-PR, Time-F and FiRe on the co-author dataset. This represents the percentages (y axis) of advisor-advisee relationships in which the advisors have rank less or equal to i (x axis).

	$\rho(i)$		
i	Rooted-PageRank	Time-F	FiRe($k = 2$)
1	39.27	62.66	60.03
2	53.26	80.20	81.35
3	59.79	87.07	82.97
4	65.66	90.41	83.26
5	69.05	93.08	83.59
6	71.38	94.32	83.73
7	73.34	95.27	83.88
8	75.48	95.94	83.88
9	77.01	96.47	83.88
10	78.44	96.85	83.88

Table 4.2: Percentage Results of Rooted-PageRank, Time-F and FiRe on Co-author dataset.

As shown in Table 4.4, in the co-author dataset, the best results for FiRe are for a cut-off position of $k = 2$, with about 60% of advisor-advisee relationships being detected at rank one, and about 81% at ranks one or two. However the cut-off at $k = 3$ is slightly better at retrieving advisors in the top three of their advisees' lists. It should be noted that by the definition of $LT_x(k)$ in Equation (4.2.6), the cut-off

at k may return a list of more than k actors. This occurs when multiple actors have the same scores around the cut off position, and explains the increase in detected relations for $k = 2$ when we look at the top two (81.35%) compared to the top three authors (82.97%). Comparing the three approaches in Figure 4.3, FiRe and Time-F perform similarly in detecting relations where the advisors come first in the ranked lists. They achieve 60% and 62% respectively. Time-F is preferable to RPR and FiRe in cases where advisors appear in the top three or more authors within the ranked lists.

cut	$\rho(i)$				
off	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
2	39.04	48.63	48.63	48.63	48.63
3	41.78	58.21	63.01	63.01	63.01
4	**43.15**	**64.38**	68.49	70.54	70.54
5	41.09	63.01	72.60	76.02	76.71
6	40.41	64.38	**73.97**	**78.08**	80.13

Table 4.3: FiRe results for Enron dataset using various cut-off values k.

cut	$\rho(i))$				
off	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
2	**60.03**	**81.35**	82.97	83.26	83.59
3	54.60	75.44	**84.97**	86.98	87.88
4	50.50	71.24	80.44	**88.22**	90.08
5	48.30	68.24	77.34	84.78	**90.89**
10	42.53	59.13	67.71	75.05	79.63
15	41.15	56.17	64.13	70.57	74.67

Table 4.4: FiRe results for co-author dataset using various cut-off values k.

In summary, considering the time-dimension of interactions between actors, as in Time-F, improves the results of detecting hierarchical relationships compared to structure-based approaches like RPR. Moreover, in some applications like email networks, hybrid models (FiRe) based on both network structure and the temporal patterns of interactions give further improvement over the pure time-based (Time-F) or structure-based models (RPR).

4.4 Case study

In this section, we consider a real case study from the Enron dataset. The goal of this section is to show how the aforementioned approaches work and to compare their results.

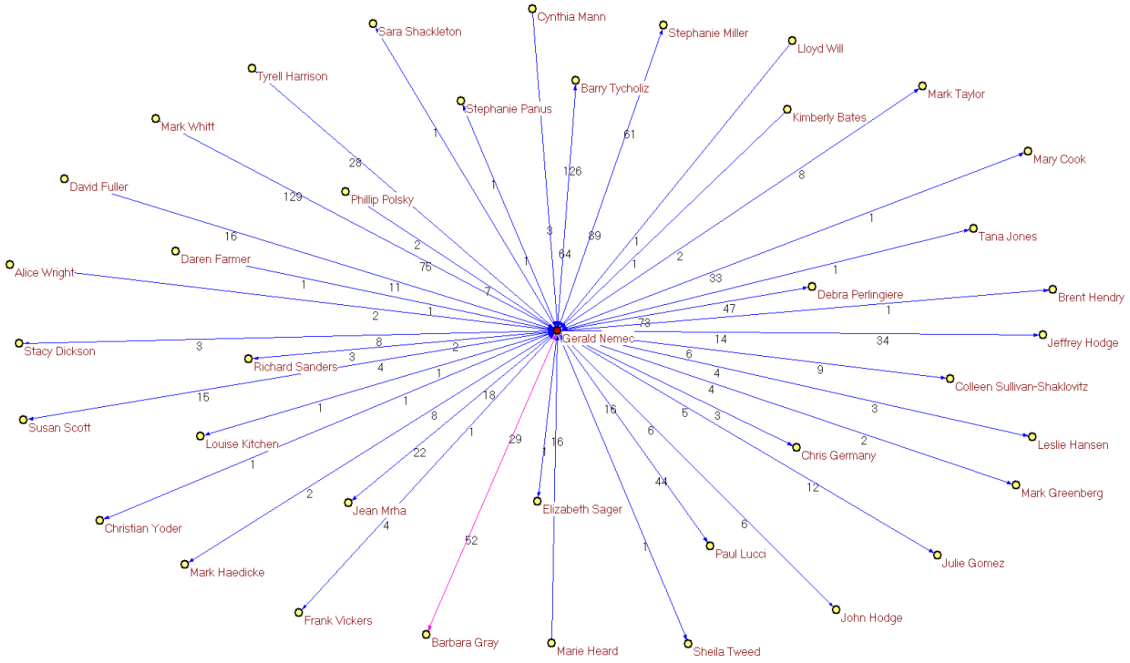

Figure 4.4: Neighbourhood of "Gerald Nemec" in the Enron interaction network.

Due to the size of Enron dataset and to give a clearer view, we only present a sub-interaction graph in Figure 4.4. Figure 4.4 views the interaction graph for the employee "Gerald Nemec" whose direct manager we are trying to detect. Nemec interacted with 38 employees over the period from Jan-2000 to Dec-2001. Each node in the graph represents an employee who sent/received at least one email to/from Nemec. Each edge from node a to node b is labelled by the total number of emails sent from employee a to employee b over the same period. Some of these edges are bidirectional. This is the case for all pairs of employees who sent and received at least one email from each other within the studied period. For readability, only the neighbourhood of "Gerald Nemec" is shown, and even though interactions occurred between neighbours of "Nemec", they are not shown in Figure 4.4.

In the following sections, we present how well Rooted-PageRank, Time-F and FiRe performs in detecting the subordinate-manager relationship between "Gerald Nemec" and "Barbara Gray". The interaction between Nemec and Gray is represented as a pink edge in Figure 4.4.

4.4.1 Rooted-PageRank Results

Although Figure 4.4 shows the weighted version of interaction sub-graph related to "Gerald Nemec", the setting used in this experiment is with the unweighted graph. We ran Rooted-PageRank on the unweighted directed interaction graph of the full dataset where the root node is the one representing "Gerald Nemec". Then, we sorted the list of employees according to the Rooted-PageRank scores.

Table 4.5 lists the top 20 employees in the final sorted list of employees with their Rooted-PageRank scores. "Barbara Gray" who is the direct manager of Nemec came in 10th place.

Pos	**Employee**	*RPRScore*	**Pos**	**Employee**	*RPRScore*
1	Louise Kitchen	0.01999	11	Sara Shackleton	0.012913
2	Jeffrey Hodge	0.01990	12	Elizabeth Sager	0.012701
3	John Lavorato	0.01626	13	Kevin Presto	0.012616
4	Mark Taylor	0.01566	14	Phillip Allen	0.012029
5	Barry Tycholiz	0.01492	15	Mary Cook	0.011864
6	Michael Grigsby	0.01406	16	Stacy Dickson	0.011855
7	Mark Haedicke	0.01323	17	Richard Sanders	0.011780
8	Stephanie Miller	0.01322	18	Debra Perlingiere	0.01161
9	Tana Jones	0.01317	19	Leslie Hansen	0.01123
10	Barbara Gray	0.01310	20	Frank Vickers	0.01114

Table 4.5: The resulting list LR_{Nemec} after applying Rooted-PageRank to detect Gerald Nemec's manager.

4.4.2 Time-F Results

To run this approach, we need to generate the time series of interactions between the subordinate whose manager we are trying to detect (in our case Gerald Nemec), and

each employee he interacted with between Jan-2000 and Dec-2001. One interaction corresponds to any sent or received email in which the subordinate was involved. Also, as mentioned earlier, each time slot represents one month. Table 4.6 shows the time series of 81 interactions between Nemec and his manager, Barbara Gray. Table 4.7 shows the time series of interactions between Nemec and all employees. As defined in Equation (4.2.1), in order to calculate $TimeScore_x(y)$ we need the time series between x and y as well as the time series of interactions between x and all employees.

2000	t_1 Jan	t_2 Feb	t_3 Mar	t_4 Apr	t_5 May	t_6 Jun	t_7 Jul	t_8 Aug	t_9 Sep	t_{10} Oct	t_{11} Nov	t_{12} Dec
	0	0	0	2	0	2	10	6	12	10	5	6
2001	t_{13} Jan	t_{14} Feb	t_{15} Mar	t_{16} Apr	t_{17} May	t_{18} Jun	t_{19} Jul	t_{20} Aug	t_{21} Sep	t_{22} Oct	t_{23} Nov	t_{24} Dec
	3	1	2	1	2	4	5	3	3	3	0	1

Table 4.6: An example of the time series of interactions between Gerald Nemec and his manager Barbara Gray.

2000	t_1 Jan	t_2 Feb	t_3 Mar	t_4 Apr	t_5 May	t_6 Jun	t_7 Jul	t_8 Aug	t_9 Sep	t_{10} Oct	t_{11} Nov	t_{12} Dec
	2	4	6	5	3	12	15	11	28	31	45	47
2001	t_{13} Jan	t_{14} Feb	t_{15} Mar	t_{16} Apr	t_{17} May	t_{18} Jun	t_{19} Jul	t_{20} Aug	t_{21} Sep	t_{22} Oct	t_{23} Nov	t_{24} Dec
	40	39	66	128	102	71	48	68	48	69	111	40

Table 4.7: The time series of interactions between Gerald Nemec and all Enron employees.

Having built all the time series related to Nemec, we calculate $TimeScore_{Nemec}$ for each employee using the function defined in Equation (4.2.1). Then, we generate a ranked list of employees LT_{Nemec} according to their time scores. Table 4.8 lists the top 20 employees in the list LT_{Nemec} with their time scores. The manager "Barbara Gray" appears in the third position.

Pos	Employee	*TimeScore*	Pos	Employee	*TimeScore*
1	Mark Whitt	4.7153	11	Jean Mrha	0.5960
2	Stephanie Miller	4.6232	12	David Fuller	0.4297
3	Barbara Gray	3.2626	13	Mary Cook	0.3649
4	Susan Scott	3.0813	14	Colleen Sullivan	0.3534
5	Barry Tycholiz	2.1215	15	Richard Sanders	0.3214
6	Debra Perlingiere	1.9530	16	Julie Gomez	0.2380
7	Paul Lucci	1.0935	17	Mark Taylor	0.2008
8	John Hodge	0.9979	18	Stacy Dickson	0.1812
9	Tyrell Harrison	0.8744	19	Leslie Hansen	0.1563
10	Jeffrey Hodge	0.6949	20	Marie Heard	0.1441

Table 4.8: The resulting list LT_{Nemec} after applying Time-F to detect Gerald Nemec's manager.

4.4.3 FiRe Results

After we generate two sorted lists of employees LR_{Nemec} and LT_{Nemec} using Rooted-PageRank and Time-F respectively, we can apply the FiRe approach to detect Nemec's manager. First, in the filtering step, we truncate the top four employees from LT_{Nemec} to be re-sorted in the refining step according to Rooted-PageRank scores. Tables 4.9 and 4.10 show the lists obtained in filtering and refining steps respectively. The final resulting list using FiRe shows "Barbara Gray", who is Nemec's direct manager, in the second position (see Table 4.10).

Pos	Employee	*TimeScore*
1	Mark Whitt	4.7153
2	Stephanie Miller	4.6232
3	Barbara Gray	3.2626
4	Susan Scott	3.0813

Table 4.9: The list obtained after the filter step in the FiRe approach to detect Gerald Nemec's manager.

Although FiRe ranks Gray (Nemec's manager) in second rather than first position, it still outperforms both RPR and Time-F which rank Gray in tenth and third positions respectively (see Chart 4.1). So Nemec is among the 64.38% of employees whose managers are ranked within the first two positions by FiRe (see Table 4.1).

Pos	Employee	$RPRScore$
1	Stephanie Miller	0.01322
2	Barbara Gray	0.01310
3	Mark Whitt	0.01070
4	Susan Scott	0.00753

Table 4.10: The final list LTR_{Nemec} obtained after the refine step in the FiRe approach to detect Gerald Nemec's manager.

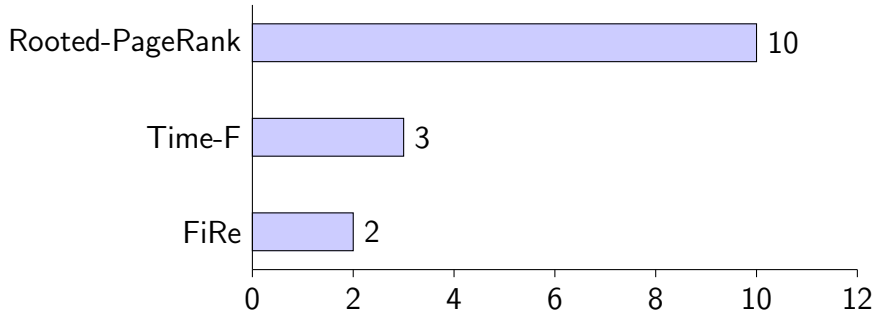

Chart 4.1: The rank of Nemec's manager calculated by Rooted-PageRank, Time-F, and FiRe.

4.5 Summary

In this chapter, we presented our work on inferring implicit hierarchical ties between actors from their online interaction networks. We considered as examples detecting manager-subordinate relations between employees from their exchanged emails and discovering supervision relationships in academia by analysing a network of co-authored papers. Our experiments showed that including temporal patterns of interactions results in considerably better predictions of hierarchical ties than models based on studying network structure alone (like Rooted-PageRank). In some applications, like email networks, heterogeneous models based on both temporal and structural features of interaction perform even better than homogeneous models. We compare the computational costs of RPR with Time-F and FiRe in Section 5.3.5.

In the next chapter, we develop a time-sensitive model based on Rooted-PageRank which captures the development of the RPR scores over time in order to improve the inference of hierarchical relationships between actors.

Chapter 5

Detecting Hierarchical Relations using Time-Sensitive Rooted-PageRank

5.1 Overview

In the previous chapters, we addressed the problem of detecting hierarchical ties between a group of actors in a social network by analysing the structure of the interaction network using Rooted-PageRank algorithm (RPR) [133]. We also proposed two time-based methods, namely Time-F and FiRe, which study the interaction patterns between the actors over time.

In this chapter, we propose another novel method, *Time-sensitive Rooted-PageRank* (*T-RPR*), to capture the RPR scores dynamics of the actors over time. The method proves to be more effective in detecting hierarchical ties, especially when the period over which the interactions occur is long enough.

The **contributions** of this chapter include:

- A novel time-sensitive method (T-RPR) which builds upon the static Rooted-PageRank (S-RPR) explained in chapter 3, and captures how the structure of the interaction network changes over time.

- Two approaches for aggregating scores from each of the time slots over which T-RPR is run, one based on a simple weighted average and the other based on voting.

- An extensive experimental evaluation of the performance of these methods in terms of recall on two large real datasets, the Enron e-mail network and a co-authorship network. Our experiments show that time-sensitive Rooted-PageRank (T-RPR) achieves considerably better results than the competitors static Rooted-PageRank (S-RPR), Time-F and FiRe. For example, in the Enron network, T-RPR detects up to 58% of manager-subordinate relationships, compared to only 34% by S-RPR, while in the co-author network it detects about 68.8% of PhD advisor-advisee relationships, a significant improvement over the 39.5% achieved by S-RPR.

The results presented in this chapter have been published previously in [56, 57].

5.2 Time-Sensitive Rooted-PageRank

We continue our investigation of whether "time matters" in detecting hierarchical social relationships; in other words, whether significant improvements in detecting hierarchical ties can be obtained by taking into account the temporal aspects of the interactions. Here, we adapt the static Rooted-PageRank (S-RPR), as described in Chapter 3, and introduce *Time-Sensitive Rooted Pagerank (T-RPR)*, which captures how the ranking scores of the actors change over time. The proposed method consists of three parts: time segmentation, ranking, and rank aggregation.

5.2.1 Time Segmentation

Instead of the single interaction graph used in S-RPR, we now consider a number of interaction graphs, each corresponding to a period of time. Let $T = [t_1, t_m]$ be the total time period over which interactions have taken place, starting at time t_1 and ending at time t_m. First, T is divided into n equal-sized non-overlapping time slots $\{T_1, T_2, \ldots, T_n\}$, with $T_j = [t_{jk}, t_{jl}]$, $\forall j \in [1, n]$, such that $t_{jl} - t_{jk} = d$, $\forall j$, where $d \in \mathbb{Z}^+$ is the size of the time segments. Observe that a time slot can be any time unit (e.g., day, fortnight, month, or year) depending on the application. Next, we define an interaction graph for each time slot.

Definition (Time-Interaction Graph): A *time-interaction graph* is defined as $G_I^k = (V_k, E_k^c, W)$, where $V_k \subseteq V$ is the set of actors who interacted with at least one other actor within time slot T_k, $E_k^c \subseteq V_k \times V_k$ is the set of edges (directed or undirected) corresponding to the interactions between the set of actors V_k which took place within T_k, and W is the vector of edge weights.

Finally, a set of time-interaction graphs $\mathcal{G}_I = \{G_I^1, \ldots, G_I^n\}$ is produced for the n time slots. The next task is to rank the nodes in each graph.

5.2.2 Segment-based Ranking

For each time-slot T_k and each actor $x \in V$, we run RPR on the corresponding time-interaction graph $G_I^k = (V_k, E_k, W)$. Let $score_{x,k}(v_i)$ denote the RPR score of actor v_i when x is used as root on G_I^k. This results in a list of actors sorted in descending order with respect to their RPR scores at time slot k:

$$L(x)_k = [v_1, v_2, \ldots, v_N] \ ,$$

where $N = |V_k| - 1$, $V_k \backslash \{x\} = \{v_1, v_2, \ldots, v_N\}$, and $score_{x,k}(v_i) \geq score_{x,k}(v_{i+1})$ for $i = 1, \ldots, N-1$.

The rankings obtained over the n time-slots are aggregated for each root actor x and all remaining actors $v_i \in V$, resulting in an aggregate score $aggScore_x(v_i)$. Finally, the aggregate scores are sorted in descending order resulting in the following aggregate list of actor ranks:

$$L(x) = [v_1, v_2, \ldots, v_M] \ ,$$

where $M = |V|-1$, $V/\{x\} = \{v_1, v_2, \ldots, v_M\}$, and $aggScore_x(v_i) \geq aggScore_x(v_{i+1})$, for $i = 1, \ldots, M-1$. More details on the aggregation techniques are given below.

Finally, as in S-RPR, the hierarchy graph G_H is inferred from $L(x)$ by assigning to each node $x \in V$ one of the candidate managers that ranked high in $L(x)$, e.g., within the top-K places, for some K.

5.2.3 Rank Aggregation

We explored two rank aggregation techniques, one based on averaging and one based on voting.

Average-based Time-sensitive RPR (AT-RPR)

In this approach, the ranking in $L(x)$ is based on a weighted average of the individual RPR scores over all time-slots. We define a set of weights $\Omega = \{\omega_1, \ldots, \omega_n\}$, where ω_k is the weight assigned to time slot T_k. Each actor $y \in L(x)$ is ranked according to the obtained scores over all time-slots:

$$aggScore_x(y) = 1/n \sum_{k=1}^{n} \omega_k \times score_{x,k}(y) \ . \tag{5.2.1}$$

where:

- x is the actor whose superior we are trying to detect,
- n is the total number of slots,

- $score_{x,k}(y)$ is the Rooted-PageRank score obtained for y within the time-slot t_k when x is the root node, and
- ω_k is the weight associated with time slot t_k.

Assigning the values in Ω is application-dependent. For example, if the interactions between actors and their superiors are distributed regularly over the whole period T, then all weights can be equal. On the other hand, the interactions between actors and their superiors may be more intensive in earlier or later time-slots. An example of the former case is when detecting PhD advisor-advisee relationships in a co-author network; higher weights are given to scores in early time-slots when the advisees are expected to publish more papers with their advisors, decreasing in later time-slots.

Vote-based Time-sensitive RPR (VT-RPR)

An alternative approach to rank aggregation is to assign candidate actors with votes at each time-slot T_k based on their rank in that slot. The final rank of an actor is determined according to the total number of votes they win over all time-slots.

More precisely, given $L(x)_k$ at slot T_k, a vote is assigned to actor $y \in V \setminus \{x\}$ - if y appears among the first c actors in $L(x)_k$. We call c the *vote-based cut-off*. Let $pos(L(x)_k, y)$ denote the position of y in $L(x)_k$. The total number of votes obtained by each candidate y is then defined as:

$$aggScore_x(y) = \sum_{k=1}^{n} \omega_k \cdot vote_{x,k,c}(y) \ , \tag{5.2.2}$$

where:

$$vote_{x,k,c}(y) = \begin{cases} 1 & \text{if } \ pos(L(x)_k, y) \leq c \\ 0 & \text{otherwise} \end{cases}$$

and ω_k is the weight of time slot T_k, which is set depending on the application, as suggested in the previous subsection.

5.3 Results and Analysis

We evaluated the AT-RPR and VT-RPR in terms of recall on the same two datasets we used before: the Enron email dataset and a co-authorship network. We first explain the approach used to evaluate the results. Then, for each dataset tested, we propose the experimental settings and discuss the results obtained.

5.3.1 Evaluation Methodology

To evaluate the performance of the two methods, we compute for each subordinate/advisee x the *rank* of their correct superior/advisor x^* in $L(x)$:

$$rank(x, x^*) = |\{y : y \in L(x) \wedge score_x(y) \geq score_x(x^*)\}| \tag{5.3.3}$$

Hence, the rank of the manager x^* of x is the number of actors in $L(x)$ who have an RPR score greater than or equal to the score of x^*.

Finally, given a threshold i, we can define the *overall rank* $\rho(i)$ of V as the percentage of actors with rank at most i over all hierarchical relations that exist in G_H:

$$\rho(i) = \frac{|\{x : rank(x, x^*) \leq i\}|}{|E^s|} \times 100 \tag{5.3.4}$$

5.3.2 Enron Results

Experimental Settings

We explored two versions of the Enron interaction graph: *directed*, where a directed edge exists from employee u to v if u sent at least one email to v; *undirected*, where any interaction (sent or received email) between u and v is represented as an edge between them.

