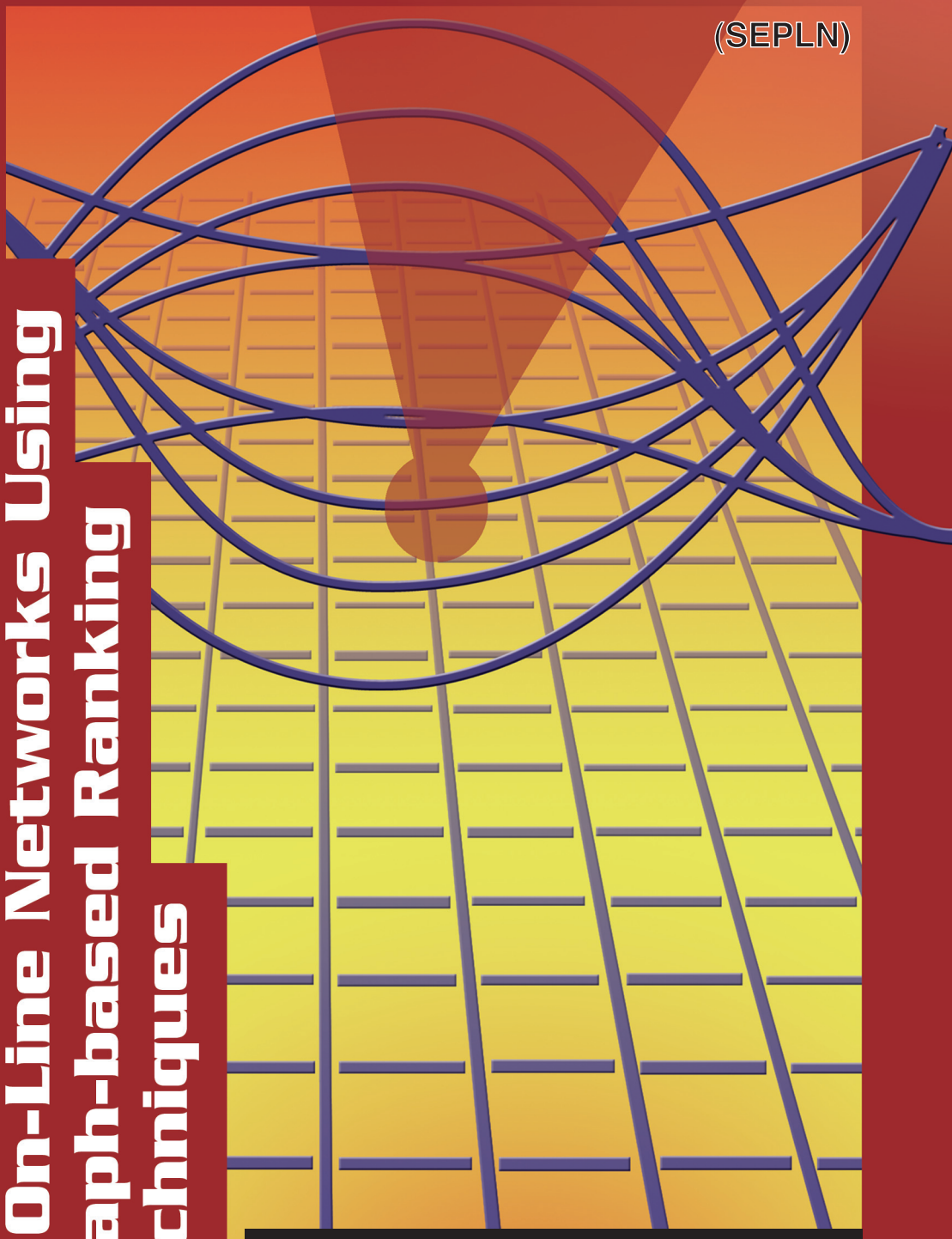
Colección de monografías: 12
Sociedad Española para el Procesamiento del Lenguaje Natural
(SEPLN)

Detection of Dishonest Behaviors in On-Line Networks Using Graph-based Ranking Techniques



F. Javier Ortega Rodríguez

Colección de Monografías de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN). Número 12

Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN)
http://www.sepln.org
secretaria.sepln@ujaen.es

Título: Detection of Dishonest Behaviors in On-Line Networks Using Graph-based Ranking Techniques
Autor: © Francisco Javier Ortega Rodríguez
ISBN: 978-84-616-5169-6
Depósito Legal: J 383-2013
Editores: L. Alfonso Ureña y Emilio Sanchís
Imprime: COMPOBELL, S.L.

Prólogo

La Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN) es una asociación científica sin ánimo de lucro creada en el año 1983 con el fin de promocionar y difundir todo tipo de actividades relacionadas con la enseñanza, investigación y desarrollo en el campo del procesamiento del lenguaje natural, tanto en el ámbito nacional como internacional.

Entre las actividades principales de la SEPLN figuran:

- La celebración de un congreso anual que sirve de punto de encuentro para los distintos grupos que trabajan en el área del procesamiento del lenguaje natural.
- La edición de la revista científica especializada *Procesamiento del Lenguaje Natural* de periodicidad semestral que cuenta con el Certificado de Revista Excelente de la Federación Española de Ciencia y Tecnología (FECYT).
- Un servidor Web (www.sepln.org) de referencia sobre procesamiento del lenguaje natural donde se encuentran en acceso abierto todas las publicaciones de la revista (journal.sepln.org).
- Una lista moderada de correo electrónico (SEPLN-L) que sirve como boletín de información periódica (quincenal) y como espacio de información y discusión para los miembros de la Asociación. La dirección para enviar cualquier comentario o aportación a la lista es sidsepln@si.ehu.es.
- Una Edición anual de Premios SEPLN a la Investigación en Procesamiento del Lenguaje Natural.

A esta XII Edición de los Premios SEPLN a la Investigación en Procesamiento del Lenguaje Natural se pudieron presentar a concurso trabajos monográficos de investigación originales e inéditos de cualquier extensión, escritos por un miembro de la SEPLN, y que no hubieran sido publicados o enviados a publicación con anterioridad a este concurso. Esta publicación presenta el trabajo premiado este año por la comisión evaluadora.