Also, we tried the unweighted and weighted versions of the interaction graph. In the former case, all edge weights are 1. However, in the latter case, the weight of the edge e between node x and y in the interaction graph within the time period t is determined by the equation:

$$w_e^t = \frac{count(x, y)}{count(x)} \tag{5.3.5}$$

where:

- $count(x, y)$ is the total number of emails sent from x to y within t if the interaction graph is directed. In the case of undirected graph, this represents the total number of interactions (sent/received emails) between x and y over the same period t.
- $count(x)$ is the total number of emails sent from x to all other employees within t when the graph is directed. In the case of undirected graph, this is the total number of interactions (sent/received emails) occurring during the period t and in which employee x is involved.

According to the aforementioned two categories of settings, we evaluated our approaches on the interaction graphs with the following features:

1. undirected, unweighted
2. undirected, weighted
3. directed, unweighted
4. directed, weighted

The weights used in each interaction graph are calculated on the basis of the time-slot to which the interaction graph relates. In the case of the undirected-weighted graph, any edge between two employees x and y should have two weights, the weight w_{xy} from x to y and the weight w_{yx} from y to x. Therefore, we converted the undirected-weighted graph to a directed-weighted graph where each edge e between x and y is replaced by an edge e_{xy} from x to y weighted by w_{xy}, and an edge e_{yx} from y to x weighted by w_{yx}.

For the T-RPR approach, each time-slot represents one month, giving 24 time-slots in total. In addition, since we expect to have regular interactions between a subordinate and their manager over the whole time period, each weight ω_k ($k = 1, \ldots, n$) in the aggregation functions given in Equations (5.2.1) and (5.2.2) was set to 1.

Experimental results using a *directed* graph

The experimental results of AT-RPR and VT-RPR against S-RPR, Time-F and FiRe using a directed graph representation are shown in Figure 5.1. This includes the results of (a) the unweighted version and (b) the weighted version. Using this figure in combination with Tables 5.1 and 5.2 we can make several observations.

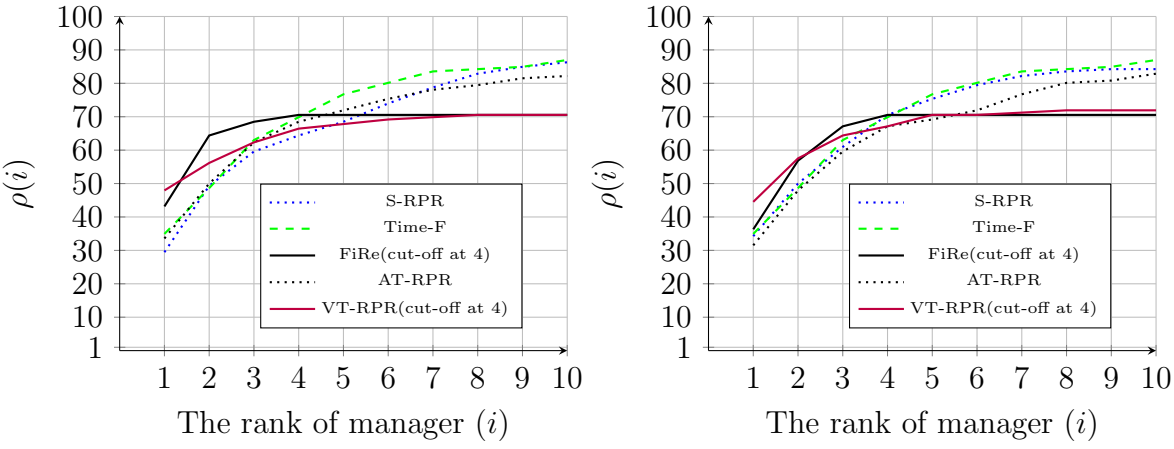

Figure 5.1: Results using S-RPR, Time-F, Fire and T-RPR (both aggregation strategies) on Enron using (a) the directed unweighted interaction graph and (b) the directed weighted interaction graph. This represents the percentages (y axis) of manager-subordinate relationships in which the managers have rank less or equal to i (x axis).

Firstly, S-RPR has nearly identical performance in both weighted and unweighted versions. For example, S-RPR can detect 49.31% of managers within the top two positions in their subordinates' sorted lists using the unweighted graph. This percentage increases to only 50% when the weighted graph is used. As explained previously, Time-F works by ranking the actors according to their scores calculated

	$\rho(i)$				
i	S-RPR	Time-F	FiRe($k=4$)	AT-RPR	VT-RPR($k=4$)
1	29.45	34.93	43.15	33.56	47.94
2	49.31	48.63	64.38	50	56.16
3	59.58	63.01	68.49	62.32	62.32
4	64.38	69.86	70.54	68.49	66.43
5	68.49	76.71	70.54	71.91	67.80
6	73.97	80.13	70.54	75.34	69.17
7	78.76	83.56	70.54	78.08	69.86
8	82.87	84.24	70.54	79.45	70.54

Table 5.1: Results on directed unweighted Enron graph.

	$\rho(i)$				
i	S-RPR	Time-F	FiRe($k=4$)	AT-RPR	VT-RPR($k=4$)
1	34.24	34.93	36.30	31.50	44.52
2	50	48.63	56.84	47.94	57.53
3	60.95	63.01	67.12	59.58	64.38
4	70.54	69.86	70.54	67.12	67.12
5	75.34	76.71	70.54	69.17	70.54
6	79.45	80.13	70.54	71.91	70.54
7	82.19	83.56	70.54	76.71	71.23
8	83.56	84.24	70.54	80.13	71.91

Table 5.2: Results on directed weighted Enron graph.

cut	$\rho(i)$				
off	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
2	39.72	52.05	56.84	60.27	62.32
3	43.83	54.79	62.32	67.12	67.80
4	47.94	56.16	62.32	66.43	67.80
5	44.52	58.90	63.01	67.12	67.80
6	43.15	58.21	63.69	65.75	67.80
7	41.09	59.58	63.69	65.75	67.12

Table 5.3: VT-RPR results on Enron dataset using vote cut-off $c = 2$–7 with *directed* and *unweighted* interaction graph.

by the time function. Therefore, we compare the same results obtained by Time-F with both the results of using weighted and unweighted graphs. Regarding the FiRe approach, using the unweighted graph returns better results than those obtained when the weighted graph is used. For example, more than 43% of managers are ranked first in the sorted lists compared to only 36.3% using the weighted graph.

cut off	$\rho(i)$				
	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
2	38.35	54.10	57.53	60.95	61.64
3	39.72	60.27	66.43	68.49	69.17
4	44.52	57.53	64.38	67.12	70.54
5	41.09	55.47	62.32	66.43	68.49
6	39.04	57.53	64.38	65.75	68.49
7	39.72	57.53	64.38	66.43	67.80

Table 5.4: VT-RPR results on Enron dataset using vote cut-off $c = 2$–7 with *directed* and *weighted* interaction graph.

In AT-RPR, the results obtained using the weighted and unweighted graphs are similar. For VT-RPR, we consider the best value for the voting cut-off. In Tables 5.3 and 5.4, we can see how different values of the voting cut-off affect the performance percentages of VT-RPR for the unweighted and weighted graphs respectively. If we consider the managers who come first in the sorted lists, the best cut-off for both cases (weighted and unweighted) is $c = 4$. However, VT-RPR performs slightly better using the unweighted graph rather than the weighted graph. In addition, VT-RPR (with cut-off at 4) is preferable to AT-RPR in both settings. This is clear for the managers who appear in the first or second positions within their subordinates' sorted lists. For example, in the unweighted graph VT-RPR detects 47.94% of managers in the first position in the sorted lists compared to only 33.56% detected by AT-RPR.

In the directed graph, when considering the case of managers who are detected at the top of the sorted list, the best results are obtained using VT-RPR in the unweighted setting. On the other hand, the FiRe approach is substantially better for managers appearing within one of the second, third or fourth ranks in the sorted lists. In general, the results of the unweighted settings are better than those obtained in the weighted settings when using the directed interaction graph in our methods.

Experimental results using an *undirected* graph

Figure 5.2 with Tables 5.5 and 5.6 show the results obtained when the interaction graph is *undirected*.

S-RPR works considerably better on the *weighted* graph than the *unweighted* graph when the interactions are undirected. This is clear for managers who are ranked first in the sorted lists. About 27% of the managers are ranked first using the weighted graph compared to only 8.9% using the unweighted graph. Regarding Time-F, the same results are obtained with both the weighted and unweighted graphs.

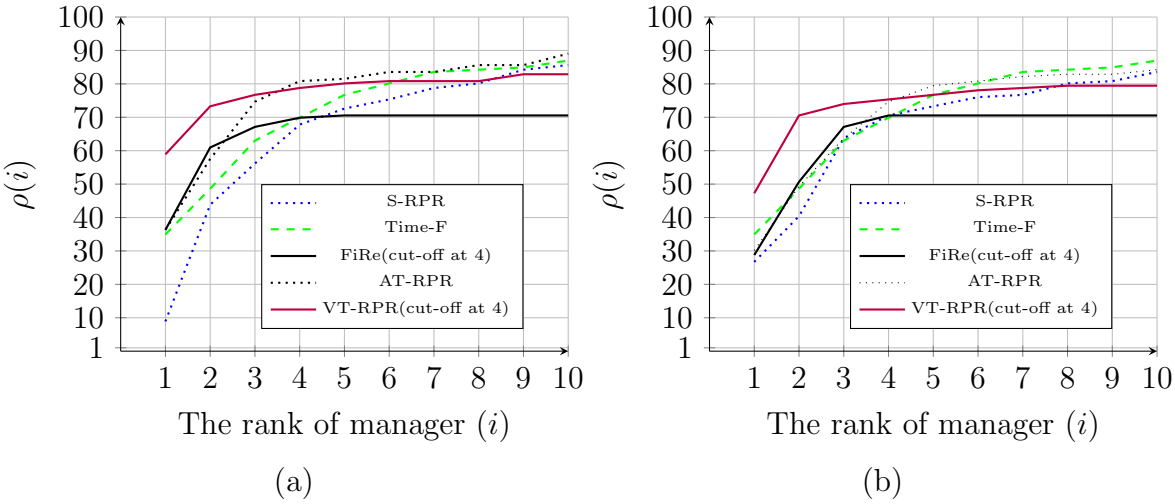

Figure 5.2: Results using S-RPR, Time-F, Fire and T-RPR (both aggregation strategies) on Enron using (a) the undirected unweighted interaction graph and (b) the undirected weighted interaction graph. This represents the percentages (y axis) of manager-subordinate relationships in which the managers have rank less or equal to i (x axis).

The picture changes with FiRe, AT-RPR and VT-RPR. In all these approaches, using the unweighted version of the undirected graph is preferable to using the weighted version. In the FiRe approach, with a cut-off at 4, the percentage of detected managers who appear within the top two positions improved by 8-10 points when the unweighted graph is used. For example, about 61% of managers have one of the two highest scores using the unweighted graph compared to only 50.6% when the weights are considered.

	$\rho(i)$				
i	S-RPR	Time-F	FiRe($k=4$)	AT-RPR	VT-RPR($k=4$)
1	8.90	34.93	36.30	36.30	58.90
2	43.83	48.63	60.95	57.53	73.28
3	56.16	63.01	67.12	74.65	76.71
4	67.80	69.86	69.86	80.82	78.76
5	72.60	76.71	70.54	81.50	80.13
6	75.34	80.13	70.54	83.56	80.82
7	78.76	83.56	70.54	83.56	80.82
8	80.13	84.24	70.54	85.61	80.82

Table 5.5: Results on the undirected unweighted Enron graph.

	$\rho(i)$				
i	S-RPR	Time-F	FiRe($k=4$)	AT-RPR	VT-RPR($k=4$)
1	26.71	34.93	28.76	30.13	47.26
2	40.41	48.63	50.68	49.31	70.54
3	63.69	63.01	67.12	63.69	73.97
4	70.54	69.86	70.54	74.65	75.34
5	73.28	76.71	70.54	79.45	76.71
6	76.02	80.13	70.54	80.82	78.08
7	76.71	83.56	70.54	82.19	78.76
8	80.13	84.24	70.54	82.87	79.45

Table 5.6: Results on the undirected weighted Enron graph.

Similarly, both AT-RPR and VT-RPR give better results using the unweighted graph. However, VT-RPR is substantially more effective in detecting manager-subordinate relationships compared to AT-RPR and all other approaches using either the weighted or unweighted graph. For example, the percentage of managers who are ranked first within their subordinates' sorted lists, does not exceed 37% in the best case using one of the four approaches: S-RPR, Time-F, FiRe and AT-RPR. On the other hand, VT-RPR can detect about 59% and 50% of the managers in the first position using the unweighted and weighted graph, respectively. Moreover, VT-RPR can infer more than 73% of relationships using the unweighted graph and by looking at only the first two employees in each sorted list. This gives an improvement over the 60% of detected relationships by the FiRe approach which is the best

among the other three approaches when we consider the top two employees in each sorted list.

cut off	$\rho(i)$				
	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
2	52.73	66.43	73.97	76.71	78.76
3	58.21	71.91	76.02	78.08	79.45
4	58.90	73.28	76.71	78.76	80.13
5	57.53	73.28	76.02	77.39	80.13
6	54.79	72.60	75.34	78.08	80.82

Table 5.7: VT-RPR results on Enron dataset using vote cut-off 2–6 with *undirected* and *unweighted* interaction graph.

cut off	$\rho(i)$				
	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
2	47.26	68.49	73.28	76.02	76.71
3	45.89	68.49	73.28	76.02	76.71
4	47.26	70.54	73.97	75.34	76.71
5	50	69.17	71.91	73.97	75.34
6	47.94	65.06	73.28	75.34	76.71

Table 5.8: VT-RPR results on Enron dataset using vote cut-off 2–6 with *undirected* and *weighted* interaction graph.

To find the best possible VT-RPR results, we tested a range of cut-off positions as shown in Tables 5.7 and 5.8 for the unweighted and weighted graph, respectively. The best cut-off position in both cases is $c = 4$ when we look for the managers within top three ranks ($i \leq 3$) of each sorted list (an exception is the weighted graph for the case of managers who ranked first, where the best cut-off is at 5 with 50% of managers detected).

In general, similarly to the directed graph, the results obtained using the unweighted graph are better than those obtained using the weighted version when the interaction graph is *undirected*.

Undirected graph vs. directed graph

As we have seen, generally in both directed and undirected interaction graphs, results for the unweighted version are better than those for the weighted version. This

finding indicates that it is not necessarily the case that a subordinate exchanges the highest fraction of interactions with his/her manager compared to other employees. In this section, we compare the results obtained by using the undirected unweighted graph against those of the directed unweighted graph.

Relating to S-RPR, this approach performs considerably better for the directed graph. This becomes clear when we consider the number of managers ranked first in a subordinate's ranked list. About 30% of the managers are ranked first when using a directed graph compared to only 9% for the undirected graph. Similarly, relating to the FiRe approach in which S-RPR is used as a refine step, using the directed interaction graph returns better results compared to those obtained in the case of the undirected graph. For example, FiRe detects about 43% and 36% of the managers come first within the sorted list using the directed and undirected settings, respectively.

On the other hand, this picture changes when we consider the time dimension in T-RPR. Both AT-RPR and VT-RPR give better results on the undirected graph and especially when using vote-based aggregation. This finding suggests that the volume of email matters more than direction for detecting hierarchical ties in a manager-subordinate setting. A possible explanation is that employees may have similar communication patterns with respect to the fraction of sent vs. received emails when they communicate with other employees and also when they communicate with their manager. However, the volume of the email traffic as a whole can be a more distinctive feature of the underlying hierarchical tie.

5.3.3 Co-author Results

Experimental Settings

For the purposes of our study, we excluded all single-author papers as well as papers without a publication date. Due to the symmetric nature of the co-author relation-

ship, the interaction graph representing this dataset is undirected. Once again, each edge weight was set to 1. For the T-RPR approach, we defined 45 time-slots, one per publication year. Moreover, the weights used by both aggregation methods were defined for each $aggScore_x(y)$, as $\omega_k = 1 - \frac{(slot_k - slot_1)}{(slot_n - slot_1)}$, where $(slot_k - slot_1)$ is the number of time-slots (years) between time-slot k and the slot in which the first paper was co-authored by x, and $(slot_n - slot_1)$ is the total number of time-slots between the first and last papers co-authored by x. We defined the weights in this way since we expect more intensive interactions between an advisee and their advisor in the early stages of the advisee's publication activity. Therefore, higher weights are given to early years. For each advisee whose advisor we wish to detect, we only consider the period between his/her first and last publication.

Due to the large size of the co-author dataset, we used *htcondor*[1] with 160 nodes in order to run the experiments, which still took 4 hours to complete in the worst case.

Experimental Results

Figure 5.3 and Table 5.9 present the performance of S-RPR, Time-F, FiRe, AT-RPR and VT-RPR for the co-author dataset. Clearly, the results for both AT-RPR and VT-RPR, are substantially better than those for S-RPR. For example, for more than 65% of the advisees, both AT-RPR and VT-RPR correctly infer their advisor as the top-ranked co-author. This gives a remarkable improvement over the results of S-RPR which only detects 39.27% of advisors correctly.

On the other hand, the results of AT-RPR are 3–7 percentage points better than VT-RPR. For instance, more than 95% and 90% of advisor-advisee relationships can be detected within the top 7 authors by AT-RPR and VT-RPR respectively. Table 5.10 shows that the best results for VT-RPR are with voting cut-off at 1.

[1] http://research.cs.wisc.edu/htcondor/

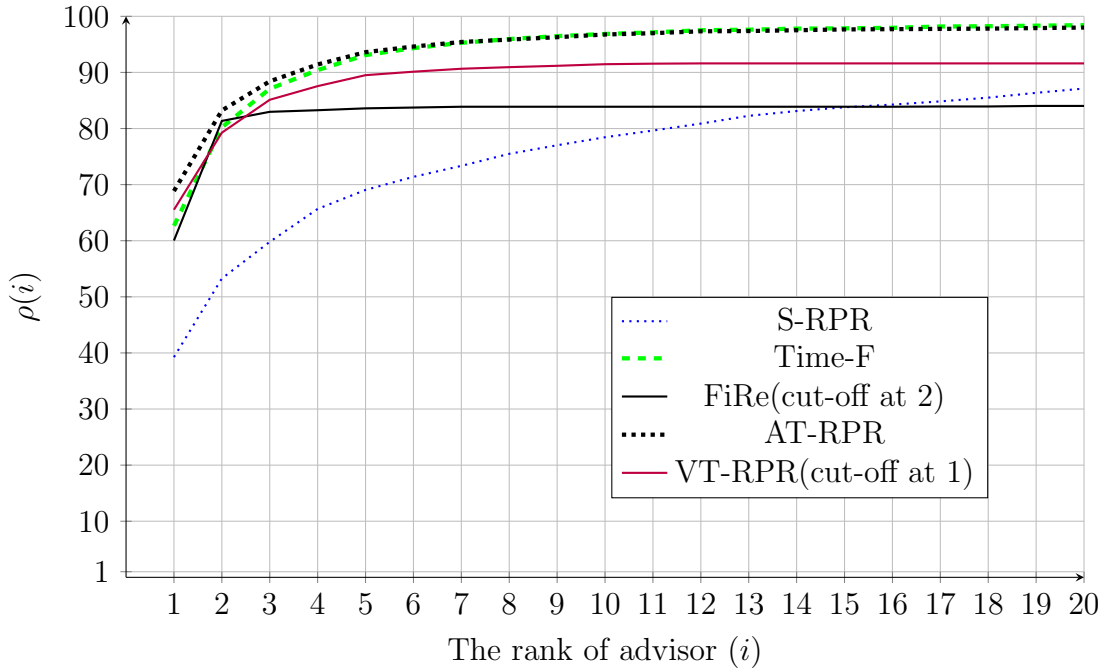

Figure 5.3: Results for S-RPR, T-RPR (both aggregations) on the co-author dataset. This represents the percentages (y axis) of advisor-advisee relationships in which the advisors have rank less or equal to i (x axis).

	$\rho(i)$				
i	S-RPR	Time-F	FiRe($k=2$)	AT-RPR	VT-RP($k=1$)
1	39.27	62.66	60.03	68.86	65.52
2	53.26	80.20	81.35	83.21	79.25
3	59.79	87.07	82.97	88.41	85.12
4	65.66	90.41	83.26	91.41	87.55
5	69.05	93.08	83.59	93.60	89.50
6	71.38	94.32	83.73	94.61	90.12
7	73.34	95.27	83.88	95.42	90.65
8	75.48	95.94	83.88	95.85	90.93
9	77.01	96.47	83.88	96.28	91.17
10	78.44	96.85	83.88	96.75	91.46
11	79.63	97.13	83.88	96.99	91.55
12	80.87	97.52	83.88	97.32	91.60

Table 5.9: Methods applied to the undirected unweighted co-author graph.

5.3.4 Main Findings

For both the Enron and co-author datasets, the time-sensitive methods AT-RPR and VT-RPR are significantly better than S-RPR. This demonstrates that time matters

cut off	$\rho(i)$				
	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
1	**65.52**	**79.25**	**85.12**	**87.55**	**89.50**
2	60.80	78.54	84.64	88.03	89.93
3	56.84	75.58	82.30	85.78	87.88
4	53.55	72.72	79.97	83.83	86.17
5	50.69	69.48	77.25	81.78	84.21

Table 5.10: VT-RPR results for co-author using vote cut-off $c = 1$–5.

when detecting hierarchical relationships in social networks. However, AT-RPR and VT-RPR perform differently on each dataset, with VT-RPR being more effective in detecting subordinate-manager relationships in the Enron data and AT-RPR being slightly better in detecting advisee-advisor relationships in the co-author network.

One interpretation of these results is that, when the interactions between actors and their superiors extend over many time-slots and the number of interactions is large, then VT-RPR is more appropriate. An example of this is the Enron dataset, where the interactions occur over 24 time-slots. On the other hand, when the interactions with the superior are intensive within a few time-slots and the number of interactions is small, AT-RPR is preferable to VT-RPR. This is the case for the co-author dataset where usually an advisee publishes papers with their advisor within only 4–5 time-slots while the advisee is completing their PhD. When compared to the Time-F and FiRe approaches of Chapters 3 and 4, AT-RPR and VT-RPR prove to be more effective in detecting hierarchical ties.