La Junta Directiva de la SEPLN, en nombre de la Sociedad, quiere dejar constancia aquí de la alta calidad de todas las obras presentadas a concurso en esta XII Edición de los Premios SEPLN, y animar a todos sus miembros a la participación en sus futuras ediciones. Con la publicación de estas contribuciones en su Colección de Monografías, la SEPLN podrá aportar lo mejor de sus esfuerzos a la actualización y divulgación de la investigación en el campo del procesamiento del lenguaje natural.

Junio 2013
Sociedad Española para el Procesamiento del Lenguaje Natural

Detection of Dishonest Behaviors in On-Line Networks Using Graph-based Ranking Techniques

Francisco Javier Ortega Rodríguez
javierortega@us.es

A todas aquellas personas que han contribuído a la realización de este trabajo, en especial a mi familia y amigos.

Contents

List of Figures

List of Tables

Abstract

This dissertation presents the research work developed in order to tackle the problem of dishonest behaviors in on-line networks by applying graph-based ranking algorithms. This work have been carried out in two complementary fields: the first one studies the dishonest behaviors that can be performed in the World Wide Web, and the second one is focused on the similar problems that suffers the Web 2.0.

In this way, we first focus our attention on the dishonest behaviors in the World Wide Web: the web spam. The main goal of web spam is to diverse web traffic to some specific web sites by cheating the web search engines in order to make these web sites as visible as possible. The key point in this task is the relevance of web pages, and the main problem is to avoid spam we pages to obtain a high relevance for the web search engine. This problem strongly affects the reliability of these systems, due to the harming effects caused in their performance in terms of providing the users with incorrect or undesirable contents. We propose a graph-based ranking algorithm that processes the web graph in order to build a ranking of web pages according to their relevance, demoting those web pages that are likely to be spam. The main novelty of our approach is the inclusion of knowledge from the textual content of the web pages into the graph-based algorithm. In this way, our approach can take into account not only the textual content of the web pages but also the hyperlinks between them.

On the other hand, we tackle the problem of trust and reputation in social networks. These systems must face several difficulties in order to avoid the negative effects caused by the dishonest behaviors that can be performed by their users. In this field, the relevance of the web pages has been changed by the reputation of the users, so the main problem is to detect the mechanisms used by malicious users in order to obtain a high reputation in the system. In this work we propose a graph-based ranking algorithm intended to detect and penalize the dishonest behaviors produced in a social network, in order to compute the trustworthiness of their users in such way that it will not be affected by the possible dishonest behaviors that can be performed.

The main novelty of this system is the ability of processing a social network with positive and negative relationships among their users, taking into account both types of relations to extract as much information as possible from the network.

The evaluation of both methods, the web spam detection technique and the trust and reputation system, shows that they are reliable in their corresponding tasks. We can conclude from these results that the enrichment of a graph-based ranking algorithm with content-based knowledge about the domain of the problem proposed in PolaritySpam, and the inclusion of the negative opinions of the users in a social network implemented in PolarityTrust are two useful techniques in order to deal with the dishonest behaviors in on-line networks.

Apart from the results, both techniques present some new ideas which can be suitable for these and other tasks where the content-based knowledge and the positive and negative relations in a network are relevant. Some of these applications are pointed out as well in this dissertation, highlighting the applicability of our research work

Chapter 1

Introduction

La meta è partire
(The goal is to depart)
Giuseppe Ungaretti

On-line networks have experienced a great expansion in the past few years. The Internet can now be accessed almost anywhere by numerous means, from PC's to mobile Internet devices, mobile phones, game consoles, etc. The growth in the number of users and in the variety of accessing ways have also provoked an evolution in the functionalities of the Internet for the users.

In the beginning, the Web was seen as a huge collection of information, organized in documents interrelated through hyperlinks. The main services in the Web were focused on the Information Retrieval task, developing tools that retrieves any kind of relevant information about some topic from the Web. The aim of these systems is to provide the users with a set of relevant documents (or web pages) related to an information need (*query*).

Last years, the paradigm of the Internet is moving on from the concept of just retrieving static information, to the ideas of interactive systems and collaborative contents. These are the basis of the Web 2.0. This new vision of the Internet has gained weight with the appearance of on-line social-based systems, so called *Social Networks*. Under this name, we can find web applications that cover a wide variety of themes and functionalities, and allow their users to share many kinds of contents and to establish different types of relationships between them.

Nowadays, both conceptions of the Internet coexist and they even share some common tasks and problems. Social-based systems are mostly made of user-generated content, so the information retrieval engines are still very

useful in order to search for relevant information in them. Furthermore, new problems have come up with these systems, such as the opinion leadership (Agarwal et al., 2008; Valente and Pumpuang, 2007), the reputation and community management (Yu and Singh, 2003), the detection of user communities in a network (Newman, 2003), the estimation of the trustworthiness of users (Golbeck, 2005; Guha et al., 2004; Kamvar et al., 2003b), etc.

In this dissertation we study the problem of dishonest behaviors in on-line networks, focusing our attention on those phenomena caused by users who try to obtain some kind of benefits by gaining a high relevance through the disturbance of the usual performance of the systems. We present two graph-based techniques intended to deal with dishonest behaviors in both the Web and social-based systems, respectively. In the Web, the dishonest behaviors are fundamentally the *web spam*, consisting in the creation of web pages with an illicit relevance in the Web for the purpose of obtaining web traffic. The principal aspect of the social-based systems is the *trust and reputation* of the users, and the main problem is caused by the dishonest behavior of users who try to gain a high reputation in the network.

1.1 Motivations

The huge growth of the Internet and the necessities of their users favored the development of new software to cover those needs, such as the web search engines, and a new conception of the Web as a social-based system. They also brought about the invention of new business models, and ways of taking advantage of them in order to obtain some kind of benefits from it. The motivations of our research are framed in this work environment.

As we stated above, web search engines have become one of the most useful (and widely used) tools in the Internet. They usually consist in three components: a web crawler, a document indexer and a document retriever. The web crawler collects the web pages in the Internet, following every links and extracting the useful information from the HTML contents. The indexer analyzes the data provided by the crawler, and stores the information of the web pages in one or more *indexes*, that is a database intended to facilitate the retrieval process making the information searching as fast as possible. Finally, given a user query, the document retriever searches in the indexes and builds a ranking with the most relevant documents that contains the information needs expressed by the query. In this way, the user have access to the most relevant resources about its information needs. The usefulness of a search engine depends on the relevance of the result set it gives back.

As an immediate consequence of this, the most relevant resources indexed

in a search engine are also the most visited ones. This fact motivated the appearance of a new business model consisting in offering advertising schemes that, in return for increased web traffic (number of visitors), pay for screen space on a web site. Web sites often aim to increase their web traffic through their inclusion on search engines (indexation). This business model acts as an incentive for web sites to generate contents that can be considered relevant by users and hence by web search engines.

This is the origin of one of the hardest problems of web search engines: the *Web Spam*. It is based on the manipulation of the *relevance* of a web page indexed by a search engine, for the purpose of obtaining some kind of benefits, usually an increased amount of web traffic. Web spam consists in the creation of web pages in order to make a web search engine to deliver undesirable results to a query (Najork, 2009). There are several works that tackle this dishonest behavior, we discuss them in depth in Chapter 4.

On the other hand, a similar problem is observed in the Web 2.0. Social networks introduce in on-line networks the novelty of allowing their users to generate contents in order to enrich the entire system and share those contents with other users. Usually, these systems also provide the users with *social functionalities*, for example establishing relationships with other users in the network, or voting the contents shared by other users. One point in common for the majority of the social networks is the necessity of qualifying their own contents in order to provide a better service and to improve the user experience. In social news sites and other content-sharing networks it is very useful to take advantage of the user opinions in order to give more relevance to some contents over others. In on-line marketplaces it is crucial to distinguish untrustworthy sellers or buyers, so these systems usually allows their users to evaluate their transactions.

The problem comes out when a user or a group of users take advantage of the voting system in order to gain any kind of benefits. For example, in on-line marketplaces a dishonest seller would want to gain high reputation in order to increase his sales. Or maybe a user in a social news site would try to give as much visibility as possible to news with a specific biased opinion about some topic. All these actions can provoke negative consequences in the services provided by these sites, disturbing the normal behavior of the social networks. *Trust and Reputation Systems* (from now on *TRS*'s) are intended to deal with this problem, avoiding the effects that users with dishonest behaviors can cause in a social network. We review the state-of-art on this field in Chapter 6.

1.2 Hypothesis

The framework where we develop our research work, the detection of dishonest behaviors in on-line networks, embrace two closely related topics: web spam detection and trust and reputation systems in social networks. Due to the great similarities and the points in common shared by both fields, we tackle them using very similar approaches, talking in an intuitive way. Both problems can be abstracted as a graph: we have a set of elements (web pages and users, respectively) which are related in some way (links between the web pages, or the different relationships in a social network). The tasks will consist, in both cases, in detecting what are the most relevant (or irrelevant, in the case of web spam) elements of the graph. This is the reason why we tackle both problems from the point of view of the graph theory, applying the logical modifications to adapt our proposals to each particular task.

The global hypothesis in our dissertation can be formulated as follows:

The detection of dishonest behaviors in on-line networks can be carried out with graph-based techniques, that are flexible enough to include in their schemes specific information (in the form of features of the elements in a graph) about the network to be processed and the concrete task to be solved.

Specifically, we formulate the next hypotheses for each problem:

- **Regarding to the web spam detection task**, the inclusion of knowledge about the textual content of the web pages in addition to the relations between the web pages (implicit in the web graph topology) into a graph-based model can improve the performance of a web spam detection system.

- **Regarding to trust and reputation task**, taking into account the negative links (edges with a negative weight) in a network in addition to the positive ones improves the discriminative ability of a system intended to detect dishonest behaviors in on-line networks.

1.3 Contributions

Following the hypotheses shown previously, the main contributions presented in this dissertation can be summarized as follows:

First, we have made a thorough analysis of the state of the art in graph-based learning related to the spam detection task, and the trust and rep-

utation systems. We have focused our attention in the different systems developed for both tasks, and the evaluation methods used in each field.

Second, we present *PolaritySpam*, a graph-based method for spam detection. It analyzes the links and contents of the web sites in order to determine their spam likelihood. The aim of our approach is to build a ranking of web pages according to their relevance, demoting the spam web pages in order to avoid them in the first positions of the ranking. The novelty in our proposal is the combination of both link and content analysis in a single method to compute the spam likelihood of the web pages.

Regarding the social-based on-line systems, in this work we propose *PolarityTrust*, a method that can take advantage of the information provided by both followers and detractors of a user in a social network, in order to obtain the trustworthiness of the users of the system. As far as we know, there are not many works on trust and reputation that process a network with negative opinions between their users, whereas there are some widely used social networks that present this feature. For example Digg [1] provides its users with the ability of *digg*-ing (vote positively) or *bury*-ing (vote negatively) the news, or Slashdot [2], where the users have a list of friends (positive relations) and foes (negative relations). Our proposal computes the trustworthiness of the users in a social network, regarding the positive and negative relations in the system. Additionally, in this work we also develop an extensible method for the generation of malicious users in on-line social networks and their behaviors, in terms of attacks against the system.

Finally, apart from the contributions about web spam and trust and reputation in social networks, we also include in this dissertation the results of some works developed on our path to this research which have contributed in different ways to the attainment of the goals of our research work. They consists in two graph-based algorithms intended to deal with two different NLP tasks: the POS-tagging and the computation of semantic orientations of the words. Some basic ideas later used and refined in the main contributions of this dissertation already appear in these first approaches.