5.3.5 Computational Cost

In this section, we discuss the computational costs of our proposed approaches in terms of the run-time required to complete the experiments. Figures 5.4 and 5.5 show the execution time in minutes of each approach on the Enron and co-author datasets respectively. This represents the average time each approach takes to detect one relationship.

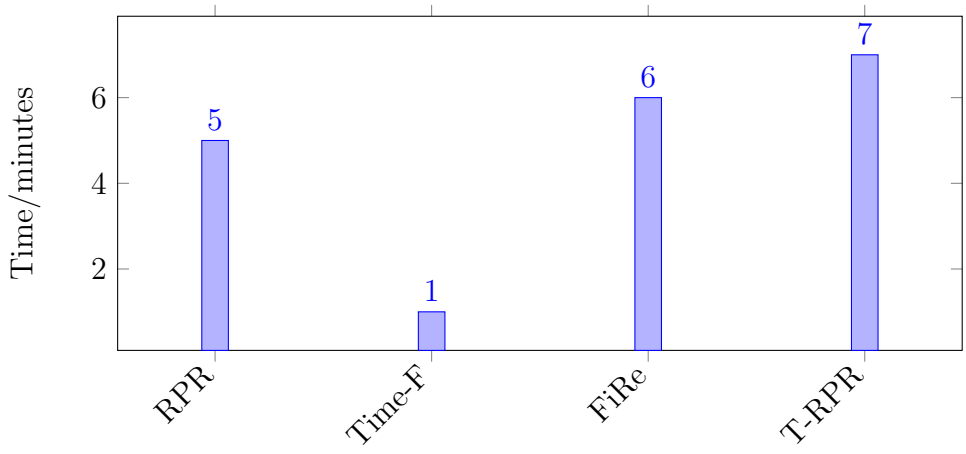

Figure 5.4: The execution time to run each method on the Enron dataset.

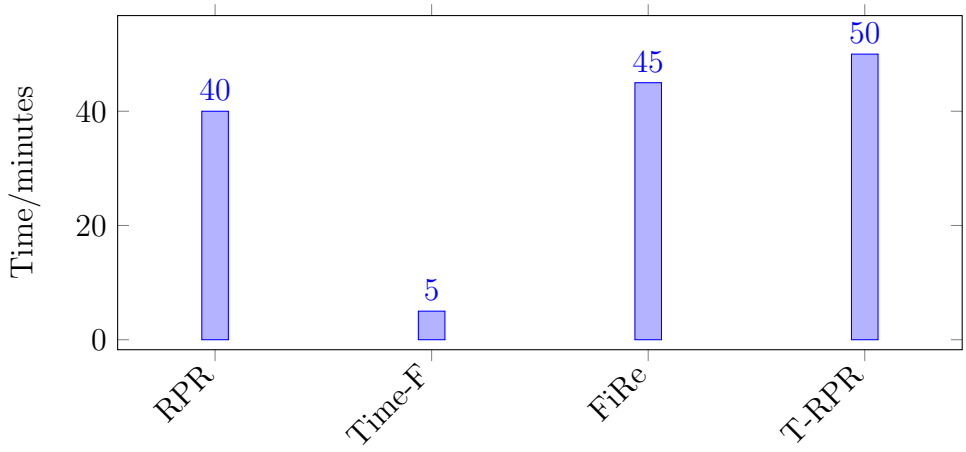

Figure 5.5: The execution time to run each method on the co-author dataset.

As mentioned previously, due to the large size of the co-author dataset, we used *htcondor*[2] to run the experiments. We used 25 machines, each with an Intel i7 CPU with 8 or 16 GB RAM. Each CPU consists of a number of cores (4 or 8). The RAM of each machine is distributed equally among the cores. For example, given a 16GB machine with 8 cores, each core has 2 GB. Each core executes one submitted job. In our case, each job is one execution of RPR, FiRe, Time-F or T-RPR.

Both Enron and co-author datasets show similar patterns in terms of execution time. Although T-RPR was the best method in detecting hierarchical relationships, it takes longer to execute compared to RPR, FiRe and Time-F. For example,

[2]http://research.cs.wisc.edu/htcondor/

it takes 50 minutes to detect one advisor-advisee relationship using T-RPR compared to only 40, 5 and 45 minutes using RPR, Time-F and FiRe respectively. The fastest method among all approaches was Time-F, taking 1 and 5 minutes to detect manager-subordinate and advisor-advisee relationships respectively.

In conclusion, T-RPR is the best method when greatest precision is required irrespective of execution time. On the other hand, when execution time is an important consideration, then Time-F is the best method.

5.3.6 Case Study

Continuing with the case study proposed in the previous chapter, we show in this section the results of using T-RPR in detecting the manager of employee "Gerald Nemec" (namely, "Barbara Gray"). We explain the results using the two aggregation approaches, namely AT-RPR and VT-RPR.

As shown in Section 5.3.2, the best results of T-RPR were obtained using an ***undirected*** and ***unweighted*** interaction graph. In this section, we explain the performance of T-RPR on our case study using these experimental settings.

The number of emails exchanged between "Nemec" and each other employee over two years is shown in Table 5.11. Each row in the table represents one employee and each column represents one month. We only include the employees who have exchanged at least one email with "Nemec" over two years.

Applying T-RPR

T-RPR starts by running Rooted-PageRank n times, where n is the total number of months (i.e., 24). In each run, we use the node representing "Nemec" as the root node. Moreover, in each run, we use the interaction graph of a different month. This generates n lists of employees sorted by the Rooted-PageRank scores. Tables 5.12 to 5.16 show the top 20 employees in each sorted list. The manager (Barbara Grey) is highlighted in each list.

Employee	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	total
Barbara Gray	0	0	0	2	0	2	10	6	12	10	5	6	3	1	2	1	2	4	5	3	3	3	0	1	81
Marie Heard	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	16
Cynthia Mann	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	3
Mary Cook	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	31	2	34
Jeffrey Hodge	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	12	3	1	3	9	7	8	2	48
Stacy Dickson	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	1	0	1	3	0	11
Jean Mrha	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	1	0	0	0	11	1	0	15	9	40
Debra Perlingiere	0	0	0	0	0	0	1	2	1	1	0	1	1	1	5	12	12	9	8	18	16	12	18	2	120
Susan Scott	0	3	5	2	2	2	1	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19
Stephanie Miller	1	1	1	0	0	0	3	0	0	0	0	7	28	7	17	17	3	9	9	16	1	8	8	12	148
Mark Whitt	0	0	0	0	0	5	0	5	6	17	18	28	3	20	20	13	19	5	4	0	8	26	4	2	204
Richard Sanders	0	0	0	0	0	3	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
Paul Lucci	0	0	0	0	0	0	0	0	4	4	5	5	4	1	5	3	9	1	1	2	1	10	4	1	60
Colleen Sullivan	0	0	0	0	0	0	0	0	0	2	13	0	0	0	0	0	0	0	0	0	0	0	0	0	15
David Fuller	0	0	0	0	0	0	0	0	0	1	3	0	0	0	0	0	2	0	5	4	2	3	7	0	27
Tyrell Harrison	0	0	0	0	0	0	0	0	1	0	2	0	0	3	14	24	2	0	4	1	5	6	1	0	63
John Hodge	1	0	0	0	1	0	0	0	0	0	2	0	0	2	3	3	0	0	0	0	0	0	0	0	12
Leslie Hansen	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	2	7
Julie Gomez	0	0	0	0	0	0	0	0	0	0	1	0	3	0	4	4	5	0	0	0	0	0	0	0	17
Mark Greenberg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	4	0	0	1	0	6
Mark Taylor	0	0	0	0	0	0	0	0	0	0	7	0	0	1	0	0	2	0	0	0	0	0	0	0	10
Barry Tycholiz	0	0	0	0	0	0	0	0	0	0	0	0	0	4	6	71	50	35	4	7	0	10	1	2	190
Christian Yoder	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	2
Phillip Polsky	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	1	1	2	0	1	0	9
Alice Wright	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	2
Mark Haedicke	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	2	0	1	2	0	10
Frank Vickers	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	1	5
Daren Farmer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	2
Kimberly Bates	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
Lloyd Will	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Stephanie Panus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2
Louise Kitchen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2
Chris Germany	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Tana Jones	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
Sheila Tweed	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Elizabeth Sager	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Brent Hendry	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
Sara Shackleton	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 5.11: Total email exchanges between “Nemec” and other employees

month-1		month-2		month-3		month-4		month-5	
John Hodge	0.2297	Stephanie Miller	0.2297	Stephanie Miller	0.1492	Susan Scott	0.1721	John Hodge	0.2297
Stephanie Miller	0.2297	Susan Scott	0.2297	Susan Scott	0.1411	Barbara Gray	0.1255	Susan Scott	0.2297
Barbara Gray	0.0	Barbara Gray	0.0	Phillip Allen	0.0698	Monique Sanchez	0.0731	Barbara Gray	0.0
Phillip Allen	0.0	Phillip Allen	0.0	Vince Kaminski	0.0514	Mark Taylor	0.0653	Phillip Allen	0.0
Tana Jones	0.0	Brent Hendry	0.0	Matthew Lenhart	0.0498	Mark Haedicke	0.0552	Frank Ermis	0.0
Lloyd Will	0.0	Lloyd Will	0.0	Monique Sanchez	0.0498	Carol St. Clair	0.0332	Barry Tycholiz	0.0
Daren Farmer	0.0	David Portz	0.0	Kimberly Watson	0.0422	Tana Jones	0.033	Tori Kuykendall	0.0
Chris Germany	0.0	Scott Goodell	0.0	Louise Kitchen	0.0205	Sara Shackleton	0.0206	Chris Germany	0.0
Kimberly Watson	0.0	Janet Moore	0.0	Harry Arora	0.0156	Susan Bailey	0.0205	Kimberly Watson	0.0
Jean Mrha	0.0	Christopher Calger	0.0	Mark Taylor	0.0154	Marie Heard	0.0193	Jean Mrha	0.0

Table 5.12: "Nemec" Rooted-PageRank results over the months 1-5

month-6		month-7		month-8		month-9		month-10	
Susan Scott	0.0717	Susan Scott	0.0905	Debra Perlingiere	0.1302	Mark Whitt	0.0527	Colleen Sullivan	0.0582
Richard Sanders	0.0708	Debra Perlingiere	0.0784	Mark Whitt	0.0826	Barbara Gray	0.0481	Debra Perlingiere	0.0522
Barbara Gray	0.0607	Barbara Gray	0.0639	Barbara Gray	0.0668	Mary Cook	0.0383	Barbara Gray	0.0497
Mark Whitt	0.0574	Stephanie Miller	0.0639	Susan Scott	0.0668	Debra Perlingiere	0.0351	Paul Lucci	0.0469
Mark Taylor	0.0508	John Arnold	0.0338	Jeffrey Hodge	0.028	Paul Lucci	0.0325	Mark Whitt	0.0456
Monique Sanchez	0.0337	Jeffrey Hodge	0.0336	Scott Neal	0.0196	Richard Sanders	0.0309	David Fuller	0.0407
Tana Jones	0.0301	Robin Rodrigue	0.0256	Tori Kuykendall	0.0195	Mark Taylor	0.0289	Matthew Lenhart	0.0208
Mark Haedicke	0.03	Stacy Dickson	0.0214	Stacy Dickson	0.0194	Tyrell Harrison	0.0289	Jeffrey Hodge	0.0202
Jeffrey Hodge	0.0273	Tana Jones	0.0201	Scott Goodell	0.0174	Susan Scott	0.0267	Scott Neal	0.0197
Carol St. Clair	0.021	John Lavorato	0.0177	Tana Jones	0.017	Elizabeth Sager	0.0216	Elizabeth Sager	0.0196

Table 5.13: "Nemec" Rooted-PageRank results over the months 6-10

month-11		month-12		month-13		month-14		month-15	
Chris Germany	0.0509	Leslie Hansen	0.051	Stephanie Miller	0.0543	Mark Taylor	0.0528	Barry Tycholiz	0.0389
Mark Taylor	0.036	Mark Whitt	0.0481	Barbara Gray	0.047	Mark Greenberg	0.0478	John Lavorato	0.0389
Scott Neal	0.0354	Barbara Gray	0.0473	Julie Gomez	0.043	Tana Jones	0.0387	Mark Whitt	0.0388
Barbara Gray	0.0346	Stephanie Miller	0.0454	Debra Perlingiere	0.0399	Mark Whitt	0.0329	Tyrell Harrison	0.0386
Colleen Sullivan	0.0328	Debra Perlingiere	0.0445	Michael Grigsby	0.0368	Tyrell Harrison	0.0313	Paul Lucci	0.0344
Paul Lucci	0.0316	Paul Lucci	0.0401	Paul Lucci	0.0328	Jeffrey Hodge	0.027	Debra Perlingiere	0.03
Mark Whitt	0.0307	Jeffrey Hodge	0.0281	John Griffith	0.0318	Stephanie Miller	0.0251	Julie Gomez	0.0283
John Hodge	0.0288	John Lavorato	0.0235	Matthew Lenhart	0.0315	Paul Lucci	0.0243	Matthew Lenhart	0.0258
Julie Gomez	0.0274	Mark Taylor	0.0219	Mark Whitt	0.0313	Barry Tycholiz	0.024	Stephanie Miller	0.0241
Tyrell Harrison	0.0191	Mark Haedicke	0.0205	Sheila Tweed	0.0209	Debra Perlingiere	0.024	John Hodge	0.024

Table 5.14: "Nemec" Rooted-PageRank results over the months 11-15

month-16		month-17		month-18		month-19		month-20	
Barry Tycholiz	0.0492	John Lavorato	0.0372	Mark Haedicke	0.0465	Mark Haedicke	0.0531	Mark Haedicke	0.0469
Tyrell Harrison	0.0307	Barry Tycholiz	0.0346	Stephanie Miller	0.0457	Barry Tycholiz	0.0462	Phillip Polsky	0.0305
Stephanie Miller	0.0295	Mark Whitt	0.0325	Barry Tycholiz	0.0385	Tyrell Harrison	0.044	Debra Perlingiere	0.0304
Paul Lucci	0.0294	Mark Taylor	0.026	Phillip Polsky	0.0351	Debra Perlingiere	0.0435	Mark Taylor	0.0279
Jeffrey Hodge	0.0281	Jeffrey Hodge	0.026	Jeffrey Hodge	0.0328	Phillip Polsky	0.0309	Stephanie Miller	0.0279
Mark Whitt	0.0273	Elizabeth Sager	0.0236	Debra Perlingiere	0.0293	Jeffrey Hodge	0.0299	Barry Tycholiz	0.0264
Julie Gomez	0.0234	Mark Haedicke	0.0226	Barbara Gray	0.027	Stephanie Miller	0.0285	Paul Lucci	0.0251
Debra Perlingiere	0.0225	Debra Perlingiere	0.0212	David Fuller	0.0251	David Fuller	0.0282	David Fuller	0.0247
Barbara Gray	0.0209	Tyrell Harrison	0.0203	Paul Lucci	0.0233	Stacy Dickson	0.0242	Jeffrey Hodge	0.0239
Scott Neal	0.0205	Paul Lucci	0.0202	Mark Whitt	0.022	Mark Whitt	0.023	Tyrell Harrison	0.0225

Table 5.15: "Nemec" Rooted-PageRank results over the months 16-20

month-21		month-22		month-23		month-24	
Iris Mack	0.0354	Barry Tycholiz	0.0373	Barry Tycholiz	0.031	Louise Kitchen	0.0431
Tyrell Harrison	0.0324	Mark Whitt	0.0329	Mark Whitt	0.0275	Cynthia Mann	0.0335
Mark Whitt	0.0317	Mark Haedicke	0.0292	Mark Haedicke	0.0274	Stephanie Panus	0.0329
Phillip Polsky	0.0314	Jeffrey Hodge	0.0275	Jeffrey Hodge	0.0255	Mary Cook	0.0282
Barry Tycholiz	0.0281	Iris Mack	0.0274	Debra Perlingiere	0.025	Barry Tycholiz	0.0265
Paul Lucci	0.028	Paul Lucci	0.0262	Marie Heard	0.0237	Mark Whitt	0.024
Stephanie Miller	0.0278	Tyrell Harrison	0.0253	Tyrell Harrison	0.0235	Sara Shackleton	0.0235
David Fuller	0.0271	David Fuller	0.0247	David Fuller	0.0234	Jeffrey Hodge	0.022
Debra Perlingiere	0.0214	Stephanie Miller	0.0241	Stephanie Miller	0.0226	Leslie Hansen	0.0218
Barbara Gray	0.0197	Stacy Dickson	0.0235	Paul Lucci	0.0213	Stephanie Miller	0.0203

Table 5.16: "Nemec" Rooted-PageRank results over the months 21-24

Average-based Time-sensitive RPR (AT-RPR)

The final list of employees, generated using the average-based aggregation approach, is sorted by the average scores of rooted-PageRank achieved by each employee over all months. The final list for our case study is shown in Table 5.17.

Rank	Employees	avg-score
1	Stephanie Miller	0.0616
2	Susan Scott	0.0446
3	Barbara Gray	0.0300
4	Debra Perlingiere	0.0276
5	Mark Whitt	0.0265
6	John Hodge	0.0239
7	Mark Taylor	0.0195
8	Jeffrey Hodge	0.0192
9	Mark Haedicke	0.0188
10	Paul Lucci	0.0183

Table 5.17: Top 10 employess from the final list sorted by average RPR scores.

Rank	Employees	total-votes
1	Barbara Gray	12
2	Mark Whitt	11
3	Stephanie Miller	10
6	Mark Haedicke	9
6	Barry Tycholiz	9
6	Mark Taylor	9
8	Susan Scott	8
8	Debra Perlingiere	8
10	Jeffrey Hodge	6
10	John Lavorato	6

Table 5.18: Top 10 employess from the final list sorted by total votes each employees obtained when using the cut-off at 4.

Vote-based Time-sensitive RPR (VT-RPR)

As we saw in Section 5.3.2, the best results for VT-RPR were obtained using the cut-off $k = 4$, so for simplicity we only present here the results of VT-RPR using the cut-off $k = 4$. Each employee appearing within the top four ranks in each of the 24 sorted lists, will receive a vote. However, it is noted here that in some months, a group of employees may share an equal rooted-PageRank score and come within the top 4 ranks in the sorted list. For example, month 5 (Table 5.12), both "John Hodge" and "Susan Scott" have *rank* 1 with a rooted-PageRank score of 0.2297. Moreover, all the other employees have *rank* 3 with rooted-PageRank scores of 0. As a result, all employees receive one vote, since all of them have $rank \leq 4$ in that month.

The final list of employees is generated and sorted by the total votes received by each employee over all months. The final list for our case study is shown in Table 5.18.

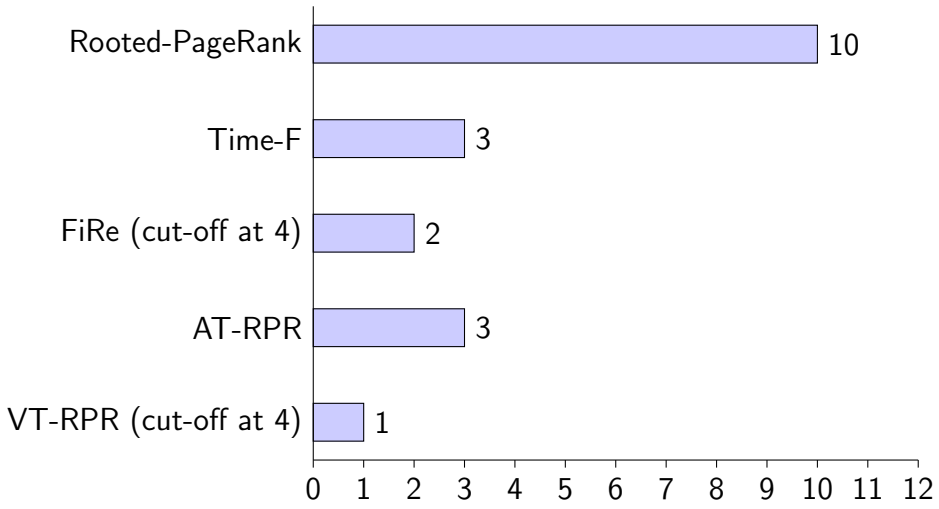

Chart 5.1: The rank of Nemec's manager in the lists sorted by Rooted-PageRank, Time-F, FiRe, Average-based Time-sensitive RPR and Vote-based Time-sensitive RPR

Chart 5.1 shows, VT-RPR is the best approach for detecting Nemec's manager (Barbara Gray). In VT-RPR, Gray came first in the final list, compared to AT-RPR, in which she appeared in the third position. However, the performance of both AT-RPR and Time-F was identical and slightly worse than the FiRe approach in which Gray was ranked second. All our approaches were considerably better than our baseline RPR where 9 other employees came before the manager in the final sorted list.

5.4 Summary

In this chapter, we introduced T-RPR, a method for detecting hierarchical ties in an interaction graph. We investigated the impact of the temporal dimension in the ranking process and adapted Rooted-PageRank to capture the dynamics of the interactions over time between the actors in the network. We explored two variants

for aggregating the rankings produced at each time slot. Experiments on two real datasets showed that, although the Time-F and FiRe approaches returned better results than those obtained by S-RPR, the approach proposed in this chapter, namely T-RPR, was superior to Time-F, FiRe and S-RPR, hence providing reasonable empirical justification and support for our claim that *"time matters"* in detecting hierarchical ties. The results also showed that T-RPR works better when the undirected interaction graph is used. This highlights the importance of the volume of the interaction traffic more than the fraction of incoming and outgoing interactions as a distinctive feature of detecting the underlying hierarchical ties when the interactions between actors are asymmetric. These findings were supported by presenting the results of applying these approaches over the same case study presented in the previous chapter.

Chapter 6

Detecting Students' Roles using Supervised Learning

6.1 Overview

In previous chapters we studied the problem of detecting hierarchical relationships between actors in an interaction network. In this chapter and the next, we change our focus to detecting the hierarchical roles of actors in networks. In particular, we consider the hierarchical roles of actors participating in a project.

Working on a project means working in a team, and a project team can be seen as a social group where team members are involved in social interactions with each other, share interests and have the common goal of completing the project. Thus, based on the learning framework presented in [4], where students assumed three different roles defined in PRINCE2^{TM}, namely Executive (EX), Project Manager (PM) and Team Member (TM), the overall objective of this chapter is to examine the relationships between students through their communication behaviour using an asynchronous communication tool. More specifically, this work analyses the capability of Data Mining field (DM) to identify patterns of interaction between students that are directly related to their position in the project. These roles include:

- EX: Executive. This role is charged with effective management of the project. Each project is managed by a team of three to five EXs.