[1] `http://digg.com`

[2] `http://slashdot.org`

1.4 Roadmap

The rest of this dissertation is organized into five parts, as follows:

- **Part I: Foundations.**
  - **Chapter 2: Background on Graph-based Ranking Algorithms and Graph Generation Techniques.** A study of the state of the art on graph-based techniques is presented in this chapter. We focus our attention on graph-based ranking algorithms, which are the basis for our contributions. We also review some relevant methods for the generation of random graphs, which have constituted an important resource for the evaluation of some of our proposals. Finally, we review some relevant applications of these techniques to different research fields.
  - **Chapter 3: Previous Research Works.** An overview of some of the main research works that we have developed previously to this work. The experience and knowledge acquired in the works reviewed in this chapter, *STR* and *PolarityRank*, have notably contributed in the attainment of the goals traced out.
- **Part II: Dishonest Behaviors on the WWW: Web Spam.**
  - **Chapter 4: Web Spam Detection.** In this chapter we study the state of the art on spam detection systems, classifying them into three categories regarding to the information used as input: link-based systems, content-based systems, and hybrid systems. We also discuss a real case study that illustrates some of the main difficulties of this task.
  - **Chapter 5: Graph-based Ranking Algorithms Applied to Spam Detection.** Here we present PolaritySpam, our proposal to deal with web spam detection. We discuss the details of our system, and provide an evaluation with a widely used dataset.
- **Part III: Dishonest Behaviors on Social Networks: Trust and Reputation.**
  - **Chapter 6: Trust and Reputation in Social Networks.** In this chapter we carry out an analysis of the computation of trust and reputation in social networks. We study separately the commercial systems that some real social networks are using in their web sites, and also some interesting research works on this

field. Furthermore, the main problems that must be faced when it comes to compute the trust and reputation of users in a social network are reviewed in this chapter, in addition to a real situation that helps us to show the difficulties in tackling this task.

  - **Chapter 7: A Trust and Reputation System Using Graph-based Ranking Algorithms.** We present here PolarityTrust, our graph-based TRS for social networks. It consists in a ranking algorithm which processes a directed, weighted graph representing a social network with positive and negative relationships among its users, and computes a ranking of the users according to their trustworthiness. Our proposal tackles some of the most common threats for a social network, demoting in the ranking those users who present a dishonest behavior.

- **Part IV: Conclusions.**

  - **Chapter 8: Final remarks.** In this chapter we present the conclusions extracted from our research work, analyzing the hypotheses stated in Chapter 1 and the results obtained by our proposals in the different tasks studied.
  - **Chapter 9: Future work.** Finally, we trace out some of the ideas for future works that this work has opened as our next challenges in the detection of dishonest behaviors in on-line networks.

- **Part V: Appendices.**

  - **Appendix A: Algorithms for the Random Generation of Graphs.** A compilation of the algorithms reviewed in this dissertation intended to generate random graphs.
  - **Appendix B: PolarityRank: Algebraic Proof and Convergence.** The algebraic justification and the convergence proof of PolarityRank, an algorithm that constitutes one of the basis of our research work.

Part I

Foundations

Chapter 2

Background on Graph-based Ranking Algorithms and Graph Generation Techniques

We are caught in an inescapable network of mutuality [...]
Whatever affects one directly, affects all indirectly
Martin Luther King

2.1 Introduction

This dissertation is focused on the application of graph-based ranking algorithms to the detection of dishonest behaviors in on-line networks. Therefore, the data to be processed consists mainly in a (usually huge) set of elements interconnected between them, forming a graph. The processes performed over these datasets consist basically in obtaining a score for the elements in the graphs by following the links between them and performing some kind of computation. The scores of the elements are taken into account in order to build a ranking representing the relevance (according to some criteria) of each element in the dataset.

On the other hand, due to the difficulties in getting an appropriate collection of datasets for the evaluation of this kind of systems, we also need certain background on random graph generation techniques. These methods are intended to provide us with suitable datasets by creating random graphs

which simulates the real-world networks that constitute the subject of our studies.

In this chapter we review the graph-based techniques which constitutes the background knowledge for our research work. Some of the most relevant graph-based ranking algorithms are discussed in Section 2.2. Then we explain a set of random graph generation techniques in Section 2.3. Finally, in Section 2.4 we comment some relevant applications of graph-based techniques in different fields framed in Computer Science area.

2.2 Graph-based Ranking Algorithms

Nowadays, graph-based ranking algorithms are widely applied to many different areas and problems. The flexibility of these techniques and their ability to model a wide variety of complex systems where the relations between their elements are the key point in order to obtain a ranking of the elements in the graph, make the ranking algorithms suitable for dealing with many tasks in different fields.

One of the problems that motivated the research on graph-based ranking algorithms in Computer Science is the web search task. Since the information retrieval is one of the prime objectives of the users on the Internet, web search engines are of crucial importance. As mentioned above, these tools obtains a set of documents which contain some information required by the user. The usefulness of the results provided by a search engine is evaluated in terms of the *aboutness* and the relevance of the retrieved documents. The aboutness of the documents measures to what extent the topic of a result matches the topic of the query or information need of the user. In the context of web search, the relevance of a web page not only includes the aboutness of the information within it, but also the importance of the web page in the Internet. This leads to the problem of building a ranking of documents (web pages) in order to obtain their relevance in the collection (the Internet).

Some of the early web search engines included the concept of *visibility* of the web pages as a feature to increase the importance of a web page in the collection, computed as the number of in-links of each web page. A more navigational model is proposed in (Marchiori, 1997), considering not only the number of in-links of a web page, but also the number of out-links and even the paths in the Web graph and the distance between the pages of each path. Another important milestone in this field is the appearance of PageRank (Page et al., 1999) and HITS (Kleinberg, 1999) in the same year, two iterative ranking algorithms intended to compute a ranking of web pages taking into consideration the link structure of the Web. The intuition

behind these algorithms is the *topical endorsement* assumption: if a page, w, focused on topic T, points to page v, we can consider that page v is relevant for topic T.

2.2.1 PageRank

PageRank (Page et al., 1999) is a graph-based ranking algorithm intended to measure the importance of each element of a hyperlinked collection of documents, such as a citation network or the WWW. The algorithm computes a numerical value, namely $PageRank$, for each element, representing the relevance of the element in the collection.