- PM: Project Manager, with management responsibilities. On behalf of the EX, the PMs have the authority to run the project on a day-to-day basis. Each project is managed by a team of seven to twelve PMs.

- TM: Team member, with engineering task development responsibilities. Each project is composed of seven to sixteen TMs.

Asynchronous communication tools can play an important role in students' collaborative learning through two types of actions: reading and writing messages [104]. We focused our study on analysing this type of communication and investigated how reading and writing activities carried out by students working on a project, can be used to detect students' roles.

The knowledge acquired by DM algorithms can help teachers understand how students' roles in a project relate to their communication behaviour and whether the students play their presumed roles in the project. Our results show that, by choosing an appropriate set of features related to students' communication patterns, a number of DM algorithms are able to classify students' roles with both precision and recall of over 95%.

The results presented in this chapter have been published previously in [55].

6.2 Problem Setting

The problem we wish to solve is as follows. We are given a set of students V who have interacted via a set of interactions I, through the use of any of the following asynchronous communication tools (as shown in Figure 6.2) provided by the project portfolio management (PPM) software used to facilitate the development of the learning experience (http://www.project.net):

- *Blogs*. Blog posts can be created either globally for the project or tied to specific tasks, keeping a complete record of activity associated with that item easily accessible. Thus, project members create blog posts to record recent activities or completed work as well as to ask something related to the work to be done (Figure 6.1). In summary, blogs:
  - record completed work and general comments,
  - allow members to view a log of all work activity for a project, and
  - facilitate two-way communication between management and team members.
- *Discussion groups*. Project members can establish threaded discussions. The centralised discussion board allows project members to consolidate thoughts and ideas and share running commentary with other project members. In this particular application, discussion posts were also used to inform those project members responsible for a deliverable that the requested work had been done. Thus, the person responsible for that deliverable replied in order to provide feedback to the performed work in a positive (acceptance) or negative (request changes) way. Furthermore, it is possible to see who posted a discussion comment and who has viewed your comments (Figure 6.2). In summary, a project member can:
  - Hold discussions around specific deliverables/documents.
  - Track who has viewed each message.

From these interactions we derive a number of features. These features might be simple, such as the total number of messages posted by each student, or more complex, such as the PageRank (PR) score of each student derived from a graph representing the set of interactions I. Given this information as input, we want to find a method to infer the different roles students play in the project conversations.

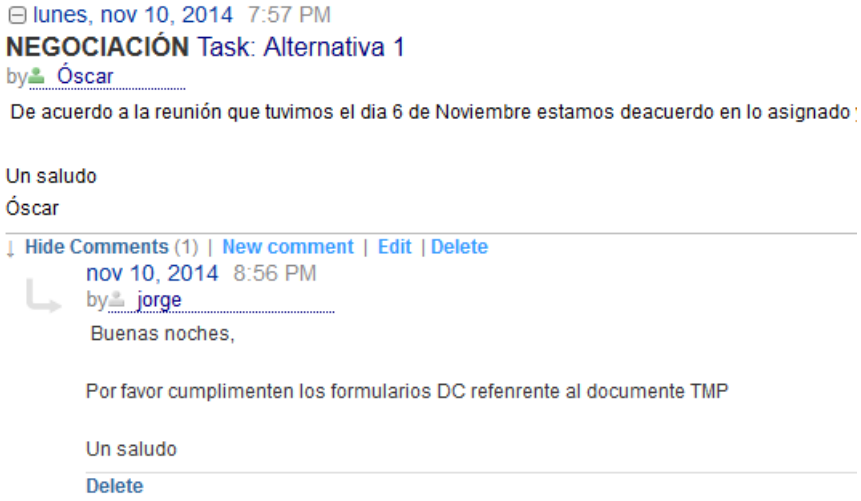

Figure 6.1: Sample screen for blog messages held within a project

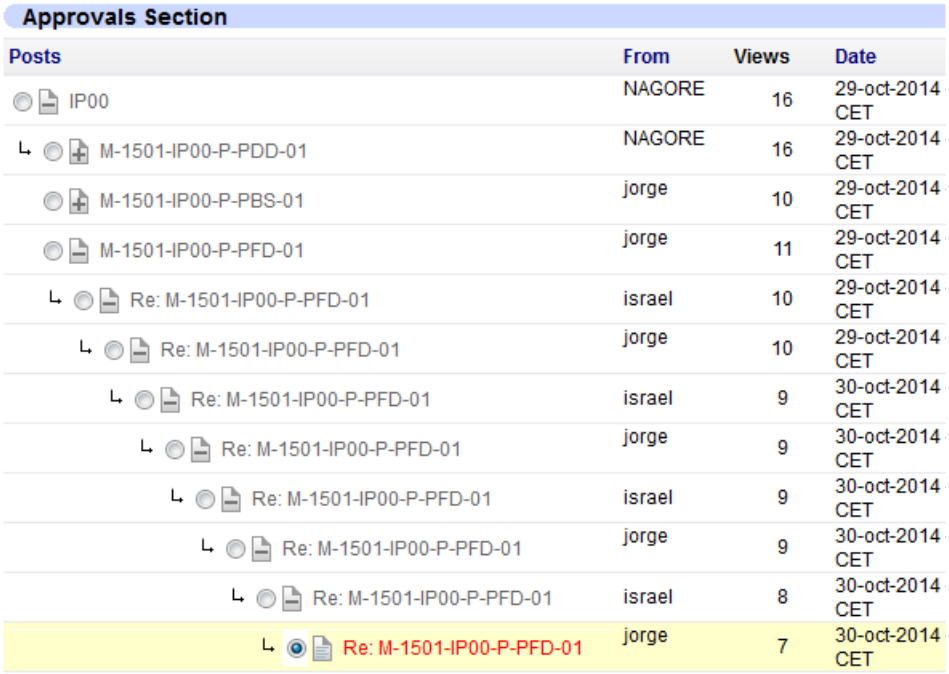

Figure 6.2: Sample screen for discussions held within a project

Input to the method also includes the number of roles; the output should be a classification of each student to a role.

We represent the input to the role-inference problem by the model $M = (V, R, I, F, M_F)$ where:

- $V = \{v_1, \ldots, v_n\}$ is the set of n students using the communication tool. We sometimes refer to individual students by u and v.

- $R = \{R_1, \ldots, R_m\}$ is the set of m possible roles assigned to students.
- I is the set of messages students submitted through the communication tool. Each message is represented by a tuple $(s, time, type, r)$, where $s \in V$ is the sender of the message, *time* is the message timestamp, and *type* is the message type which takes its value from a known finite set of types. If the message is not a reply to a previous message, then r is zero; otherwise, $r \in V$ is the student who sent or posted the message to which the current message is a reply.
- $F = \{f_1, f_2, \ldots, f_k\}$ is a set of k features derived from I.
- M_F is an $n \times k$ matrix mapping students to their feature values. For example, $M_F(1, 2) = 10$ means that the first student has value 10 for the second feature.

Given the above model M as input, we want to infer the n-dimensional vector M_R which maps each student to his or her role in the conversation. For example, $M_R(3) = 2$ would mean that the third student has role 2.

6.3 Supervised Approach

The approach we used to detect the students' roles consists of four stages as shown in Figure 6.3. Firstly, we collected the message data I which extends over three months. Then, we pre-processed the collected data in order to produce the matrix M_F. This required first finding the set of features F. Next, various supervised learning approaches were applied to build the models which classify the students according to their roles. Finally, the results of detecting students' roles using the obtained models (i.e. the vector M_R) were compared against the actual vector of student roles in terms of recall, precision and F-measure. Each of these stages are described in more detail below.

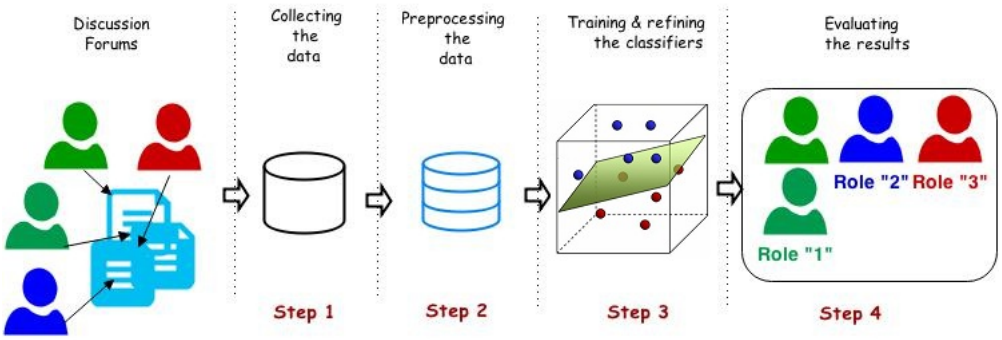

Figure 6.3: Processing stages

6.3.1 Data collection

The dataset we used is from on-line asynchronous communication tools belonging to Universidad de la Rioja and Universidad Politécnica de Madrid. These tools are based on the PPM software used to support the learning experience (http://www.project.net) and are used as a tool for coordinating groups of students in order to accomplish and complete the projects they are working on. We gathered the usage data for 194 students organised in 8 different projects. In each project, there are about 25 students. Six projects started in October and finished at the end of December in 2013. The remaining two projects extended over the same period in 2014.

Recall that three different roles could be played by the students in the projects: students in Role-1 are executives (EX), those in Role-2 are project managers (PM), and those in Role-3 are team members (TM). Also, the communication among students is via a blog entry or discussion post. These can be categorised as follows:

- BW: blog entry related to reported work.
- BT: blog entry related to a task. This can be used to ask something about the work to be done.
- BE: blog entry related to anything else.
- BR: reply to a blog entry.
- PE: post entry.
- PR: reply to a post.

Table 6.1: Statistics about students and messages for each project.

	Numbers of students				Numbers of messages						
Project	Role-1	Role-2	Role-3	total	BW	BT	BE	BR	PE	PR	total
1	3	12	11	26	641	18	39	92	57	374	1221
2	3	11	10	24	475	49	87	54	35	509	1209
3	3	11	10	24	401	43	97	39	54	741	1375
4	4	10	8	22	484	32	223	259	68	580	1646
5	4	10	9	23	426	9	190	182	38	746	1591
6	5	10	7	22	440	59	34	72	42	669	1316
7	3	9	16	28	342	39	42	50	36	510	1019
8	3	7	15	25	545	29	56	60	79	784	1553
All	28	80	86	194	3760	278	768	808	409	4913	10936

Recall from Section 6.2, in the case of post/blog reply, the message to which the post/blog is replying, is known in the data. Table 6.1 lists the full statistics of the collected data.

6.3.2 Data pre-processing

In this step, a set of features is generated for each student. These features are used to train the classification models. The generated features can be organised into four different categories as described below.

Quantitative features

These features are based on statistical information about student activities within the communication tools. They include:

- total-sent: the total number of messages sent by the student over the full period.
- total-viewed: the total number of messages viewed by the student over the full period.
- total-BW, total-BT, total-BE, total-BR, total-PE, and total-PR: These are the total numbers of messages of different types sent by the student over the full period.

Frequency-based feature

We use a feature, which we call *viewingCommitment*, to measure a student's commitment in viewing the messages sent by other students in their project. We refer to this feature as "viewing" instead of "reading" because we can be sure that a message has been displayed to the student but it is not possible to know if the student has effectively read it, even if the student replies to that message. In spite of this uncertainty, and because students are not being evaluated according to the messages they view which could corrupt the students' behaviour, we think that this feature can provide useful information about the students' interest in the project. This feature is defined as:

$$viewingCommitment(v) = \frac{1}{t} \times \sum_{d=1}^{t} \frac{S(v,d)}{A(d)}$$

where d is the day index ($d = 1$ is the day of first reading), t is the total number of project days, $S(v,d)$ is the total number of messages the student v has viewed from the day of first reading until day d, and $A(d)$ is the total number of messages that have been viewed by at least one student in the project from the first day until day d.

The motivation behind defining the function in this way is that we want to measure the viewing activity of a student relative to the other students who are working on the same project. A student v may view a message only a few days after the same message has been viewed by another student. The definition penalises the student for each day of delay in which the student defers viewing messages that have been viewed previously by others. Defining the function in this cumulative way captures the student's viewing pattern. Moreover, this definition avoids "division by zero" when none of the students view any messages on a particular day.

From the definition, $viewingCommitment(v) \in [0,1]$, where a higher score means that student v is more active in viewing messages relative to other students' viewing activities.

Interaction-based features

These features capture the interactions between students who are working on the same project. Firstly, we need to generate the reply-graph $G_{reply}(V_i, E_i)$, where V_i is the set of students who are working on project i, and $(v, u) \in E_i$ if u , $v \in V_i$ and v replied to one of u's messages. Having built the reply-graph, we run two known algorithms, PageRank [93] and HITs (described in Chapter 2) [66], in order to generate the interaction-based features as follows:

- PageRank-feature: this is the PageRank score that the student achieved when we run PageRank on the reply-graph.
- Authority-feature and Hub-feature: these are the authority and hub scores that the student achieved when we run HITs on the reply-graph.

Time-based features

These features capture the dynamics of the quantitative features and how they change over the time. We divided the project period into n equal time-slots, and experimented with different numbers of time-slots ($n = 10, 20$). In this chapter, we only report the best results which were obtained for $n = 20$ (the full set of results can be found in Appendix B). In this case, each time-slot represents about 3 days of the project period. For each time slot, we calculate the total number of messages sent by each student for each message type individually and for all types together. The result of this process is 140 time-based features (7 features over 20 time-slots). Each of these features relates to one time-slot. For example, total-sent(3) is the total number of messages sent by the student within the third time-slot. Similarly, total-BT(5) is the total number of type BT messages sent within the fifth slot by the student.

6.3.3 Classifier training and refinement

The aim of this step is to build a classification model that is able to detect each student's role from their on-line activities. We used various classification algorithms that belong to different categories, based on those available in Weka [134]:

- **Bayes-based algorithms** are probabilistic classifiers based on Bayes theorem. We tried both "Bayes Net", which uses a Bayes Network classifier such as K2 and B [12], and "NaiveBayes", which uses a simple Naive Bayes classifier in which numeric attributes are modelled by a normal distribution [38].

- **Function-based algorithms** try to fit a function to the data. "Logistic" builds and uses a multinomial logistic regression model with a ridge estimator [71]. "MultilayerPerceptron" uses a back-propagation network to classify instances [111]. "RBFNetwork" implements a normalised Gaussian radial basis function network [97]. "SMO" implements a specific sequential minimal optimisation algorithm for training a support vector classifier [100].

- **Rules-based algorithms** learn classification rules. DTNB builds a decision table/naive Bayes hybrid classifier [48]. JRip implements a propositional rule learner as an optimised version of the IREP algorithm [28]. NNge is a nearest-neighbour-like algorithm using non-nested generalised exemplars which are hyperrectangles that can be viewed as rules [83]. Ridor is the implementation of a Ripple-Down Rule learner [43].

- **Tree-based algorithms** build decision trees. BFTree uses binary split for both nominal and numeric attributes [42, 114]. J48 is an optimized version of the C4.5 decision tree [101]. LADTree generates a multiclass alternating decision tree using the LogitBoost strategy [53]. RandomForest constructs random forests based on Breiman's algorithm [13].

In order to find the best classification model, we considered various groups of features in building the models. For each group of features explained below, we trained all the aforementioned algorithms and compared their results with the results obtained by using the other groups. The following three sets of features were used to train the classification models:

- **Basic Set**: This set represents the basic features relating to student activities: (1) total-sent, (2) total-viewed and (3) *viewingCommitment*.

- **Basic$^+$ Set**: In addition to the features included in the Basic set, this set includes the features related to each message type, i.e. total-BW, total-BT, total-BE, total-BR, total-PE, and total-PR. Moreover, the three interaction-based features, i.e. PageRank-feature, authority-feature and hub-feature, were also included.

- **Full Set**: All the features generated in the pre-processing stage were included. This includes all the features of the "Basic$^+$" set, as well as all 140 time-based features.

- **Filtered Set**: As the full set of features consists of a large number of features (152 features), it is likely that not all these features are relevant for detecting students' roles. If we use all features, some of these features may cause noise in the results. We used a subset of features by filtering out those that are not discriminative in detecting student roles. In order to select the most relevant time-based features, we applied an approach similar to that used by [80, 137], using the following ten feature-selection algorithms:

    1. CfsSubsetEval evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

2. ConsistencySubset-Eval evaluates the worth of a subset of attributes by the level of consistency in the class values when the training instances are projected onto the subset of attributes.
3. ChiSquaredAttributeEval evaluates the worth of an attribute by computing the value of the chi-squared statistic with respect to the class.
4. SignificanceAttributeEval evaluates the worth of an attribute by computing the probabilistic significance as a two-way function.
5. SymmetricalUncertAttributeEval evaluates the worth of an attribute by measuring the symmetrical uncertainty with respect to the class.
6. GainRatio-AttributeEval evaluates the worth of an attribute by measuring the gain ratio with respect to the class.
7. InfoGainAttributeEval evaluates the worth of an attribute by measuring the information gain with respect to the class.
8. OneRAttributeEval evaluates the worth of an attribute by using the OneR classifier.
9. ReliefFAttributeEval evaluates the worth of an attribute by repeatedly sampling an instance and considering the value of the given attribute for the nearest instance of the same and different class.
10. SVMAttributeEval evaluates the worth of an attribute by using an SVM classifier.

The first two algorithms return a subset of relevant features. However, the remaining algorithms return a ranked list of all features. In these cases, we considered only the top k features returned. The final set of filtered features consists of those selected by at least m algorithms out of the ten algorithms used. We tried several combinations of k and m values ($k = 10, 20, 30$ and $m = 1, 5$). The full set of results can be found in Appendix B. In this chapter, we only present the best results ob-

tained, which were when $k = 10$ and $m = 1$, giving rise to 19 selected features out of 152 possible features. The selected features are shown in Tables 6.2 and 6.3.

Type	Time-slots																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
BW	6		6	4		3														
BT																				
BE																				
BR																				
PE																				
PR		1	2																	
All-types	2	1	9	10		2														

Table 6.2: Frequency of appearance of time-based features using 10 feature-selection algorithms.

Basic and Basic$^+$ Features											
total BW	total BT	total BE	total BR	total PE	total PR	total sent	total viewed	viewing commitment	PageRank feature	Authority feature	Hub feature
	1				4	7	5	9	7	9	9

Table 6.3: Frequency of appearance of Basic and Basic$^+$ features using 10 feature-selection algorithms.

For example, the entry 6 for blogs of type BW in time slot 3 means that 6 out of 10 feature-selection algorithms selected the number of BW-blogs submitted by students as a discriminitive feature of students' roles.

6.3.4 Evaluating the results

In order to evaluate classification performance, we use the three scores: *precision*, *recall* and *F-measure*. First, we calculate these three scores for each role individually. Then, the weighted average is used to evaluate the overall results. This is computed by weighting the measures of role (precision, recall, F-Measure) by the proportion of students there are in that role:

$$Precision = \sum_{i=1}^{m} \omega_i \cdot Precision_i$$

$$Recall = \sum_{i=1}^{m} \omega_i \cdot Recall_i$$

$$F\text{-}Measure = \sum_{i=1}^{m} \omega_i \cdot F\text{-}Measure_i$$

where m is the total number of roles, and $Precision_i$, $Recall_i$, $F\text{-}Measure_i$ are the three scores for detecting the students of role i and ω_i is the proportion of students who have been assigned role i.

6.4 Results and Analysis

All the experiments were run using the Weka tool [134]. In order to estimate the accuracy of the obtained models, we use 10-fold cross validation in all executions. The model is built by partitioning the dataset into 10 equal subsets. Then each algorithm is executed 10 times. Each time, one subset is used as the testing set, while the other 9 form the training set. The final evaluation is based on the mean of all runs. As we mentioned before, we applied several supervised algorithms to build the classification models for detecting students' roles. For each algorithm, we used four groups of features, as described in Section 6.3.3. The full list of results can be found in Table 6.4, while the F-measure scores are shown visually in Figure 6.4.

For the "Basic" features, the best classification was generated by the NaiveBayes algorithm. The results of all algorithms ranged between 0.66 and 0.83 for precision, recall and F-measure. On the other hand, the results were better for all algorithms except Ridor and LADTree when we used the "Basic$^+$" group of features. This means that including the "interaction-based" features as well as the total count of each message type improves the classification of roles. This is clear for all the function-based algorithms particularly. For example, the best model was built by MultilayerPerceptron and Logistic which achieved around 0.86 for precision, recall and F-measure.