Formally, given a graph $G = (V, E)$ where V is a set of nodes and E a set of directed edges between two nodes, we define two operations, $In(V_i)$ and $Out(V_i)$, to obtain the set of in and out-links, respectively, of node V_i. From these two basic operations, we define the PageRank of a given vertex applying the following equation:

$$PR(V_i) = (1 - d) + \ d \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} \ PR(V_j),$$

where d is a damping factor that allows the convergence of the algorithm. In the context of navigation in the Internet, this factor represents the probability that a user accesses a page through a link at the current page, making $(1 - d)$ the probability of the user jumping to a random page not linked to the current page. In the original definition of PageRank a value of 0.85 for factor d is recommended.

Starting from arbitrary values for the scores of the nodes of a graph, the algorithm iterates until it converges. The convergence is reached when the largest difference between the scores of each node in two consecutive iterations is less than a certain threshold. Once the algorithm has finished, the score attained by each node represents its importance in the collection.

The previous formula is the most popular version of PageRank. According to it, all nodes are in equality of conditions and the ranking assigned to each one depends exclusively on the topology of the network. However, the original formulation of PageRank includes a parameter in order to insert a bias in the ranking computation, giving more weight to some nodes over the rest. The formula including this bias is as follows:

$$PR(V_i) = (1 - d)e_i + \ d \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} \ PR(V_j)$$

where the values e_i must satisfy that $\sum_{i=1}^{n} e_i = n$. If the distribution of the values e_i is uniform, the a priori relevance of each node is the same and this formula behaves the same way as the previous one. However, if there are nodes with higher e_i than the rest, these nodes become sources of relevance, propagating their influence through their neighbors with more strength.

A study of the complexity and the convergence of PageRank algorithm can be found in (Haveliwala, 1999). In this work they also propose a method intended to improve the resource required for the computation of PageRank, although nowadays there exist other approaches that achieves better performances in this sense. A more recent work on this field is presented in (Kamvar et al., 2003a), where they present an adaptation of PageRank intended to accelerate the convergence of the algorithm by avoiding the computation of the PageRank of those web pages that have already converged in a previous iteration.

The other main problem in the application of PageRank algorithm to the Web graph is the scalability, due to the huge size of the dataset that must be processed. Many works address this issue by proposing different parallelization methods intended to reduce the complexity in time and memory of the algorithm. In that regard, an efficient computation of PageRank is proposed in (Kohlschütter et al., 2006). They re-formulate the equation by establishing a new schema for the identification of the web pages. A web page p is now replaced by a tuple (p, l), where p is the identifier of the host and l is the identifier of the page inside host p. In this way, they compute the ranking corresponding to the inter-links between web pages of the same host, and then include the inter-host links in the computation. The first step is completely parallelizable at host level. An improvement of this work is presented in (Wicks and Greenwald, 2007), emphasizing the scalability of the parallelization of the algorithm.

A *divide and conquer* strategy is proposed in (Desikan et al., 2006) in order to improve the performance of PageRank. It is based on the intuition that PageRank of the web pages belonging to a set A, does not depend on the web pages from set B if there are no links from nodes in B to nodes in A, so they can be computed independently. In this way, the PageRank of non-connected partitions can be computed independently.

2.2.2 Hyperlink-Induced Topic Search (HITS)

HITS (Kleinberg, 1999) is another graph-based algorithm designed to value websites based on their links. Unlike PageRank, HITS calculates two values for each node: *Authority* and *Hub* scores. Both values are defined by mutual recursion and are calculated through an iterative process. The meaning of

both the *Authority* and *Hub* values for a given node is represented in Figure 2.1.

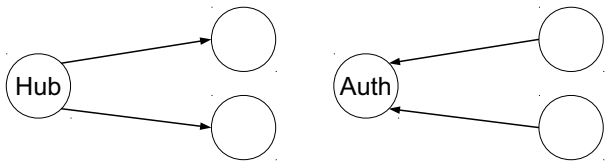


Figure 2.1 – Graphical representation of the intuition behind HITS algorithm: *hub score* takes into account the out-links of a node, while *authority score* considers the in-links.

Each node acts as a hub for the nodes where they point to, and as an authority for the nodes pointing to them. Intuitively, the *Authority* score assesses how relevant is the node itself and it is computed based on the *Hub* values of the links pointing to the node, while *Hub* score assesses how relevant is the node as a recommender, and it is calculated using the *authority* values of the nodes which it points to.

The definitions of both values, which support the iterative algorithm for calculating the relevance, are the following:

$$Authority(V_i) = \sum_{j \in In(V_i)} Hub(V_j)$$

$$Hub(V_i) = \sum_{j \in Out(V_i)} Authority(V_j)$$

After each iteration the values are normalized, ensuring that

$$\sum_{i=1}^{n} (Authority(i))^2 = 1, \ and \ \sum_{i=1}^{n} (Hub(i))^2 = 1$$

Similarly to PageRank algorithm, high values of *Authority* and *Hub* reflect a higher relevance of the web page. In this case, the concept of relevance of a node differs depending on the considered role, authority or hub. The intuitive justification of these definitions is the following: if a node points to many nodes with a high *Authority* it must have a high value of *Hub*; in the same way, if a node is pointed by many nodes with a high *Hub*, it should have a high value of *Authority*.

2.2.3 Stochastic Approach for Link-Structure Analysis

Stochastic Approach for Link-Structure Analysis (SALSA) algorithm (Lempel and Moran, 2001) combines concepts from both PageRank and HITS algorithms. It applies a random-walk algorithm to a collection of web pages, computing two scores for each one: hub and authority scores. SALSA algorithm builds a bipartite graph from the Web collection, placing a copy of the nodes with out-links (*hubs*) in one side of the graph, and a copy of the nodes with in-links (*authorities*) in the other side of the graph. Note that a node with both out-links and in-links will be copied twice placing each copy in the corresponding side of the bipartite graph. A graphical example is shown in Figure 2.2.

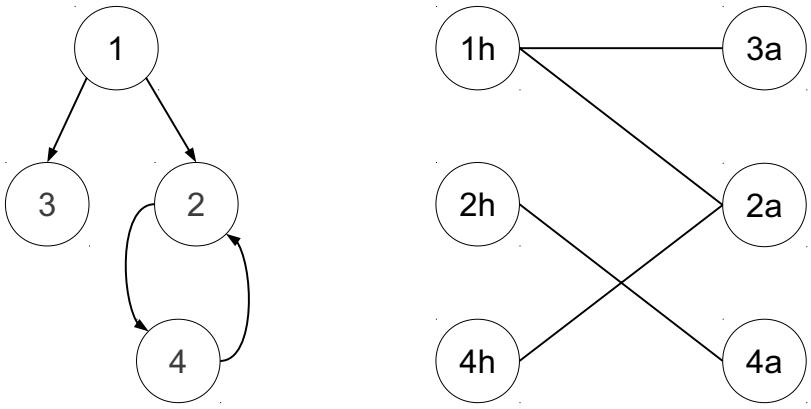


Figure 2.2 – Construction of the SALSA bipartite graph of hubs and authorities, given a Web graph. Nodes noted with a represent the role as authority, and the ones noted with h represent the role as a hub.

SALSA performs two random-walks, each of them are confined in one side of the bipartite graph. This is done by starting each random-walk from one of the parts of the graph, and then forcing them to cross two edges of the graph in each step. For example, given the bipartite graph in the Figure 2.2, focusing on the random walk starting from node **1h**, since the random surfer will cross two edges in each step, it will probably visit the node **4h** in the next step. In this way, the random-walk starting from the hub-side of the graph is expected to visit more frequently the most relevant hubs, and the same for the authority-side of the graph.

This technique successfully deals with highly connected sets of pages, namely *Tightly Knit Community* (TKC) Effect. This phenomenon harms the performance of Kleinberg's Mutual Reinforcement, because it can cause HITS to score unjustifiably high the pages in the link farms. *TKC* makes HITS vulnerable to some spam techniques (we will discuss them in depth in Chapter 4), while SALSA presents a good behavior against it.

2.2.4 TextRank

The motivation of TextRank is to apply a graph-based ranking algorithm to problems related to Natural Language Processing. Independently of its applications, the most interesting aspect of TextRank is that it generalizes the PageRank algorithm so that it can be applied to graphs whose edges have weights. In this case, the score of each node is computed as follows:

$$TR(V_i) = (1-d) + \ d \sum_{j \in In(V_i)} \frac{p_{ji}}{\sum_{k \in Out(V_j)} p_{jk}} \ TR(V_j)$$

where p_{ji} is the weight of the edge that goes from the vertex V_j to V_i.

The application of TextRank to the analysis of texts in natural language only requires the building of a graph from the input text, taking as nodes the linguistic units of the task being solved, and creating edges between the related elements. TextRank algorithm processes the text graph computing a score for each node, in order to build a ranking according to these scores.

This algorithm has been applied to different NLP tasks such as keyword extraction and summary generation (Mihalcea, 2004; Mihalcea and Tarau, 2004) or word sense disambiguation (Sinha and Mihalcea, 2007) with very good results. In each case, the construction of the graph varies depending on the task. For example, in the keyword extraction the nodes represent words and the edges are established between the nodes whose corresponding words appear close together in a sentence. On the other hand, for the summarization task each node represents a sentence of the input text, and the edges contain information about the co-occurrence of words in the sentences represented by their ending nodes.

2.3 Random Graph Generation

The goal of our research is the detection of dishonest behaviors in on-line networks. Most of the times, the experiments and evaluation in this and other areas related to network analysis are complex because of the difficulty of having access to real networks to test the systems on them. Research works

on this and other fields, such as Biology, Sociology and Genetics, sparked a growing interest on the research on random graph generation, in order to model the behavior of the real-world networks appearing in those fields.

The random graph generation techniques focus their attention in the creation of synthetic networks which present similar topologies and properties than the real-networks to be studied. These generation methods also provide useful information about how the real-world networks behave over the time, and the phenomena that occur in them.

There exist many works on the generation of random graphs for many different fields, such as Sociology, Physics, Biology and, of course, Computer Science (Chakrabarti and Faloutsos, 2006; Iribarren and Moro, 2011). Obviously, we are interested in the last group, so this section is focused on the study of some generation methods intended to obtain random graphs similar to those produced in on-line networtks, such as social-based systems and the WWW.

We review three methods intended to ensure the creation of graphs with some specific patterns observed in real-world on-line networks. It has been empirically observed in many studies that most of the real-world networks present some common properties:

- **Power-law Distribution**: the distribution of degrees (especially the number of in-links of a node) follows a scale-free power law. It means that the probability that a node has in-degree i is proportional to $\frac{1}{i^x}$, for some $x > 1$.

- **Preferential Attachment**(Barabási and Albert, 1999): when a new user is added to a network, it will choose with a higher probability the most linked nodes in the graph to be linked to. This phenomenon is often formulated as the *rich-gets-richer rule*. This property explains the power-law distribution previously mentioned.

- **Shrinking Diameters**(Leskovec et al., 2005): although the nodes in a social network tend to increase in number, the diameter of the graph grows slowly in function of the size of the graph, or it even suffers a slight decrease. This is caused by the *small-world* model of the social graphs. They are mostly formed by cliques, sets of highly interconnected nodes, so the maximum distance between two nodes stays almost constant. Moreover, this distance can decrease when a new user arrives to the network creating links to two different cliques.

The graph generation methods discussed in this section are intended to build graphs with similar characteristics than the real on-line networks, trying to model (some of) these properties.

2.3.1 Barabási-Albert model

This model is proposed in (Barabási and Albert, 1999). This work analyzes some real-world networks (WWW, citation networks, collaboration graph of movies actors), finding out a common property in their organization: the *Preferential Attachment*. This property explains the way in which real networks grow over time. It consists in the fact that the newcomers of a network attach preferentially to the most connected nodes of the system. The preferential attachment property leads to the generation of scale-free networks whose nodes have a probability, $P(k)$, of interacting with k other nodes that follows a power-law distribution of the form $P(k) \sim ck^{-\gamma}$.

The Barabási-Albert (BA) method is an algorithm intended to generate random scale-free networks with the preferential attachment property. The pseudo-code of the algorithm is shown in the Appendix A (Algorithm 2). A random graph with 40 nodes built using the BA model is shown in Figure 2.3(1.a). In Figure 2.3 (1.b), we plot the distribution of the degrees of the nodes in the graph randomly generated by this method. As it can be observed, it follows a power-law distribution of degrees.