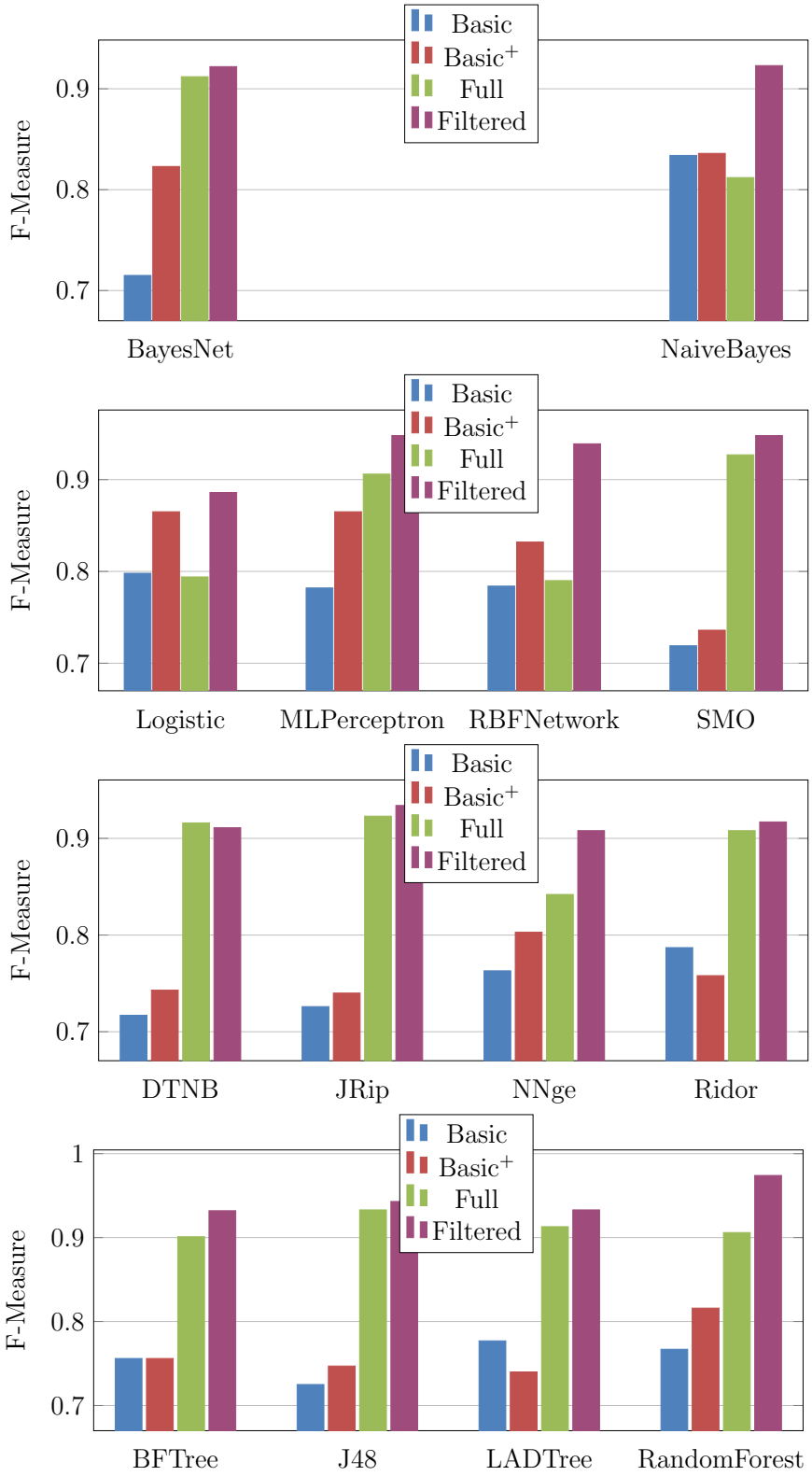

Figure 6.4: F-measure scores for each classifier using the Basic, Basic+, Full, and Filtered sets of features.

The performance of the models trained by the "Full" set of features were substantially superior. In most cases, the results of classifying students using the "Full" set of features were better than the results obtained using the "Basic" or "Basic$^+$" sets. For example, in JRip, the F-Measure improved from 0.74 in "Basic$^+$" to 0.923 using the "Full" set of features. The RBFNetwork and Logistic algorithms were the only exceptions. In the former, the F-Measure decreased from 0.832 to 0.79 using the "Basic$^+$" and "Full" set of features, respectively. The best classification model among those trained by the "Full" set of features was J48 with an F-measure of about 0.93.

As mentioned previously, the "Full" set includes a large number of features (152). In order to reduce the number of features and remove irrelevant ones, we produced a "Filtered" set of features by keeping only those selected within the top 10 features by at least one of the ten feature-selection algorithms we used. The majority of algorithms trained by the "Filtered" set of features returned better or similar results to those obtained using the "Full" set. For example, the NaiveBayes algorithm performs better in the case of the "Filtered" set, achieving an F-measure of 0.923 compared to only 0.836 and 0.812 obtained for the "Basic$^+$" and "Full" sets respectively. For BayesNet, JRip, Ridor and J48, the results obtained using the "Full" and "Filtered" sets are similar, with minor improvements using the latter set. Although DTNB worked better when it was trained with the "Full" set, the difference between the "Full" set results and "Filtered" set results is very slight. In general, all algorithms achieved an F-measure above 0.9 for the "Filtered" set. The best results using the "Filtered" set were obtained in the case of RandomForest with above 0.97 for precision, recall and F-measure.

6.4.1 Main Findings

As expected, individual attributes ("Basic" features) were partially useful to correctly classify the students' roles in the project. Quantitative and frequency-based features alone do not provide a complete picture of the interactions between project members.

On the other hand, although the information captured from the social network analysis ("interaction-based" features) generally improved mapping students to their roles, the use of "time-based" features was crucial to correctly identify students' roles. It must be noted that the complete set of these "time-based" features was not necessary to achieve good classification performances: by using the 8% of the "time-based" features — 11 variables — it was possible to achieve an F-measure above 0.95.

Additionally, the feature-selection methods showed that most of the selected "time-based" features coincide with the first weeks of working on the project, which indicates the importance of initial interactions between project members. Interaction-based features were also selected by most of the feature-selection methods. This confirms the importance of the reply-relationship among the students in their discussions. This finding is consistent with the claim that the relationships among the individuals is important to understand individual and group behaviour and/or attitudes [99]. The viewing/reading activities also were recognised as discriminative features for detecting students' roles. This is for both "total-viewed" and "*viewingCommitment*" features which were selected by 5 and 9 out of 10 feature-selection methods respectively.

The good classification results illustrate that the interactions corresponding to particular roles show distinctive patterns and asynchronous conversations have proven to be useful in identifying these project roles.

6.5 Summary

This chapter has presented an application of data mining (DM) to the detection of students' roles in a project according to their use of online communication tools (discussion posts and blogs). The analysed data included individual attributes related to messages sent and read, as well as information about the interactions between the project members provided by two social network analysis measures (PageRank [93] and HITs [66]).

Based on the results obtained using several sets of features and classification algorithms, it is possible to confirm the usefulness of DM to analyse the online interactions between students working together in a project. Moreover, it has been shown that considering information about the reply relations among the project members is more relevant than the individual attributes of students. Another interesting result is the selection of "time-based" features as relevant to identify the students' roles. Taking into account that most of these features coincide with the first weeks of the project, it seems to corroborate that the PRINCE2^{TM} project structure facilitates the students' learning process because it clarifies the project team organisation.

Table 6.4: The results of classifying students using different supervised algorithms and different sets of features.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered
Bayes-based	BayesNet	Precision	0.71	0.829	0.912	0.921
		Recall	0.722	0.825	0.912	0.923
		F-Measure	0.715	0.823	0.912	0.922
	NaiveBayes	Precision	0.836	0.849	0.835	0.926
		Recall	0.84	0.835	0.804	0.923
		F-Measure	0.834	0.836	0.812	0.923
Function-based	Logistic	Precision	0.795	0.865	0.799	0.886
		Recall	0.814	0.866	0.799	0.887
		F-Measure	0.798	0.865	0.794	0.886
	MultilayerPerceptron	Precision	0.782	0.867	0.907	0.948
		Recall	0.784	0.866	0.907	0.948
		F-Measure	0.782	0.865	0.906	0.948
	RBFNetwork	Precision	0.782	0.83	0.813	0.941
		Recall	0.789	0.835	0.778	0.938
		F-Measure	0.784	0.832	0.79	0.939
	SMO	Precision	0.669	0.687	0.927	0.949
		Recall	0.778	0.794	0.928	0.948
		F-Measure	0.719	0.736	0.927	0.948
Rules-based	DTNB	Precision	0.713	0.754	0.916	0.911
		Recall	0.753	0.742	0.918	0.912
		F-Measure	0.717	0.743	0.916	0.911
	JRip	Precision	0.729	0.74	0.922	0.935
		Recall	0.727	0.742	0.923	0.933
		F-Measure	0.726	0.74	0.923	0.934
	NNge	Precision	0.762	0.813	0.858	0.911
		Recall	0.768	0.814	0.84	0.907
		F-Measure	0.763	0.803	0.842	0.908
	Ridor	Precision	0.785	0.76	0.909	0.917
		Recall	0.794	0.758	0.907	0.918
		F-Measure	0.787	0.758	0.908	0.917
Tree-based	BFTree	Precision	0.754	0.753	0.902	0.933
		Recall	0.758	0.763	0.902	0.933
		F-Measure	0.756	0.756	0.901	0.932
	J48	Precision	0.731	0.747	0.933	0.943
		Recall	0.747	0.747	0.933	0.943
		F-Measure	0.725	0.747	0.933	0.943
	LADTree	Precision	0.777	0.737	0.914	0.933
		Recall	0.784	0.742	0.912	0.933
		F-Measure	0.777	0.74	0.913	0.933
	RandomForest	Precision	0.768	0.814	0.914	0.974
		Recall	0.768	0.82	0.912	0.974
		F-Measure	0.767	0.816	0.906	0.974

Chapter 7

A Multi-granularity Pattern-based Sequence Classification Framework

7.1 Overview

In many applications (including that of detecting students' roles from the previous chapter), sequences of events occurring over time and in order need to be studied in order to understand the generative process behind these sequences, and hence classify new examples. As a result, the need for developing efficient and flexible techniques for sequence classification that can be applied in different domains has become in demand. One of the key challenges of classification is how to identify and extract appropriate features from the data in order to train and build robust and effective classification models. This task becomes even more challenging in the case of sequences, since there are no explicit features at our disposal. Such features should be captured in a way that both the temporal dimension and the sequential order of the properties of the sequence are maintained. In addition, the number of extracted features can, in general, be rather large. Hence, the need for employing appropriate feature selection methods arises. The latter is not always trivial, due to the time dimension, which makes the feature selection process more complicated. This chapter addresses these two challenges in the area of sequence classification.

The contributions of this chapter can be summarised as follows: We formulate a multi-granularity framework for classifying sequences of discrete events. The framework consists of three phases: feature generation, feature selection, and model construction. The proposed feature generation technique can effectively capture the inherent temporal structure of the sequences by mining frequent sequential patterns at different window sizes. The extracted features capture not only the temporal aspects of the underlying sequences, but also their variability at multiple levels of time granularity. Next, the most important features are identified by applying standard variable importance algorithms for feature selection. The classification model is then constructed by using the selected features. In our experimental evaluation, we demonstrate how the proposed framework can be used for classifying students working on the same project, while interacting through asynchronous communication tools. We study the performance of our framework in terms of recall, precision, F-measure, and area under the ROC, and compare with a similarity-based baseline approach. The experimental results show that our framework is able to detect correctly the role of more than 90% of the students, compared to only 57% using the baseline similarity-based model, in the best case.

Our goal in this chapter is not to compete with the literature of sequence classification, but to introduce a multi-granularity pattern-based classification framework that employs the novel idea of using frequent patterns at variable window lengths as class features, and demonstrate its high applicability to the application area of education.

7.2 Baseline: Nearest Neighbour classifier

A baseline method for classifying sequences of discrete events is to apply a standard nearest neighbour classifier under a string similarity or distance function, such as Smith-Waterman [115] or edit-distance [88]).

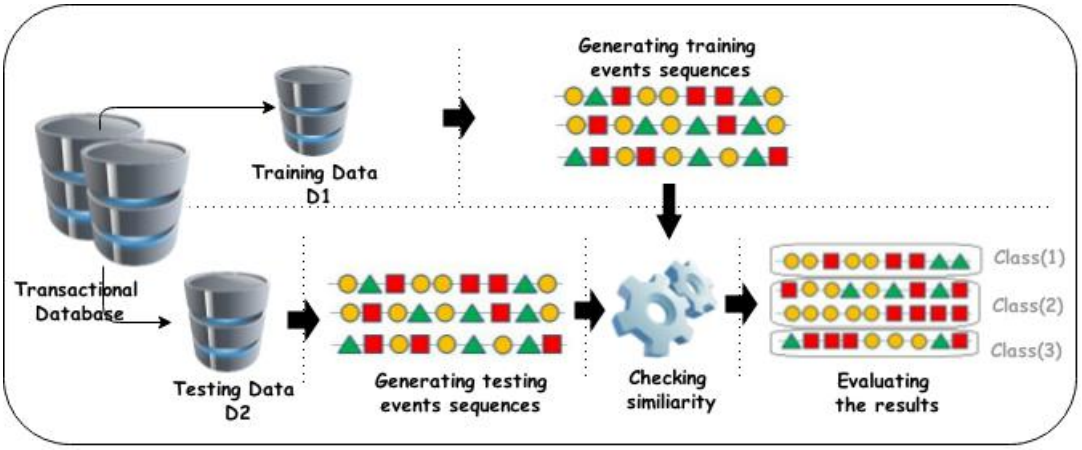

Figure 7.1: Stages of a Nearest Neighbour classifier.

Consider a collection of labelled sequences, D, where each record $s \in \mathcal{D}$ is a sequence of discrete events. Suppose that $\mathcal{D}$ is split into two non-overlapping parts $\mathcal{D}_1$ and $\mathcal{D}_2$, the training and testing datasets, respectively. Building a Nearest Neighbour classifier consists of several stages as shown in Figure 7.1:

- In Stage 1, we retrieve all event sequences from $\mathcal{D}_1$. We call the list of obtained sequences the *training list* L_{train}. Each sequence in the list consists of a set of events that occurred in order. In addition, each sequence belongs to one of the known classes.

- In Stage 2, we retrieve all event sequences from $\mathcal{D}_2$. The resulting sequences form the *testing list* L_{test}. Our aim is to build a classifier that is able to detect the class of each sequence in the testing list using the list of events comprising the sequence and the training list.

- In Stage 3, for each sequence $s \in \mathcal{D}_2$, we identify the most similar sequence in $\mathcal{D}_1$. Similarity is computed by using a common similarity function for strings, such as edit-distance [88] or Smith-Waterman [115]). The class of each test sequence s is determined as follows:

  - If only a single sequence in $r \in D_2$ has the highest similarity with s, then the class of sequence s is taken to be the class of r. In other

words, for each sequence $s \in L_{test}$, $class(s)$ is made equal to $class(r)$ if $\arg\max_{r \in L_{train}} sim(s, r)$ and $\nexists r' \in L_{train}(r \neq r') : sim(s, r) = sim(s, r')$

- If the highest score is shared by multiple sequences $R = \{r_1, r_2, .., r_n\} \subseteq \mathcal{D}_2$, then the class of s is the majority class in R.
- If there is no majority class among the sequences in R, we randomly choose one of classes in R.

- In Stage 4, we evaluate the classification performance in terms of *precision*, *recall*, *F-measure*, and AUC. First, we calculate these three scores for each class individually. Then, the weighted average is used to evaluate the overall results. This is computed by weighting the measures of class (precision, recall, F-Measure) by the proportion of sequences there are in that class. We also use "AUC" (area under the ROC) to evaluate the performance of our approach.

7.3 Multi-granularity pattern-based classification

In this section, we introduce a multi-granularity sequence classification framework. The framework consists of three phases as shown in Figure 7.2: the feature generation phase, the feature selection phase, and finally the model construction phase. Next, we first provide some definitions and then describe the three phases.

7.3.1 Definitions

Let $\mathcal{E}$ be the space of possible events that can occur in a sequence. A *transaction* is a triple $T = \langle id, e, t \rangle$, where $T.id$ is the identifier of the transaction, $T.e \subseteq \mathcal{E}$ is a single event or a set of events from $\mathcal{E}$, and $T.t$ is the time-stamp of the transaction. For example, a transaction may correspond to the set of student communication activities (events) during a day or a week (time-stamp).

Given a set of predefined classes $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, a *transactional sequence* S is of the form $\langle id, c, (T_1, \ldots, T_n)\rangle$, where $S.id$ is the identifier of S, $S.c \in \mathcal{C}$ is the class of S, and $(T_1, \ldots, T_n)$ is an ordered set of transactions, such that, if $1 \leq i < j \leq n$ then T_i occurs before T_j. In other words, the transactional sequence respects the order of transactions within it. A collection of transactional sequences defines a *dataset* $\mathcal{D}$.

Before proceeding to the first phase of the proposed framework, we assume that our dataset $\mathcal{D}$ is partitioned into two non-overlapping parts: a training set $\mathcal{D}_1$ and a validation set $\mathcal{D}_2$. We will use $\mathcal{D}_1$ to generate our feature space and $\mathcal{D}_2$ to validate the constructed model.

7.3.2 Feature generation phase

The aim of this phase is to generate sequential features that capture the inherent time dependencies between the transactions and are highly correlated with the class label. These features correspond to maximum sequences that are characteristic of a class in $\mathcal{D}_1$, i.e., occur frequently in that class, but at the same time they are infrequent in other classes. These frequent sequences will be used as the set of features when building the multi-granularity classifier in the next phase.

Firstly, we employ SPAM, an efficient algorithm proposed by Ayres et al. [7] for mining frequent sequential patterns within a transactional dataset. SPAM has been shown to be efficient in mining frequent sequences when the sequential patterns in the data are very long. Specifically, SPAM is applied to $\mathcal{D}_1$ and the set of frequent patterns per class is extracted. We should note, however, that our framework is flexible enough to allow for any alternative sequential pattern mining algorithm to be applied.

More formally, a *sequential pattern* $p = (p_1, \ldots, p_m)$ is a sequence of *patterns*, where each pattern p_i is a subset of $\mathcal{E}$. Let $\mathcal{P}$ define the space of possible sequential

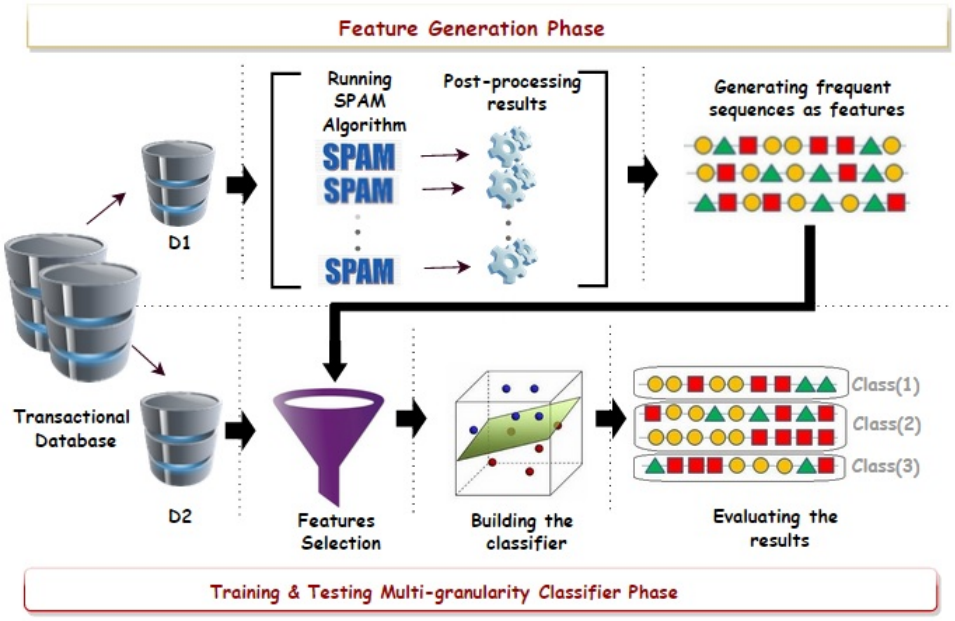

Figure 7.2: Stages of the multi-granularity classification framework.

patterns that can be generated from $\mathcal{E}$. We say that a transaction $T = \langle id, e, t \rangle$ *supports* pattern p, denoted $p \prec T$, if $p \subseteq e$. In addition, a transactional sequence S supports sequential pattern $p \in \mathcal{P}$, denoted $p \prec S$, if

$$\forall p_i : i \in \{1, 2, \ldots, m\} \begin{cases} \exists T_j : p_i \prec T_j \text{ if } i = 1 \\ \exists T_j : p_i \prec T_j \text{ and } \exists T_k : p_{i-1} \prec T_k \text{ and } k < j \text{ if } i > 1 \end{cases}$$

For each class $c \in \mathcal{C}$, let $|c|$ denote the number of sequences in $\mathcal{D}$ which belong to class c, i.e. $|c| = |\{S \in \mathcal{D}_1 \mid S.class = c\}|$. The *frequency* of a sequential pattern $p \in \mathcal{P}$ in class $c \in \mathcal{C}$ is defined as follows:

$$Freq_D(p, c) = \frac{|\{S \in \mathcal{D}_1 \mid p \prec S \wedge S.class = c\}|}{|c|}.$$

Given a user-specified minimum frequency threshold σ, the set $\mathcal{P}_c$ of *frequent sequences of class* c is the following:

$$\mathcal{P}_c = f(\mathcal{P}, D, \prec, \sigma, c) = \{p \in \mathcal{P} \mid Freq_D(p, c) \geq \sigma\},$$

where f is a function that corresponds to the algorithm producing these patterns, in our case SPAM.

Sequence Id	Class	Transactions
1	c_1	$(\{a\}, \{a, b\}, \{e\})$
2	c_1	$(\{a\}, \{c, d\}, \{a\}, \{a, b\})$
3	c_1	$(\{e\}, \{a, b\}, \{a, b\}, \{c, d, e\})$
4	c_1	$(\{a\}, \{b\}, \{c\}, \{a\})$
5	c_1	$(\{a, b\}, \{a, b\}, \{a, b\})$
6	c_2	$(\{a\}, \{a\}, \{a, b\})$
7	c_2	$(\{a, c, b\})$
8	c_2	$(\{b\}, \{d\}, \{a, d\}, \{a, b\})$
9	c_2	$(\{a, b\})$
10	c_2	$(\{e\}, \{a, b\}, \{e\}, \{a, b\})$

Table 7.1: Example of a transactional dataset.

Example. Assume that we have an event space $\mathcal{E} = \{a, b, c, d, e\}$ and a transactional dataset $\mathcal{D}$ consisting of the 10 sequences shown in Table 7.1. Each row in the table represents one sequence. The first column shows the sequence identifier, the second column indicates the class of the sequence, and the third column shows the events of transactions forming the sequence. Each transaction in the sequence consists of a set of one or more events. To simplify the example, we have omitted the identifiers and durations of transactions. Now consider the sequential pattern $p = (\{a\}, \{a, b\})$ and the minimum frequency threshold $\sigma = 0.7$. Pattern p is a frequent sequence for class c_1, because p is supported by 4 out of the 5 sequences in class c_1, namely sequences 1, 2, 3 and 5, hence

$$Freq_D((\{a\}, \{a, b\}), c_1) = 4/5 = 0.8 \geq 0.7.$$

It becomes apparent that each set $\mathcal{P}_c$ captures the sequential patterns that occur frequently in a class c. Nonetheless, it fails to take into consideration the exact location of these patterns in the sequences. For example, in the student project application used in the previous chapter, a similar frequent communication pattern

could occur between a project executive and a project manager, but the location of this pattern in the sequences might be different between the two roles. Hence, using the pattern as a classification feature and ignoring the temporal location would increase the classification error.