2.3.2 Forest Fire model

ForestFire model is proposed in (Leskovec et al., 2005). This work focuses on the study of the growing patterns followed by several real networks, such as paper citation and patent citation networks and the Internet AS[1] graph. Two observations are pointed out by this study: first, most of the networks densifies over time, with the number of edges growing exponentially in the number of nodes as follows:

$$E(t) \propto N(t)^{\alpha}$$

where $E(t)$ and $N(t)$ are the number of edges and nodes, respectively, at time t, and $1 \leq \alpha \leq 2$ is called the *densification power-law exponent*. The second observation, namely *shrinking diameters*, states that the average distance between nodes often shrinks over time, in contrast to the conventional wisdom that such parameters should increase slowly as a function of the number of nodes. A recent research work carried out over Facebook (Backstrom

[1]The Internet can be viewed as a network of autonomous systems. An autonomous system (AS) is a subnetwork under separate administrative control and can consist of tens to thousands of routers and hosts. Examples of AS's are networks of big companies or universities, national research networks, local or national Internet service providers (ISPs), or international backbone providers.

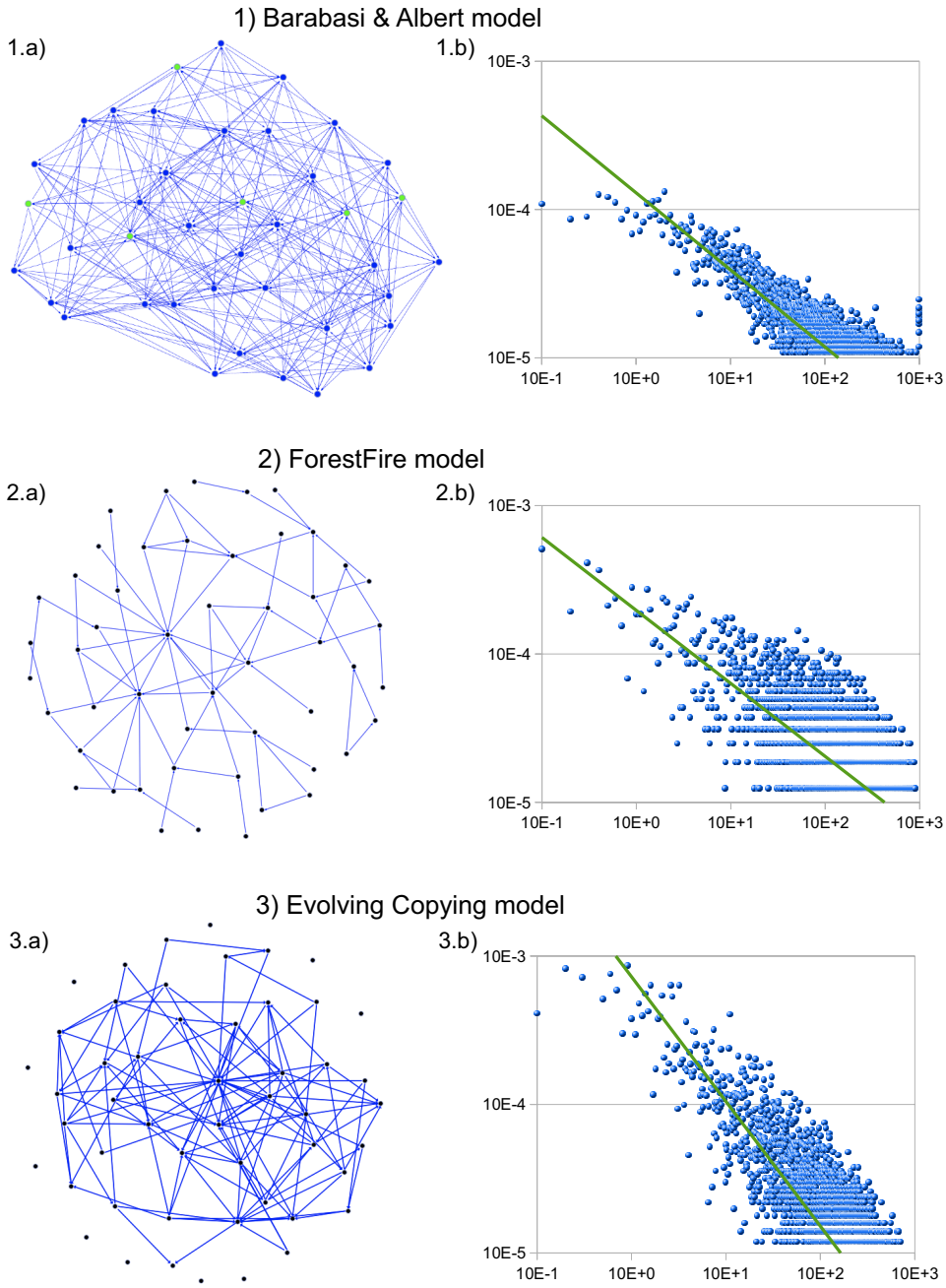


Figure 2.3 – Examples of random graphs generated by the models: Barabasi & Albert (1), ForestFire (2) and Evolving Copying (3). (a) Graphs generated by each model with 40 nodes. (b) Out-degree power-law distributions of random graphs generated by each model with 10^4 nodes.

et al., 2011) confirms these statements, observing a progressive decrease of the diameter of this social network.

ForestFire model follows these two observations, in addition to the preferential attachment property and the community guided attachment (we explain it more detailed in Section 2.3.3), in order to produce random graphs with similar growing patterns than the studied networks. The functioning of the model is shown in pseudo-code in the Appendix A (Algorithm 3).

The intuition behind the model is the *forest fire* spreading process. A new node is added to the graph in each step of the algorithm. The newcomer chooses an *ambassador* among the nodes in the graph, and creates a link to it. The "fire" starts at the ambassador, so the newcomer will copy a random number of edges from it, and then the process spreads probabilistically to the neighbors. Due to the use of an *edge copying* mechanism, this model presents similar characteristics than the ones in the next section (community guided attachment). An example of a random graph generated using the ForestFire model is shown in Figure 2.3 (2.a). Its corresponding degree distribution is shown in Figure 2.3 (2.b).