In order to capture time dependencies between the patterns and classes in the sequences at different levels of granularity, we segment the time-line of the dataset into n non-overlapping windows $\{w_1, \ldots, w_n\}$. Given a minimum frequency threshold σ, we run SPAM to find the set of frequent sequences $\mathcal{P}_c^k$ for each class c and each window w_k:

$$\mathcal{P}_c^k = f(\mathcal{P}, D_1, \prec, \sigma, c, n, k) = \{p \in \mathcal{P} \mid Freq_{\mathcal{D}_1}(p, c, n, k) \geq \sigma\},$$

where f is again the algorithm that generates the set of patterns, $\prec$ is the support operator, n is the total number of windows, $k \in [1, n]$ is the window index, $Freq_{\mathcal{D}_1}(p, c, n, k)$ is the frequency of the sequence (pattern) p that occurs within the window w_k, when we consider sequences of class c in $\mathcal{D}_1$.

Example. To clarify the idea, we give a simple example of a dataset that consists of five transactional sequences. Each sequence consists of 10 transactions, where each transaction consists of a single event from the set $\mathcal{E} = \{a, b, c, d, e\}$. Given a threshold σ of 0.8, Figure 7.3 shows the frequent patterns in each window when we divide the time-line into (a) 2 and (b) 5 windows.

Next, we post-process the SPAM output to reduce the amount of redundancy and dependencies in the features. More precisely, we reduce the number of returned patterns by keeping only the *maximal* ones, i.e., those for which no superset is frequent. For notation purposes, we use f_{max} to denote the whole algorithmic procedure: running SPAM and pruning out the non-maximal patterns. For example, Figure 7.3(a) shows 7 frequent sequences for the first window and 5 frequent sequences for the second. In each case, there is only one maximal sequence: (a, b, c) for the first window, and (a, c, c) for the second.

Id	1	2	3	4	5	6	7	8	9	10
1	a	e	b	e	c	e	a	c	c	e
2	c	d	a	b	c	a	b	b	c	c
3	a	a	b	b	c	a	c	d	c	b
4	d	a	b	c	b	d	c	a	c	c
5	a	b	b	b	c	b	a	c	d	c
	$k=1$					$k=2$				
	(a)					(a)				
	(a,b)					(a,c)				
	(a,b,c)					(a,c,c)				
	(a,c)					(c)				
	(b)					(c,c)				
	(b,c)									
	(c)									

(A)

Id	1	2	3	4	5	6	7	8	9	10
1	a	e	b	e	c	e	a	c	c	e
2	c	d	a	b	c	a	b	b	c	c
3	a	a	b	b	c	a	c	d	c	b
4	d	a	b	c	b	d	c	a	c	c
5	a	b	b	b	c	c	a	c	d	c
	$k=1$		$k=2$		$k=3$		$k=4$		$k=5$	
	(a)		(b)		(c)		(c)		(c)	

(B)

Figure 7.3: Generated features with $\sigma = 0.8$ using (A) 2 windows, and (B) 5 windows.

Our framework observes the data at multiple levels of granularity by using multiple window sizes. In other words, we repeat the previous steps several times, each time using a different number of windows $n \in \mathcal{N}$, with $\mathcal{N}$ denoting the set of window sizes employed in this step. The motivation behind considering different window sizes is that a particular window size could be useful to detect one class, but irrelevant for another. The final set of patterns is, hence, the following:

$$\mathcal{F} = \bigcup_{c \in C} \bigcup_{n \in N} \bigcup_{k \in [1,n]} f_{max}(\mathcal{P}, \mathcal{D}_1, \prec, \sigma, c, n, k).$$

where c is the sequence class, n is the total number of windows, and k is the window index. Any frequent pattern $f \in \mathcal{F}$ is a triple $f = (p, k, n)$ where $F.p$ is a frequent sequential pattern appearing in window k, given that the time-line is divided into n windows.

In spite of the large number of frequent patterns obtained using this approach, many patterns can be removed during the feature-selection phase that follows.

7.3.3 Feature selection and model construction

We will now use the set of features $\mathcal{F}$ generated in the previous phase to build a multi-granularity sequence classification model. For this purpose, we will use the validation set $\mathcal{D}_2$. Firstly, we map $\mathcal{D}_2$ to a binary feature matrix that is used for building the classification model. For each sequence s in $\mathcal{D}_2$, we check, for each feature $f = (e, k, n) \in F$, whether s supports e, using the following function:

$$check(s, f) = \begin{cases} 1 & \text{if } e \prec s \\ 0 & \text{otherwise} \end{cases}$$

Assume that $\mathcal{D}_2 = \{s_1, \ldots, s_n\}$ and $\mathcal{F} = \{f_1, \ldots, f_m\}$. Then, this step will result in an $n \times m$ Boolean matrix M, where $M_{i,j} = check(s_i, f_j)$, $\forall\ i \in [1, n]$ and $j \in [1, m]$. This is in fact the matrix that contains all the features (Boolean) of the validation set. The process of building the classifier consists of the following steps:

Feature selection: a feature-selection algorithm is applied to M, so that only the most discriminative class features are selected. Depending on the classification method at hand, we may use different alternatives, such as SVM feature selection [132] or Random Forest [13].

Model construction: using the selected set of features from the previous step and the validation set $\mathcal{D}_2$, we build a classifier for the transactional sequences. Our framework provides us with the flexibility to use any supervised learning algorithm at this step.

7.4 Experiments

As in Chapter 6, we studied the performance of our framework on a dataset representing on-line interactions between students while undertaking projects. The task

was to correctly detect the team-member role of each student. The full details of the dataset used can be found in Chapter 6/Section 6.3.1. Below, we first describe the details of the experimental setting, followed by a discussion of the obtained results.

7.4.1 Setup

In order to generate stable results, we ran the experiments several times. For each run, we used two projects as the training dataset $\mathcal{D}_1$ and the remaining six projects as the validation dataset $\mathcal{D}_2$. Since dataset $\mathcal{D}$ consists of 8 projects, this yields 28 different ways to divide the full dataset into the training and testing datasets. The overall results are reported as the average obtained over all 28 executions. The four evaluation metrics used were precision, recall, F-measure, and area under the ROC.

Nearest-Neighbour classifier. Each student in the dataset is considered as a sequence of events. An event is either *reading* or *sending* a message, hence $\mathcal{E} = \{reading, sending\}$. We experimented with two functions: (1) edit-distance (NN-ED), which counts the minimum number of edit operations (insertion, deletion, substitution) required to transform one sequence into another, and (2) Smith-Waterman (NN-SW), which performs local sequence alignment.

Multi-granularity pattern-based classifier. Each student is modelled as a sequence of transactions, and each transaction represents one day. This means that the event space becomes $\mathcal{E} = \{reading, sending, gap\}$, where the *gap* event means that no reading or sending activity was carried out by the student in a particular transaction (day). We used a minimum frequency threshold of $\sigma = 0.8$ when running the SPAM algorithm. As mentioned before, we generated the set of frequent sequences by dividing the time-line of the projects in the dataset $\mathcal{D}_1$ into n windows of equal size, applying the SPAM algorithm on each of these n windows. We explored different levels of time granularity by iterating this process using four different window sizes, i.e., $\mathcal{N} = \{10, 15, 20, 25\}$.

For training and testing our classifier, we used Weka [134] for all experiments. For the feature-selection step, we investigated two algorithms:

- SVM [132]: returns a ranked list of features. In this case, we denote the classifier by MG-SVM-l, where we consider the top l features when building the classifier in the next step. We evaluated the results obtained for $l \in \{10, 20, 30, 40, 50\}$.
- RandomForest [13]: returns a subset of features to be used for training the classifier. We denote this classifier by MG-RF.

After selecting the features, the final model can be built using any supervised-learning algorithm. In our experiments, we used the Random Forest classifier.

Finally, we used 10-fold cross validation by partitioning the validation dataset $\mathcal{D}_2$ consisting of 6 projects into 10 equal subsets. The values of the evaluation metrics were based on their means over all runs.

7.4.2 Experimental results

The results for the MG-SVM classifiers when using various top-l features for $l \in \{10, 20, 30, 40, 50\}$ are highly similar. As the results for $l = 40$ are slightly better than the others, we used them (MG-SVM-40) when comparing to MG-RF and the two baselines NN-SW and NN-ED in Table 7.2 and Figure 7.4. Clearly, MG-RF returns a result very close to that obtained by MG-SVM-40, with an F-measure of 0.886 compared to 0.912. However, both MG-RF and MG-SVM-40 are substantially better than the two baseline classifiers, where the F-measure scores were only 0.313 for NN-SW and 0.567 for NN-ED. The reason behind the difference in these results is that NN-ED performs global sequence alignment, and hence local structure within the classes may be hidden by the global structure captured by NN-ED. On the other hand, NN-SW performs local alignment, hence favouring local structural similarity between the classes.

	NN-SW	NN-ED	MG-RF	MG-SVM-40
precision	0.548	0.611	0.886	0.914
recall	0.387	0.563	0.890	0.915
F-measure	0.313	0.567	0.886	0.912

Table 7.2: Average precision, recall, and F-measure using NN-SW, NN-ED, MG-RF and MG-SVM-40.

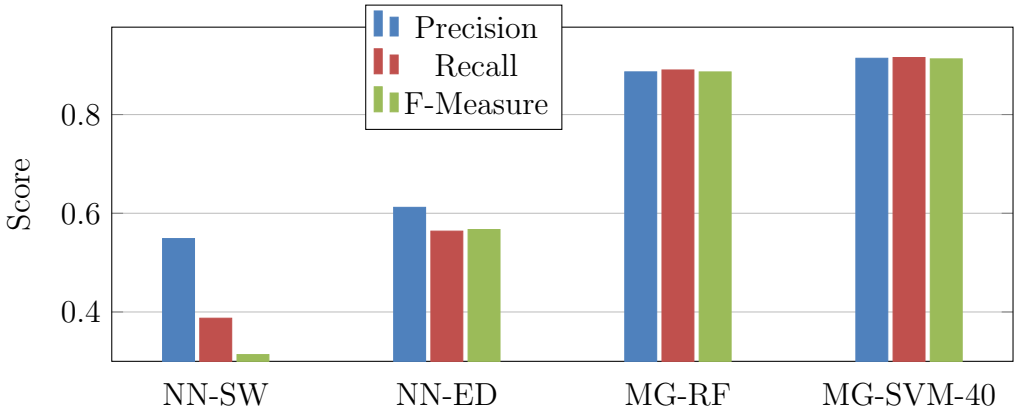

Figure 7.4: Comparison between NN-SW, NN-ED, MG-RF and MG-SVM-40

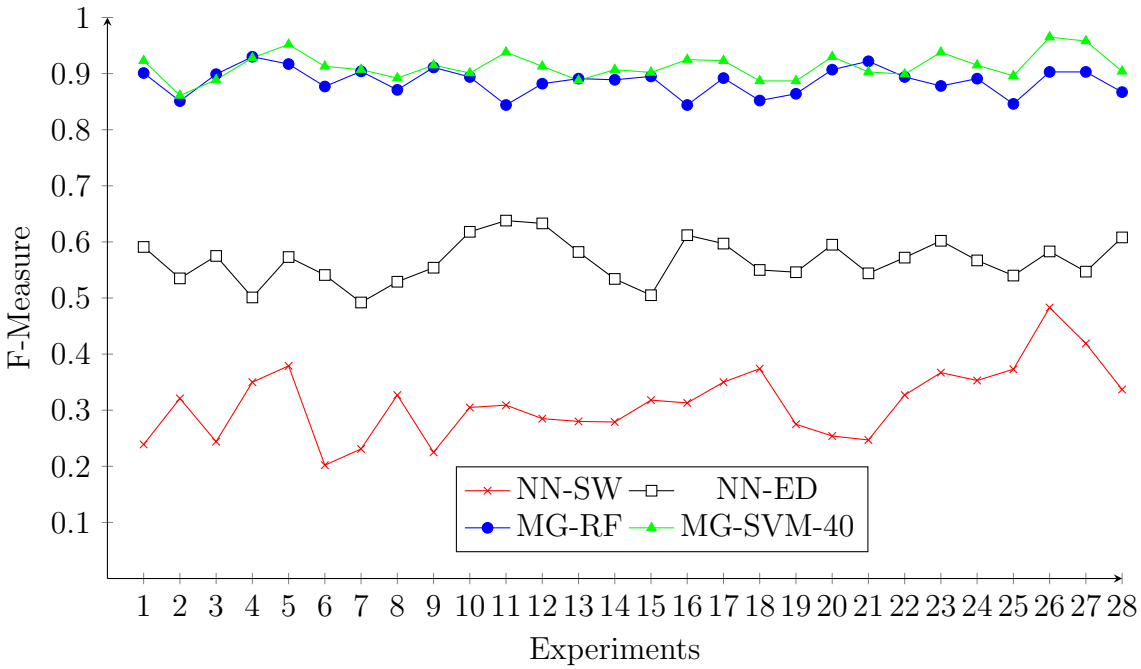

Figure 7.5: F-measure per experiment using NN-SW, NN-ED, MG-RF, and MG-SVM-40.

When considering the individual results of the 28 experiments, a similar pattern emerges, as shown in Figure 7.5. In all experiments, NN-ED performs better than NN-SW, while the performance of MG-SVM-40 and MG-RF are quite similar and considerably better than both NN-ED and NN-SW. When algorithm execution time is a factor, then MG-SVM is preferable to MG-RF; each experiment using MG-SVM takes about 15 minutes compared to more than 60 minutes for MG-RF. In addition, the ROC scores in Table 7.3 show that, for all the top-l features we considered, MG-SVM is better than MG-RF. On the other hand, using MG-SVM is not recommended when there are many redundant features. If the set of features generated from the first phase includes many redundant features, MG-RF is more effective in classifying the transactional sequences since it selects only a subset of features. This subset will include only non-redundant features if those features are discriminative for classifying the sequences.

	MG-RF	SVM-10	SVM-20	SVM-30	SVM-40	SVM-50
ROC	0.942	0.957	0.967	0.971	0.973	0.973

Table 7.3: The average ROC scores using MG-RF and all MG-SVM variants.

7.5 Summary

We proposed a multi-granularity framework for classifying sequences of discrete events which employs frequent sequential patterns at different time granularity levels as distinctive class features. We applied our framework to detect the roles of students working in a project and interacting via an online asynchronous communication. We approached the problem as a sequence classification problem in which students can be represented by sequences of their online activities. Our results demonstrate the superiority of the multi-granularity pattern-based classifier against the baseline Nearest-Neighbour classifier which built using a similarity-based function. Our multi-granularity pattern-based classifier can detect the correct student role on average more than 90% of the time.

Chapter 8

Conclusions and Future Work

In this thesis, we have addressed two problems within the area of social network analysis: (1) detecting hierarchical ties between users, and (2) inferring users' hierarchical roles. In both cases, we analysed the interaction network between users to address the problem. Our main focus was to demonstrate that taking into account the time-dimension of interactions (including at different levels of granularity) improves the quality of results obtained compared to when temporal aspects are ignored.

In this chapter, we provide a brief summary of the contributions achieved in this thesis. Then, we present possible directions for our research in the immediate future.

8.1 Summary of Thesis

The contributions of this thesis are summarised in the following two subsections.

8.1.1 Detecting hierarchical ties

We studied the problem of inferring hierarchical ties between users in online interaction networks. We approached the problem as a ranking problem. In other

words, to infer the hierarchical tie between x and y, where y is the superior of x, we rank all users other than x according to their scores calculated from analysing the interaction network. The better the ranking approach, the higher the superior y is ranked.

Based on the promising results we obtained when we employed link-analysis ranking methods such as PageRank and Rooted-PageRank (RPR) (Chapter 3), we proposed three novel approaches which took the temporal dimension of the interactions into account:

- **Time-F** (Chapter 4): This approach was built using a time-function whose definition depended on which periods of interactions in the application were considered to be important. Higher weights were assigned to the interactions that occurred in periods considered to be more important. For example, in academia, a person's early papers are more likely to be co-authored with their advisor than later papers. In such a case, higher weights were given to early interactions (shared papers).
- **Filter-and-Refine (FiRe)** (Chapter 4): This was a hybrid approach based on both RPR and Time-F. We filtered the list of users ranked according to Time-F by considering only the top k users. Then we reordered the filtered list using RPR scores.
- **Time-sensitive Rooted PageRank (T-RPR)** (Chapter 5): This method was designed to capture the dynamics of RPR scores of the network users which considerably improved the detection of hierarchical relationships compared to Rooted-PageRank. The method starts by dividing the time period of interactions into n time-slots. Then RPR is applied n times, once for each interaction graph corresponding to one of the n time-slots. This generates n ranked lists of users which are aggregated to generate the final ranked list. We proposed two approaches to aggregation, one based on a simple weighted average and the other based on voting.

Our extensive experimental evaluation on two real large datasets, the Enron e-mail network and a co-authorship network, provides reasonable empirical justification for our claim that "time matters" in detecting hierarchical ties. The results of detecting hierarchical ties in both datasets using our time-based methods including Time-F, FiRe and T-RPR, were substantially superior to link-analysis methods such as PageRank, Degree Centrality, and Rooted-PageRank, which do not consider the temporal aspect of interactions. Of our methods, T-RPR performed best overall when it was applied on an undirected interaction graph. For example, using recall for evaluation, T-RPR scored 0.58 in detecting manager-subordinate Enron relationships, compared to only 0.43, 0.34 and 0.3 in the best case using FiRe, Time-F and RPR respectively.

8.1.2 Inferring hierarchical roles

We studied the problem of inferring hierarchical roles in an educational environment, where students are interacting via an online communication medium to accomplish a project they are working on. By analysing their online interaction, we aimed to detect the hierarchical role of each student in the project. We addressed the problem using two approaches as follows:

- **Supervised Learning** (Chapter 6): We used features including individual attributes related to messages sent and read, as well as reply-based features provided by two social network analysis measures, namely PageRank and HITs. We also used time-based features to capture the dynamics of the quantitative features and how they change over time. We applied a number of feature-selection algorithms to find the best set of features which were used later to train a classification model. We tested a number of classification algorithms that belong to different categories, such as Tree-based algorithms and Bayes-based algorithms.

- **Sequence Classification** (Chapter 7): Given that each student can be represented as a sequence of his/her online activities, we proposed a multi-granularity feature-based framework for classifying sequences of discrete events. Our framework employed frequent sequential patterns at different time granularity levels as distinctive class features. In this way, we captured the inherent temporal structure of the sequences by mining frequent sequential patterns in different window sizes. The irrelevant features were filtered out by applying a feature selection algorithm on the full set of features. Then, the selected features were used to construct the classification model.

We evaluated our approach on real educational data collected from asynchronous communication tools in which students can interact to collaborate on a project they are working on. Each student can play one of three possible roles, project manager, executive or team member. Our empirical results showed that, in supervised learning, the reply-based features were selected as distinctive features for detecting students' roles. Moreover, many time-based features related to the first stages of working on the project, were selected as relevant features. This highlighted the importance of the initial days of launching a project in detecting the members' roles. Generally, all classification algorithms we used achieved an F-measure above 0.9 when we used the filtered set of features.

Similarly, our multi-granularity feature-based framework was effective in detecting students' roles with more than 0.9 scored for F-measure, compared to only 0.57 using a baseline similarity-based model. The way we built this framework makes it flexible and applicable to any domain in the area of classifying sequences of discrete events.

8.2 Limitations and Constraints

Below we identify some limitations of our work:

- In Time-F, FiRe and T-RPR we captured the temporal dimension of interactions using an application-dependent methodology. For example, when detecting advisor-advisee relationships, we assigned higher weights to early papers since advisees are expected to interact more with their advisors in the early stages of their academic activity. On the other hand, all exchanged emails were assigned equal weights in the case of detecting manager-subordinate relationships. Given a new dataset without any prior knowledge about the nature of communications, how do we define an appropriate time function? One possible solution would be to learn the weights using a supervised learning model.

- In all our time-based approaches, the size of time slot which returns the best results is not known beforehand. For example, a day, week, month or any other temporal duration can be used as one time slot in T-RPR. Further studies are required to discover the ideal size of time slots for each application.

- Running Rooted-PageRank (RPR) on a large graph can be very costly in terms of computational resources and the time taken by the algorithm. For example, RPR took about one hour when applied on the co-author network which consists of more than one million nodes. To address this issue, pruning techniques to reduce the size of the graph should be investigated.

- The approaches we proposed to detect the roles of users are based on the assumption that all users play their assigned roles from the start to the end of the project they are working on. However, in some cases users can play different roles according to the stage of project development. We need to extend our approaches in order to capture the dynamics of roles in such cases.

8.3 Other Directions for Future Work

In this section, we discuss further future directions for our research. We present each group of directions according to the problems we addressed in this thesis.

8.3.1 Detecting hierarchical ties

We plan to validate the obtained results further using different datasets. Also, since we addressed this problem using an unsupervised learning model, we intend to explore it also using a supervised learning model. Moreover, we aim to consider some features that were not taken into account in our methodology in order to improve our results. These include the following:

- Analysing the content of interactions: The communication between two users may contain interactions that are not related to their relationship. By analysing the content of interactions, we can filter out interactions that cause noise in the results. For example, by analysing the text of emails exchanged between employees, we can keep only the emails that are related to their work, and drop the others.
- The order of users involved in each interaction: For example, in bibliographic networks, an advisee is often the first author in a paper coauthored with his/her advisor. In addition, a subordinate often cc's their manager to keep them in the loop.
- The average response time between a user's interactions: In some cases, such as email networks, a subordinate is expected to respond faster to an email sent from his/her manager, compared to his/her response to emails from others. We intend to explore the average response time between messages and their replies to detect managers from their subordinates' replies.

In the models discussed in this thesis, the aforementioned features can be captured and considered in calculating the edge weights in the interaction networks. In supervised learning, these features can be used as attributes to train the classification model.

Relating to our Filter-and-Refine (FiRe) approach built using RPR and Time-F approaches, it is worth investigating the results obtained using other methods in each of the filtering and refining steps.

8.3.2 Inferring hierarchical roles

For our supervised learning model, we plan to validate the obtained results further using different educational datasets. We also intend to test our proposed model on other educational mining problems such as predicting the final marks of students from their online interactions. It would also be interesting to analyse message content as a way to improve the prediction of team member roles by filtering out messages unrelated to the project.

For our multi-granularity sequence classification framework, directions for future work include the validation of the proposed framework in other domains. In addition, we plan to compare results obtained by our framework with other competitive approaches proposed in the field.

It is also worth addressing this problem using an unsupervised learning approach (clustering) for the cases when the number of possible roles is not known beforehand. Premised on an intuition that users of similar role experience similar interaction features, we can group similar people in clusters based on their online interactions.

Appendix A

Time-based Methods

In this appendix, we present the results of detecting managers in the Enron dataset using the Time-F and FiRe methods. First, we present the results of the Time-F and FiRe approaches when the time-slots are months, the filter step is RPR and the refine step is Time-F. Then we present the full results when we use week-based time-slots.

A.1 FiRe results (month-based time-slots)

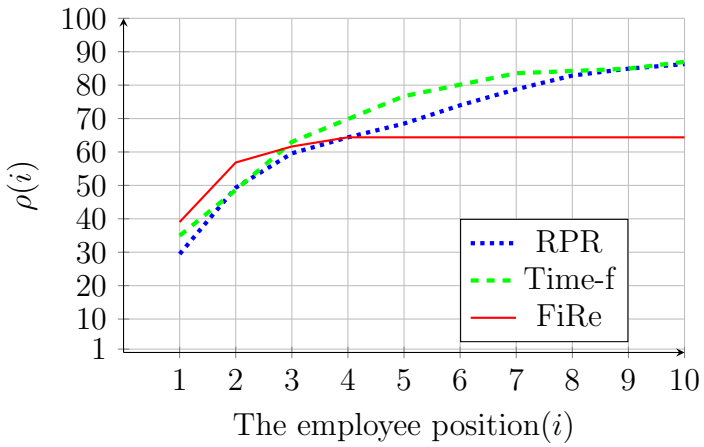

Figure A.1: Results of RPR, Time-F and FiRe (RPR is the filter and Time-F is the refiner) on the Enron dataset using month-based time-slots.

	$\rho(i)$		
i	Rooted-PageRank	Time-F	FiRe($k=4$)
1	29.45	34.93	39.04
2	49.31	48.63	56.84
3	59.58	63.01	61.64
4	64.38	69.86	64.38
5	68.49	76.71	64.38
6	73.97	80.13	64.38
7	78.76	83.56	64.38
8	82.87	84.24	64.38

Table A.1: Percentage Results for Rooted-PageRank, Time-F and FiRe (RPR is the filter and Time-F is the refiner) on Enron dataset using month-based time-slots.

cut	$\rho(i)$				
off	$=1$	$i=2$	$i=3$	$i=4$	$i=5$
2	35.61	49.31	49.31	49.31	49.31
3	39.04	57.53	59.58	59.58	59.58
4	39.04	56.84	61.64	64.38	64.38
5	39.72	58.21	63.69	67.12	68.49
6	37.67	56.84	68.49	72.60	72.60

Table A.2: FiRe (RPR is the filter and Time-F is the refiner) month-based results for Enron dataset using various cut-off values k.

A.2 FiRe results (week-based time-slots)

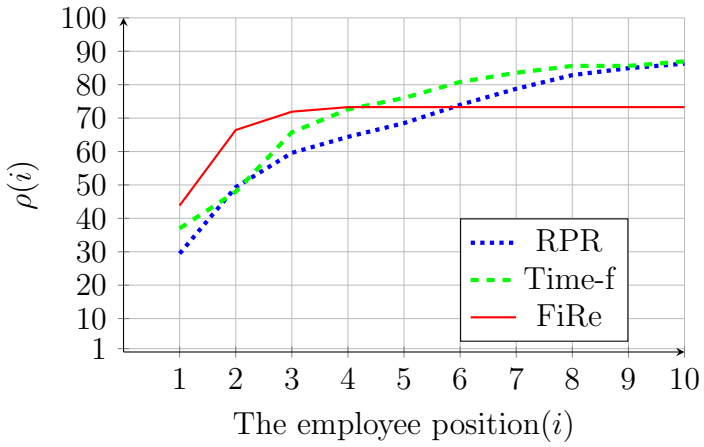

Figure A.2: Results of RPR, Time-F and FiRe (Time-F is the filter and RPR is the refiner) on the Enron dataset using week-based time-slots.

	$\rho(i)$		
i	Rooted-PageRank	Time-F	FiRe($k=4$)
1	29.45	36.98	43.83
2	49.31	47.94	66.43
3	59.58	65.75	71.91
4	64.38	72.60	73.28
5	68.49	76.02	73.28
6	73.97	80.82	73.28
7	78.76	83.56	73.28
8	82.87	85.61	73.28

Table A.3: Percentage Results for Rooted-PageRank, Time-F and FiRe (Time-F is the filter and RPR is the refiner) on Enron dataset using week-based time-slots.

cut	$\rho(i)$				
off	$i=1$	$i=2$	$i=3$	$i=4$	≤ 5
2	36.98	48.63	48.63	48.63	48.63
3	43.83	60.27	65.75	65.75	65.75
4	43.83	66.43	71.91	73.28	73.28
5	40.41	63.013	73.97	76.02	77.39
6	41.09	63.01	72.60	78.76	80.82

Table A.4: FiRe (Time-F is the filter and RPR is the refiner) week-based results for Enron dataset using various cut-off values k.

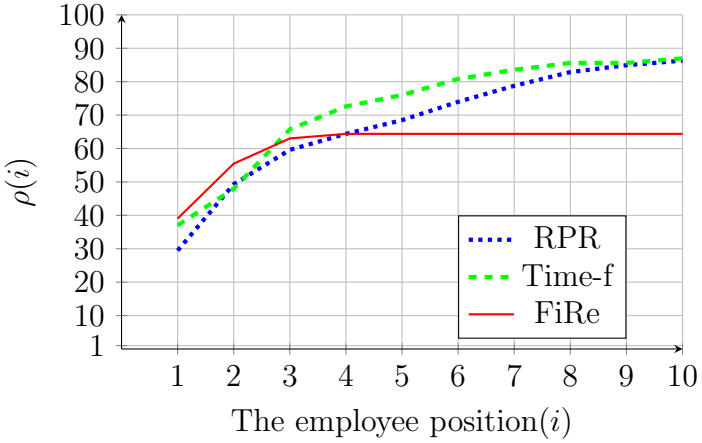

Figure A.3: Results of RPR, Time-F and FiRe (RPR is the filter and Time-F is the refiner) on the Enron dataset using week-based time-slots.

	$\rho(i)$		
i	Rooted-PageRank	Time-F	FiRe($k = 4$)
1	29.45	36.98	39.04
2	49.31	47.94	55.47
3	59.58	65.75	63.01
4	64.38	72.60	64.38
5	68.49	76.02	64.38
6	73.97	80.82	64.38
7	78.76	83.56	64.38
8	82.87	85.61	64.38

Table A.5: Percentage Results for Rooted-PageRank, Time-F and FiRe (RPR is the filter and Time-F is the refiner) on Enron dataset using week-based time-slots.

cut	$\rho(i)$				
off	$i = 1$	$i = 2$	$i = 3$	$i = 4$	≤ 5
2	36.30	49.31	49.31	49.31	49.31
3	40.41	57.53	59.58	59.58	59.58
4	39.04	55.47	63.01	64.38	64.38
5	39.72	57.53	65.06	66.43	68.49
6	39.04	56.16	70.54	71.91	72.60

Table A.6: FiRe (RPR is the filter and Time-F is the refiner) week-based results for Enron dataset using various cut-off values k.

Appendix B

Detecting students' roles

In this appendix, we present the full results of applying our supervised model discussed in Chapter 6 in order to detect the roles of students working in a project. According to the number of windows used to generate the time-based features, we list two groups of results: (1) results using 10 non-overlapping windows, and (2) results using 20 non-overlapping windows.

In addition, according to the filtering technique we used to find the filtered set of features, we present the results of different cases. Let k be the number of top features considered for each feature-selection algorithm, and m be the number of algorithms (out of 10) that should select the feature within their top k in order for the feature to be placed in the filtered set. We considered the following cases:

Case	k	m
1	10	5
2	10	1
3	20	5
4	20	1
5	30	5
6	30	1

First we present the results using 10 time-based windows in Tables B.1 to B.4 and in Figures B.1 to B.4. Then, we present the results using 20 time-based windows in Tables B.5 to B.8 and in Figures B.5 to B.8.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered appeared 5 times in top 10	Filtered appeared once in top 10	Filtered appeared 5 times in top 20	Filtered appeared once in top 20	Filtered appeared 5 times in top 30	Filtered appeared once in top 30
Bayes-based	BayesNet	Precision	0.71	0.829	0.889	0.91	0.922	0.893	0.894	0.903	0.892
		Recall	0.722	0.825	0.881	0.907	0.918	0.881	0.887	0.897	0.887
		F-Measure	0.715	0.823	0.883	0.908	0.919	0.884	0.889	0.899	0.888
	NaiveBayes	Precision	0.836	0.849	0.857	0.918	0.923	0.905	0.908	0.895	0.891
		Recall	0.84	0.835	0.84	0.918	0.923	0.902	0.907	0.892	0.881
		F-Measure	0.834	0.836	0.845	0.918	0.923	0.903	0.908	0.893	0.884

Table B.1: Results of Bayes-based algorithms using 10 time-based windows

Figure B.1: F-measure scores for Bayes-based algorithms using 10 time-based windows.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered appeared 5 times in top 10	Filtered appeared once in top 10	Filtered appeared 5 times in top 20	Filtered appeared once in top 20	Filtered appeared 5 times in top 30	Filtered appeared once in top 30
Function-based	Logistic	Precision	0.795	0.865	0.805	0.883	0.886	0.861	0.824	0.855	0.808
		Recall	0.814	0.866	0.799	0.887	0.887	0.861	0.825	0.851	0.809
		F-Measure	0.798	0.865	0.799	0.884	0.886	0.86	0.824	0.849	0.807
	MultilayerPerceptron	Precision	0.782	0.867	0.909	0.872	0.915	0.884	0.918	0.915	0.918
		Recall	0.784	0.866	0.907	0.866	0.912	0.887	0.918	0.912	0.918
		F-Measure	0.782	0.865	0.908	0.868	0.913	0.885	0.917	0.913	0.918
	RBFNetwork	Precision	0.782	0.83	0.854	0.918	0.933	0.898	0.906	0.876	0.891
		Recall	0.789	0.835	0.845	0.918	0.933	0.897	0.907	0.871	0.892
		F-Measure	0.784	0.832	0.849	0.917	0.933	0.897	0.906	0.873	0.891
	SMO	Precision	0.669	0.687	0.917	0.862	0.892	0.89	0.907	0.89	0.913
		Recall	0.778	0.794	0.918	0.866	0.892	0.892	0.907	0.892	0.912
		F-Measure	0.719	0.736	0.917	0.856	0.886	0.886	0.907	0.888	0.912

Table B.2: Results of Function-based algorithms using 10 time-based windows

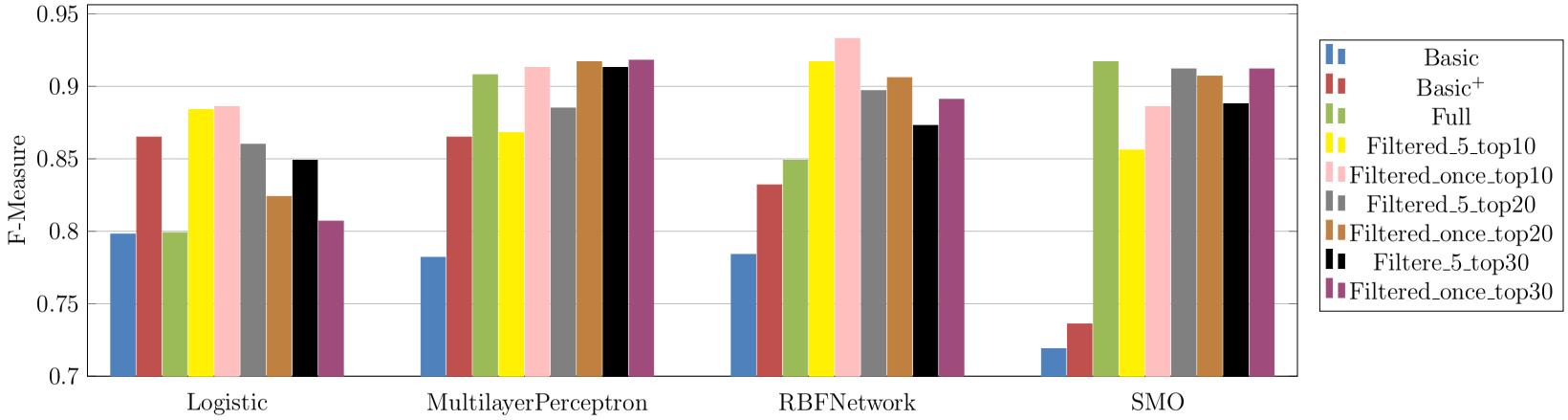

Figure B.2: F-measure scores for Function-based algorithms using 10 time-based windows.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered appeared 5 times in top 10	Filtered appeared once in top 10	Filtered appeared 5 times in top 20	Filtered appeared once in top 20	Filtered appeared 5 times in top 30	Filtered appeared once in top 30
Rules-based	DTNB	Precision	0.713	0.754	0.883	0.872	0.877	0.892	0.9	0.9	0.886
		Recall	0.753	0.742	0.887	0.871	0.861	0.887	0.897	0.897	0.892
		F-Measure	0.717	0.743	0.885	0.871	0.864	0.889	0.898	0.898	0.888
	JRip	Precision	0.729	0.74	0.899	0.867	0.878	0.885	0.901	0.848	0.904
		Recall	0.727	0.742	0.902	0.876	0.881	0.887	0.902	0.851	0.897
		F-Measure	0.726	0.74	0.9	0.87	0.879	0.886	0.901	0.849	0.899
	NNge	Precision	0.762	0.813	0.85	0.862	0.859	0.845	0.862	0.838	0.853
		Recall	0.768	0.814	0.84	0.861	0.856	0.845	0.856	0.84	0.845
		F-Measure	0.763	0.803	0.832	0.86	0.854	0.842	0.848	0.83	0.837
	Ridor	Precision	0.785	0.76	0.872	0.874	0.894	0.881	0.878	0.869	0.888
		Recall	0.794	0.758	0.876	0.871	0.892	0.881	0.876	0.871	0.887
		F-Measure	0.787	0.758	0.873	0.872	0.893	0.881	0.877	0.87	0.887

Table B.3: Results of Rule-based algorithms using 10 time-based windows

Figure B.3: F-measure scores for Rule-based algorithms using 10 time-based windows.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered appeared 5 times in top 10	Filtered appeared once in top 10	Filtered appeared 5 times in top 20	Filtered appeared once in top 20	Filtered appeared 5 times in top 30	Filtered appeared once in top 30
Tree-based	BFTree	Precision	0.754	0.753	0.896	0.875	0.88	0.884	0.897	0.912	0.885
		Recall	0.758	0.763	0.902	0.881	0.887	0.892	0.902	0.908	0.892
		F-Measure	0.756	0.756	0.896	0.878	0.882	0.885	0.898	0.927	0.887
	J48	Precision	0.731	0.747	0.893	0.888	0.882	0.91	0.933	0.916	0.895
		Recall	0.747	0.747	0.897	0.892	0.887	0.912	0.933	0.918	0.897
		F-Measure	0.725	0.747	0.894	0.89	0.884	0.911	0.933	0.917	0.896
	LADTree	Precision	0.777	0.737	0.88	0.862	0.895	0.878	0.885	0.88	0.876
		Recall	0.784	0.742	0.881	0.861	0.897	0.881	0.887	0.881	0.881
		F-Measure	0.777	0.74	0.879	0.861	0.896	0.878	0.885	0.879	0.878
	RandomForest	Precision	0.768	0.814	0.922	0.936	0.923	0.917	0.907	0.909	0.91
		Recall	0.768	0.82	0.923	0.938	0.923	0.918	0.907	0.912	0.912
		F-Measure	0.767	0.816	0.92	0.936	0.921	0.913	0.901	0.907	0.909

Table B.4: Results of Tree-based algorithms using 10 time-based windows

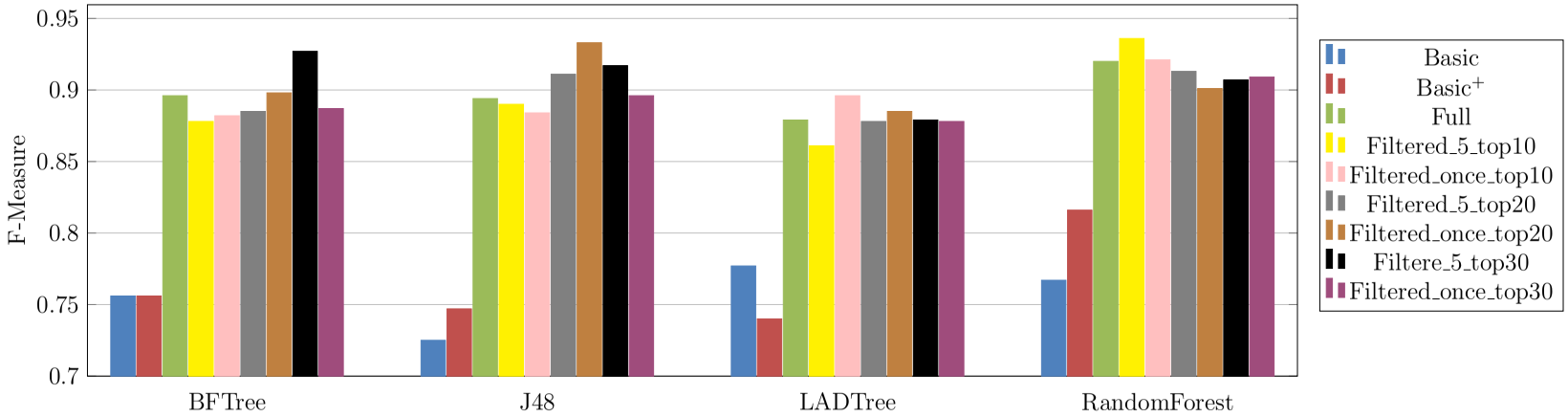

Figure B.4: F-measure scores for Tree-based algorithms using 10 time-based windows.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered appeared 5 times in top 10	Filtered appeared once in top 10	Filtered appeared 5 times in top 20	Filtered appeared once in top 20	Filtered appeared 5 times in top 30	Filtered appeared once in top 30
Bayes-based	BayesNet	Precision	0.71	0.829	0.912	0.916	0.921	0.919	0.918	0.912	0.918
		Recall	0.722	0.825	0.912	0.912	0.923	0.918	0.918	0.912	0.918
		F-Measure	0.715	0.823	0.912	0.913	0.922	0.918	0.918	0.912	0.918
	NaiveBayes	Precision	0.836	0.849	0.835	0.941	0.926	0.922	0.916	0.911	0.92
		Recall	0.84	0.835	0.804	0.938	0.923	0.918	0.912	0.907	0.918
		F-Measure	0.834	0.836	0.812	0.938	0.923	0.919	0.913	0.908	0.918

Table B.5: Results of Bayes-based algorithms using 20 time-based windows

Figure B.5: F-measure scores for Bayes-based algorithms using 20 time-based windows.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered appeared 5 times in top 10	Filtered appeared once in top 10	Filtered appeared 5 times in top 20	Filtered appeared once in top 20	Filtered appeared 5 times in top 30	Filtered appeared once in top 30
Function-based	Logistic	Precision	0.795	0.865	0.799	0.886	0.886	0.903	0.912	0.868	0.902
		Recall	0.814	0.866	0.799	0.886	0.887	0.902	0.912	0.866	0.902
		F-Measure	0.798	0.865	0.794	0.886	0.886	0.902	0.911	0.866	0.901
	MultilayerPerceptron	Precision	0.782	0.867	0.907	0.94	0.948	0.922	0.933	0.933	0.943
		Recall	0.784	0.866	0.907	0.938	0.948	0.923	0.933	0.933	0.943
		F-Measure	0.782	0.865	0.906	0.938	0.948	0.922	0.933	0.933	0.943
	RBFNetwork	Precision	0.782	0.83	0.813	0.959	0.941	0.94	0.946	0.936	0.908
		Recall	0.789	0.835	0.778	0.959	0.938	0.938	0.943	0.933	0.902
		F-Measure	0.784	0.832	0.79	0.959	0.939	0.939	0.944	0.934	0.904
	SMO	Precision	0.669	0.687	0.927	0.895	0.949	0.913	0.944	0.938	0.944
		Recall	0.778	0.794	0.928	0.897	0.948	0.912	0.943	0.938	0.943
		F-Measure	0.719	0.736	0.927	0.894	0.948	0.912	0.943	0.938	0.943

Table B.6: Results of Function-based algorithms using 20 time-based windows

Figure B.6: F-measure scores for Function-based algorithms using 20 time-based windows.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered appeared 5 times in top 10	Filtered appeared once in top 10	Filtered appeared 5 times in top 20	Filtered appeared once in top 20	Filtered appeared 5 times in top 30	Filtered appeared once in top 30
Rules-based	DTNB	Precision	0.713	0.754	0.916	0.921	0.911	0.921	0.906	0.902	0.916
		Recall	0.753	0.742	0.918	0.923	0.912	0.923	0.907	0.902	0.918
		F-Measure	0.717	0.743	0.916	0.922	0.911	0.921	0.906	0.901	0.916
	JRip	Precision	0.729	0.74	0.922	0.948	0.935	0.928	0.941	0.943	0.945
		Recall	0.727	0.742	0.923	0.948	0.933	0.928	0.938	0.943	0.943
		F-Measure	0.726	0.74	0.923	0.948	0.934	0.927	0.939	0.943	0.944
	NNge	Precision	0.762	0.813	0.858	0.915	0.911	0.911	0.9	0.896	0.881
		Recall	0.768	0.814	0.84	0.912	0.907	0.907	0.892	0.892	0.871
		F-Measure	0.763	0.803	0.842	0.912	0.908	0.907	0.893	0.893	0.873
	Ridor	Precision	0.785	0.76	0.909	0.928	0.917	0.917	0.917	0.912	0.897
		Recall	0.794	0.758	0.907	0.928	0.918	0.918	0.918	0.912	0.897
		F-Measure	0.787	0.758	0.908	0.927	0.917	0.917	0.917	0.912	0.897

Table B.7: Results of Rule-based algorithms using 20 time-based windows

Figure B.7: F-measure scores for Rule-based algorithms using 20 time-based windows.

Category	Classification Algorithm	Evaluation Method	Basic	Basic$^+$	Full	Filtered appeared 5 times in top 10	Filtered appeared once in top 10	Filtered appeared 5 times in top 20	Filtered appeared once in top 20	Filtered appeared 5 times in top 30	Filtered appeared once in top 30
Tree-based	BFTree	Precision	0.754	0.753	0.902	0.949	0.933	0.927	0.917	0.927	0.917
		Recall	0.758	0.763	0.902	0.948	0.933	0.928	0.918	0.928	0.918
		F-Measure	0.756	0.756	0.901	0.947	0.932	0.927	0.917	0.927	0.917
	J48	Precision	0.731	0.747	0.933	0.953	0.943	0.948	0.938	0.948	0.938
		Recall	0.747	0.747	0.933	0.954	0.943	0.948	0.938	0.948	0.938
		F-Measure	0.725	0.747	0.933	0.953	0.943	0.948	0.938	0.948	0.938
	LADTree	Precision	0.777	0.737	0.914	0.954	0.933	0.933	0.933	0.933	0.935
		Recall	0.784	0.742	0.912	0.954	0.933	0.933	0.933	0.933	0.933
		F-Measure	0.777	0.74	0.913	0.954	0.933	0.933	0.933	0.933	0.933
	RandomForest	Precision	0.768	0.814	0.914	0.954	0.974	0.948	0.948	0.942	0.947
		Recall	0.768	0.82	0.912	0.954	0.974	0.948	0.948	0.943	0.948
		F-Measure	0.767	0.816	0.906	0.954	0.974	0.948	0.948	0.943	0.948

Table B.8: Results of Tree-based algorithms using 20 time-based windows

Figure B.8: F-measure scores for Tree-based algorithms using 20 time-based windows.

Bibliography

[1] J. B. Agapito and A. Ortigosa. Detecting symptoms of low performance using production rules. In T. Barnes, M. C. Desmarais, C. Romero, and S. Ventura, editors, *Educational Data Mining*, pages 31–40. www.educationaldatamining.org, 2009.

[2] A. Agarwal, A. Omuya, A. Harnly, and O. Rambow. A comprehensive gold standard for the Enron organizational hierarchy. In *ACL (2)*, pages 161–165. The Association for Computer Linguistics, 2012.

[3] A. Al-Herz and M. Ahmed. Model-based web components testing: A prioritization approach. In *Software Engineering and Computer Systems*, volume 181 of *Communications in Computer and Information Science*, pages 25–40. Springer Berlin Heidelberg, 2011.

[4] F. Alba-Elías, A. González-Marcos, and J. Ordieres-Meré. An ict based project management learning framework. In *EUROCON, 2013 IEEE*, pages 300–306, July 2013.

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[6] S. Angeletou, M. Rowe, and H. Alani. Modelling and analysis of user behaviour in online communities. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein,

L. Kagal, N. F. Noy, and E. Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7031 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2011.

[7] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 429–435, New York, NY, USA, 2002. ACM.

[8] L. Backstrom and J. M. Kleinberg. Romantic partnerships and the dispersion of social ties: A network analysis of relationship status on facebook. *CoRR*, abs/1310.6753, 2013.

[9] A. Bakharia and S. Dawson. Snapp: A bird's-eye view of temporal participant interaction. In *Proceedings of the 1st International Conference on Learning Analytics and Knowledge(LAK)*, pages 168–173, New York, NY, USA, 2011. ACM.

[10] V. Batagelj and M. Cerinšek. On bibliographic networks. *Scientometrics*, 96(3):845–864, 2013.

[11] P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[12] R. Bouckaert. *Bayesian Network Classifiers in Weka for Version 3-5-6*. The University of Waikato, 2007.

[13] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] S. Brin and L. Page. The anatomy of large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

[15] M. Burke, L. A. Adamic, and K. Marciniak. Families on facebook. In E. Kiciman, N. B. Ellison, B. Hogan, P. Resnick, and I. Soboroff, editors, *ICWSM*. The AAAI Press, 2013.

[16] R. S. Burt. *Structural holes: The social structure of competition*. Harvard University Press, Cambridge, MA, 1992.

[17] O. Casquero, J. Portillo, R. Ovelar, M. Benito, and J. Romo. iple network: An integrated elearning 2.0 architecture from a university's perspective. *Interactive Learning Environments*, 18(3):293–308, 2010.

[18] F. Castro, A. Vellido, A. Nebot, and F. Mugica. *Evolution of Teaching and Learning Paradigms in Intelligent Environment (Studies in Computational Intelligence)*, volume 62, chapter Applying data mining techniques to e-learning problems, pages 183–221. Springer-Verlag, New York, 2007.

[19] J. Chan, C. Hayes, and E. Daly. Decomposing Discussion Forums using Common User Roles. In *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line*, 2010.

[20] J. Chan, C. Hayes, and E. M. Daly. Decomposing discussion forums and boards using user roles. In W. W. Cohen and S. Gosling, editors, *International AAAI Conference on Weblogs and Social Media*. The AAAI Press, 2010.

[21] C. Chen, M. Chen, and Y. Li. Mining key formative assessment rules based on learner profiles for web-based learning systems. In *Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies, ICALT*, pages 584–588, 2007.

[22] C. K. Cheng, D. E. Paré, L.-M. Collimore, and S. Joordens. Assessing the effectiveness of a voluntary on-line discussion forum on improving students'

course performance. *Computers and Education*, 56(1):253 – 261, 2011. Serious Games.

[23] M. D. Choudhury, H. Sundaram, A. John, and D. D. Seligmann. Dynamic prediction of communication flow using social context. In *HYPERTEXT 2008, Proceedings of the 19th ACM Conference on Hypertext and Hypermedia, Pittsburgh, PA, USA, June 19-21, 2008*, pages 49–54, 2008.

[24] N. A. Chuzhanova, A. J. Jones, and S. Margetts. Feature selection for genetic sequence classification. *Bioinformatics*, 14(2):139–143, 1998.

[25] K. Cios, R. Swiniarski, W. Pedrycz, and L. Kurgan. The knowledge discovery process. In *Data Mining*, pages 9–24. Springer US, 2007.

[26] G. Cobo, D. García-Solórzano, E. Santamaria, J. A. Morán, J. Melenchón, and C. Monzo. Modeling students' activity in online discussion forums: A strategy based on time series and agglomerative hierarchical clustering. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. C. Stamper, editors, *Educational Data Mining (EDM)*, pages 253–258. www.educationaldatamining.org, 2011.

[27] M. Cocea and S. Weibelzahl. Can log files analysis estimate learners' level of motivation?. In K.-D. Althoff and M. Schaaf, editors, *Proceedings of Lernen - Wissensentdeckung - Adaptivität (LWA2006)*, volume 1 of *Hildesheimer Informatik-Berichte*, pages 32–35. University of Hildesheim, Institute of Computer Science, 2006.

[28] W. W. Cohen. Fast effective rule induction. In *Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[29] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[30] M. Cristian and B. Dan. Testing attribute selection algorithms for classification performance on real data. In *Intelligent Systems, 2006 3rd International IEEE Conference on*, pages 581–586, 2006.

[31] E. David and K. Jon. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.

[32] J. Davis and S. Leinhardt. *The Structure of Positive Interpersonal Relations in Small Groups*. Darthmouth College, 1967.

[33] G. Dekker, M. Pechenizkiy, and J. Vleeshouwers. Predicting students drop out: A case study. In T. Barnes, M. C. Desmarais, C. Romero, and S. Ventura, editors, *Educational Data Mining*, pages 41–50. www.educationaldatamining.org, 2009.

[34] H. Deng, J. Han, M. R. Lyu, and I. King. Modeling and exploiting heterogeneous bibliographic networks for expertise ranking. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 71–80. ACM, 2012.

[35] M. Deshpande and G. Karypis. Evaluation of techniques for classifying biological sequences. In *Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 417–431, 2002.

[36] C. P. Diehl, G. Namata, and L. Getoor. Relationship identification for social network discovery. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 546–552. AAAI Press, 2007.

[37] G. Dong and J. Pei. *Sequence Data Mining*, volume 33 of *Advances in Database Systems*. Kluwer, 2007.

[38] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, 2 edition, 2000.

[39] D. Fiala. Time-aware pagerank for bibliographic networks. *J. Informetrics*, 6(3):370–388, 2012.

[40] D. Fiala, F. Rousselot, and K. Ježek. Pagerank for bibliographic networks. *Scientometrics*, 76(1):135–158, 2008.

[41] A. W. P. Fok, H. S. Wong, and Y. S. Chen. Hidden markov model based characterization of content access patterns in an e-learning environment. In *ICME*, pages 201–204. IEEE Computer Society, 2005.

[42] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression : A statistical view of boosting. *Annals of statistics*, 28(2):337–407, 2000.

[43] B. Gaines and P. Compton. Induction of ripple-down rules applied to modeling large databases. *Journal of Intelligent Information Systems*, 5(3):211–228, 1995.

[44] A. J. Girasoli and R. D. Hannafin. Using asynchronous av communication tools to increase academic self-efficacy. *Computers & Education*, 51(4):1676–1682, 2008.

[45] S. A. Golder and J. Donath. Social roles in electronic communities. In *Proc. of Association of Internet Researchers Conference*, 2004.

[46] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 403–412, New York, NY, USA, 2004. ACM.

[47] M. Gupte, P. Shankar, J. Li, S. Muthukrishnan, and L. Iftode. Finding hierarchy in directed online social networks. In *Proceedings of the 20th International*

Conference on World Wide Web, (WWW '11), pages 557–566, New York, NY, USA, 2011. ACM.

[48] M. Hall and E. Frank. Combining naive bayes and decision tables. In *Proceedings of the 21st Florida Artificial Intelligence Society Conference (FLAIRS)*, pages 318–319. AAAI press, 2008.

[49] M. Hammond. A review of recent papers on online discussion in teaching and learning in higher education. *Journal of Asynchronous Learning Networks*, 9(3):9–23, 2005.

[50] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[51] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW*, pages 517–526, 2002.

[52] I. Himelboim, E. Gleave, and M. A. Smith. Discussion catalysts in online political discussions: Content importers and conversation starters. *J. Computer-Mediated Communication*, 14:771–789, 2009.

[53] G. Holmes, B. Pfahringer, R. Kirkby, E. Frank, and M. Hall. Multiclass alternating decision trees. In *ECML*, pages 161–172. Springer, 2001.

[54] G.-J. Hwang, C.-C. Tsai, and S. J. Yang. Criteria, strategies and research issues of context-aware ubiquitous learning. *Educational Technology & Society*, 11(2):81–91, 2008.

[55] M. Jaber, P. Papapetrou, A. Gonzalez-Marcos, and P. Wood. Analysing online education-based asynchronous communication tools to detect students'roles. In *Proceedings of the 7th International Conference on Computer Supported Education (CSEDU)*, pages 416–424. SCITEPRESS, 2015.

[56] M. Jaber, P. Papapetrou, S. Helmer, and P. Wood. Using time-sensitive rooted pagerank to detect hierarchical social relationships. In *Advances in Intelligent Data Analysis XIII*, volume 8819 of *Lecture Notes in Computer Science*, pages 143–154. Springer International Publishing, 2014.

[57] M. Jaber, P. T. Wood, P. Papapetrou, and S. Helmer. Inferring offline hierarchical ties from online social networks. In *WWW Companion '14*, pages 1261–1266, 2014.

[58] S. B. Jagtap and K. B. G. Census data mining and data analysis using WEKA. *CoRR*, 2013.

[59] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 271–279, 2003.

[60] L. Kaján, A. Kertész-Farkas, D. Franklin, N. Ivanova, A. Kocsor, and S. Pongor. Application of a simple likelihood ratio approximant to protein sequence classification. *Bioinformatics*, 22(23):2865–2869, 2006.

[61] P. Kazienko, R. Michalski, and S. Palus. Social network analysis as a tool for improving enterprise architecture. In J. O'Shea, N. T. Nguyen, K. A. Crockett, R. J. Howlett, and L. C. Jain, editors, *KES-AMSTA*, volume 6682 of *Lecture Notes in Computer Science*, pages 651–660. Springer, 2011.

[62] A. Keleş, R. Ocak, A. Keleş, and A. Gülcü. Zosmat: Web-based intelligent tutoring system for teaching–learning process. *Expert Systems with Applications*, 36(2):1229–1239, 2009.

[63] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[64] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 285–289, 2000.

[65] T. M. Khan, F. Clear, and S. S. Sajadi. The relationship between educational performance and online access routines: Analysis of students' access to an online discussion forum. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge(LAK)*, pages 226–229, New York, NY, USA, 2012. ACM.

[66] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[67] B. Klimt and Y. Yang. The Enron corpus: A new dataset for email classification research. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Machine Learning: ECML 2004*, volume 3201 of *Lecture Notes in Computer Science*, pages 217–226. Springer, 2004.

[68] D. Kudenko and H. Hirsh. Feature generation for sequence categorization. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence (AAAI/IAAI)*, pages 733–738. American Association for Artificial Intelligence, 1998.

[69] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pages 611–617, New York, NY, USA, 2006. ACM.

[70] P. F. Lazarsfeld, B. Berelson, and H. Gaudet. *The People's Choice: How the Voter Makes Up His Mind in a Presidential Campaign.* Columbia University Press, 1948.

[71] S. le Cessie and J. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[72] J. Lerner. Role assignments. In U. Brandes and T. Erlebach, editors, *Network Analysis*, volume 3418 of *Lecture Notes in Computer Science*, pages 216–252. Springer, Berlin / Heidelberg, 2005.

[73] N. Lesh, M. J. Zaki, and M. Ogihara. Mining features for sequence classification. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 342–346, 1999.

[74] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370. ACM, 2010.

[75] C. Leslie, R. Kuang, and K. Bennett. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 2004.

[76] C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002.

[77] X. Li, B. L. 0001, and P. S. Yu. Time sensitive ranking with application to publication search. In *ICDM*, pages 893–898. IEEE Computer Society, 2008.

[78] X. Liu, P. Zhang, and D. Zeng. Sequence matching for suspicious activity detection in anti-money laundering. In *Proceedings of the IEEE International Workshops on Intelligence and Security Informatics (ISI)*, pages 50–61, 2008.

[79] S. Lonn and S. D. Teasley. Saving time or innovating practice: Investigating perceptions and uses of learning management systems. *Computers & Education*, 53(3):686–694, 2009.

[80] M. I. López, C. Romero, S. Ventura, and J. M. Luna. Classification via clustering for predicting final marks starting from the student participation in forums. In *Proceedings of the 5th International Conference on Educational Data Mining*, pages 148–151, 2012.

[81] M. Maia, J. Almeida, and V. Almeida. Identifying user behavior in online social networks. In *Proceedings of the 1st Workshop on Social Network Systems (SocialNets '08)*, pages 1–6, New York, NY, USA, 2008. ACM.

[82] P. Marsden. Measuring tie strength. *Social Forces*, 63:482–501, 1984.

[83] B. Martin. Instance-based learning : Nearest neighbor with generalization. Technical report, University of Waikato, 1995.

[84] N. Memon, J. J. Xu, D. L. Hicks, and H. Chen. Social network data mining: Research questions, techniques, and applications. In N. Memon, J. J. Xu, D. L. Hicks, and H. Chen, editors, *Data Mining for Social Network Data*, volume 12 of *Annals of Information Systems*, pages 1–7. Springer, 2010.

[85] R. Michalski, S. Palus, and P. Kazienko. Matching organizational structure and social network extracted from email communication. In W. Abramowicz, editor, *BIS*, volume 87 of *Lecture Notes in Business Information Processing*, pages 197–206. Springer, 2011.

[86] V. Muley and V. Acharya. *Genome-Wide Prediction and Analysis of Protein-Protein Functional Linkages in Bacteria*. Springer Briefs in Systems Biology. Springer, 2012.

[87] E. Nankani, S. Simoff, S. Denize, and L. Young. Supporting strategic decision making in an enterprise university through detecting patterns of academic collaboration. In *Information Systems: Modeling, Development, and Integration*,

Lecture Notes in Business Information Processing, pages 496–507. Springer Berlin Heidelberg, 2009.

[88] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33:2001, 1999.

[89] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[90] J. D. Novak and A. J. Cañas. The theory underlying concept maps and how to construct and use them. Technical Report IHMC CmapTools 2006-01 Rev 2008-01., Institute for Human and Machine Cognition, Florida, 2008.

[91] Office Of Government Commerce. *Managing Successful Projects with PRINCE2(^{TM})*. Office Of Government Commerce, 2009.

[92] N. Padhy, P. Mishra, and R. Panigrahi. The survey of data mining applications and feature scope. *CoRR*, abs/1211.5723, 2012.

[93] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[94] S. Palmer, D. Holt, and S. Bray. Does the discussion help? the impact of a formally assessed online discussion on final student results. *British Journal of Educational Technology*, 39(5):847–858, 2008.

[95] S. Palus, P. Bródka, and P. Kazienko. How to analyze company using social network? In *Knowledge Management, Information Systems, E-Learning, and Sustainability Research*, volume 111 of *Communications in Computer and Information Science*, pages 159–164. Springer, 2010.

[96] S. Palus, P. Bródka, and P. Kazienko. Evaluation of organization structure based on email interactions. *International Journal of Scientific and Engineering Research*, 2(1):1–13, 2011.

[97] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Comput.*, 3(2):246–257, 1991.

[98] N. Peirce, O. Conlan, and V. Wade. Adaptive educational games: Providing non-invasive personalised learning experiences. In *Second IEEE International Conference on Digital Games and Intelligent Toy Enhanced Learning*, pages 28–35, November 2008.

[99] C. Pinheiro. *Social Network Analysis in Telecommunications*. Wiley and SAS Business Series. Wiley, 2011.

[100] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, January 1998.

[101] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[102] R. Rabbany, M. Takaffoli, and O. R. Zaïane. Analyzing participation of students in online courses using social network analysis techniques. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. C. Stamper, editors, *Educational Data Mining (EDM)*, pages 21–30. www.educationaldatamining.org, 2011.

[103] R. Rallo, M. Gisbert, and J. Salinas. Using data mining and social networks to analyze the structure and content of educative online communities. In *Proceedings of 3rd Internactional Conference on Multimedia and Information & Communication Technologies in Education*, pages 1–10, 2005.

[104] P.-L. P. Rau, Q. Gao, and L.-M. Wu. Using mobile communication technology in high school education: Motivation, pressure, and learning performance. *Comput. Educ.*, 50(1):1–22, Jan. 2008.

[105] C. Reffay and T. Chanier. How social network analysis can help to measure cohesion in collaborative distance-learning. In *international conference on computer support for collaborative learning*, pages 343–352, 2003.

[106] P. Reyes and P. Tchounikine. Mining learning groups' activities in forum-type tools. In *Proceedings of th Conference on Computer Support for Collaborative Learning*, pages 509–513. International Society of the Learning Sciences, 2005.

[107] C. Romero, M.-I. López, J.-M. Luna, and S. Ventura. Predicting students' final performance from participation in on-line discussion forums. *Computer and Eduction*, 68:458–472, 2013.

[108] C. Romero and S. Ventura. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(6):601–618, 2010.

[109] M. Rowe, M. Fernandez, H. Alani, I. Ronen, C. Hayes, and M. Karnstedt. Behaviour analysis across different types of enterprise online communities. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 255–264, 2012.

[110] R. Rowe, G. Creamer, S. Hershkop, and S. J. Stolfo. Automated social hierarchy detection through email network analysis. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, WebKDD/SNA-KDD '07, pages 109–117, New York, NY, USA, 2007. ACM.

[111] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.

[112] H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

[113] J. Scott. *Social Network Analysis: A Handbook*. SAGE Publications, 2000.

[114] H. Shi. Best-first decision tree learning. Master's thesis, University of Waikato, Hamilton, NZ, 2007.

[115] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[116] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *International Conference on Advances in Social Networks Analysis and Mining, ASONAM*, pages 121–128, 2011.

[117] J. F. Superby, J. P. Vandamme, and N. Meskens. Determination of factors influencing the achievement of the first-year university students using data mining methods. In *Proceeding of International Conference of Intelligent Tutoring Systems. Workshop Educational Data Mining, Taiwan*, pages 1–8, 2006.

[118] J. Tang, T. Lou, and J. Kleinberg. Inferring social ties across heterogenous networks. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 743–752, New York, NY, USA, 2012. ACM.

[119] J. Tang, M. Musolesi, C. Mascolo, and V. Latora. Temporal distance metrics for social network analysis. In *Proceedings of the 2nd ACM Workshop on Online Social Networks*, WOSN '09, pages 31–36. ACM, 2009.

[120] J. Tang, M. Musolesi, C. Mascolo, V. Latora, and V. Nicosia. Analysing information flows and key mediators through temporal centrality metrics. In *Proceedings of the 3rd Workshop on Social Network Systems*, SNS '10, pages 3:1–3:6, New York, NY, USA, 2010. ACM.

[121] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, NY, USA, 2008. ACM.

[122] W. Tang, H. Zhuang, and J. Tang. Learning to infer social ties in large networks. In *Proceedings of the ECML/PKDD 2011*, pages 381–397, 2011.

[123] D. Trenholme and S. P. Smith. Computer game engines for developing first-person virtual environments. *Virtual Reality*, 126(3):181–187, 2008.

[124] B. Uzzi and J. Spiro. Collaboration and creativity: The small world problem. *American Journal of Sociology*, 111(2):447–504, 2005.

[125] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., 1995.

[126] G. E. Vlahos, T. W. Ferratt, and G. Knoepfle. The use of computer-based information systems by german managers to support decision making. *Information and Management*, pages 763–779, July 2004.

[127] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[128] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.

[129] L. Wei and E. Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 748–753, 2006.

[130] T. Weinberg. *The New Community Rules - Marketing on the Social Web.* O'Reilly, 2009.

[131] H. T. Welser, D. Cosley, G. Kossinets, A. Lin, F. Dokshin, G. Gay, and M. A. Smith. Finding social roles in wikipedia. In *iConference*, pages 122–129. ACM, 2011.

[132] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Advances in Neural Information Processing Systems 13*, pages 668–674. MIT Press, 2000.

[133] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *KDD*, pages 266–275, 2003.

[134] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[135] Z. Wu and C. Chen. User classification and relationship detecting on social network site. In *International Conference on Control, Automation and Systems Engineering (CASE)*, pages 1–4, 2011.

[136] Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *SIGKDD Explorations Newsletter*, 12(1):40–48, 2010.

[137] J. Yoo and J. Kim. Predicting learner's project performance with dialogue features in online Q&A discussions. In *Intelligent Tutoring Systems*, volume 7315 of *Lecture Notes in Computer Science*, pages 570–575. Springer Berlin Heidelberg, 2012.

[138] P. S. Yu, X. Li, and B. Liu. Adding the temporal dimension to search - a case study in publication search. *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 0:543–549, 2005.

[139] D. Zakrzewska. Cluster analysis for users' modeling in intelligent e-learning systems. In *New Frontiers in Applied Artificial Intelligence, Proceedings of 21st International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems(IEA/AIE)*, pages 209–214, 2008.