2.3.3 Evolving Copying models

These methods are proposed in (Kumar et al., 2000) with the aim of creating random graphs which model the *community guided attachment* property, in addition to power-law degree distributions. The community guided attachment is produced, in the case of the Web, by pages on the same topic which are prone to be highly interconnected forming communities. In this work, two methods are proposed: linear growth copying model and exponential growth copying model. In the first one, a new node is added in each time step t, with a constant degree $d \geq 1$. The out-links are chosen from the targets of the out-links of another node with a probability α, causing the copying effect. The pseudo-code is shown in the Appendix A (Algorithm 4).

An example of a small random graph obtained by this model is shown in Figure 2.3 (3.a), and a chart with the representation of the degree distribution of a large graph is shown in Figure 2.3 (3.b). A similar method is proposed in (Kleinberg et al., 1999), obtaining also random graphs with the community guided attachment property, in addition to a mechanism that models the preferential attachment property, giving more probability of being chosen to those nodes with highest in-degree.

Unlike the linear growth model, the exponential growth model adds a random number of nodes in each time step, following a binomial distribution. It establishes the probability, γ', of creating a new edge pointing to one of the nodes inserted in the last step. Otherwise, it is created an edge pointing

to a node inserted in previous steps of the algorithm. The out-degrees of the nodes are also taken into account when the endpoints of the edges are chosen.

2.4 Applications of graph-based ranking algorithms

In this section we show some applications of graph-based techniques on different Computer Science research areas. We review some of the main works related to graph-based applications for Information Retrieval (Section 2.4.1), Natural Language Processing (Section 2.4.2), Social Network Analysis (Section 2.4.3) and Recommender Systems (Section 2.4.4).

2.4.1 Information Retrieval

The inclusion of the hyperlinks between web pages in the computation of the relevance of documents in Web Information Retrieval systems has been already introduced in Section 2.2. This is one of the most important applications of graph-based ranking algorithms to IR. Early web search engines, such as Excite[2], WebCrawler[3] and Lycos[4] already provided higher rankings to those web pages with a higher amount of in-links, that are the most visible ones. Other proposals in this direction are presented in (Bray, 1996; Marchiori, 1997). These works also exposed the problem of the *Search Engines Persuasion*, mechanisms intended to make a web page as visible as possible by studying the ways to rank the web pages in the top positions of search engines. It could be considered as the ancestor of the current *web spam*.

Precisely one of the aims of PageRank (Page et al., 1999) and HITS (Kleinberg, 1999) was to minimize the impact of *search engines persuasion* techniques on the ranking of pages. Nevertheless, spam techniques have evolved in order to take advantage of these algorithms, arising a new challenge for IR systems. Apart from the computation of the relevance of documents, graph-based ranking algorithms have been also applied to other IR tasks.

A topic-biased Pagerank is presented in (Haveliwala, 2003). They use the personalization vector, $\vec{e}$, in the PageRank equation to include information about the relevance of each document for the topic of a given query, improving the ranking of documents in a search engine. In (Cho et al., 1998) a graph-based method for web crawling is presented, using PageRank as an indicator

[2] `http://www.excite.com/`

[3] `http://www.webcrawler.com/`

[4] `http://www.lycos.com/`

of the order in which a web crawler must visit the URL's in order to fetch more relevant pages first. A categorization method for web pages is presented in (Chakrabarti et al., 1998), it uses the hyperlinks between pages to enhance the performance of the system.

Measuring the similarity between documents is also a key task for IR systems. SimRank (Jeh and Widom, 2002) is a graph-based method that obtains a similarity measure based on the assumption that two documents are similar if they are linked to other similar documents. They validate their proposal by applying SimRank to two co-citation datasets. A similar approach is presented in (Ding, 2011), where they introduce a topic-biased PageRank in the computation of a ranking of authors, given a dataset of scientific papers. Entity search is a IR topic consisting in providing a ranking of entities (instead of documents) related to a given query. This task is tackled in (Pehcevski et al., 2008; Zaragoza et al., 2007) using graph-based methods applied to the link structure of the Wikipedia [5]. An adaptation of TextRank (Mihalcea and Tarau, 2004) for term weighting is proposed in (Blanco and Lioma, 2007). They use the algorithm to compute term weights in a graph that represents the co-occurrence of the terms.

Finally, in (Vallejo et al., 2010) it is introduce a graph-based algorithm to solve a retrieval task related to the data mining field, ISR (*Instance Selection based on Ranking*). ISR is an instance selection technique that uses InstanceRank, a ranking algorithm that reflects the relevance of the instances within a dataset. This algorithm chooses the most representative instances from a learning database, obtaining similar results in accuracy to other instance reduction techniques, noticeably reducing the size of the instance set.

2.4.2 Natural Language Processing

In recent years, graph theoretic models have been increasingly used in Natural Language Processing (NLP) in many different ways, and dealing with several related tasks. Due to the nature of the human language, graph theory provides a very intuitive representation of the relations among the linguistic units on the different levels of a language (lexical, syntactic and semantic). In this section we make a summary of some representative examples of the application of graph-based techniques to computational linguistics.

Even though it is not a graph-based ranking algorithm, we think that it is worth to include in this section a brief discussion about one of the most relevant NLP resources in the past few years: WordNet (Fellbaum, 1998). It is a graph-based resource consisting in a lexical database of English that

[5] `http://www.wikipedia.org/`

includes definitions of concepts in addition to lexical and semantic relations between those concepts. The basic element in WordNet is the *synset*, a group of words that express (approximately) the same meaning or define the same concept, that is a set of synonyms. Each synset is linked to others by different kind of linguistic relations, such as antonymy (two words expressing the opposite concept), hypernymy (*is-a* relation between synsets), meronymy (*part-of* relation between synsets), entailment (relation between two verbs, X and Y, when doing X implies that you are also doing Y), etc. In Figure 2.4 it is shown an example of the relations between synsets, taking the word *graph*.

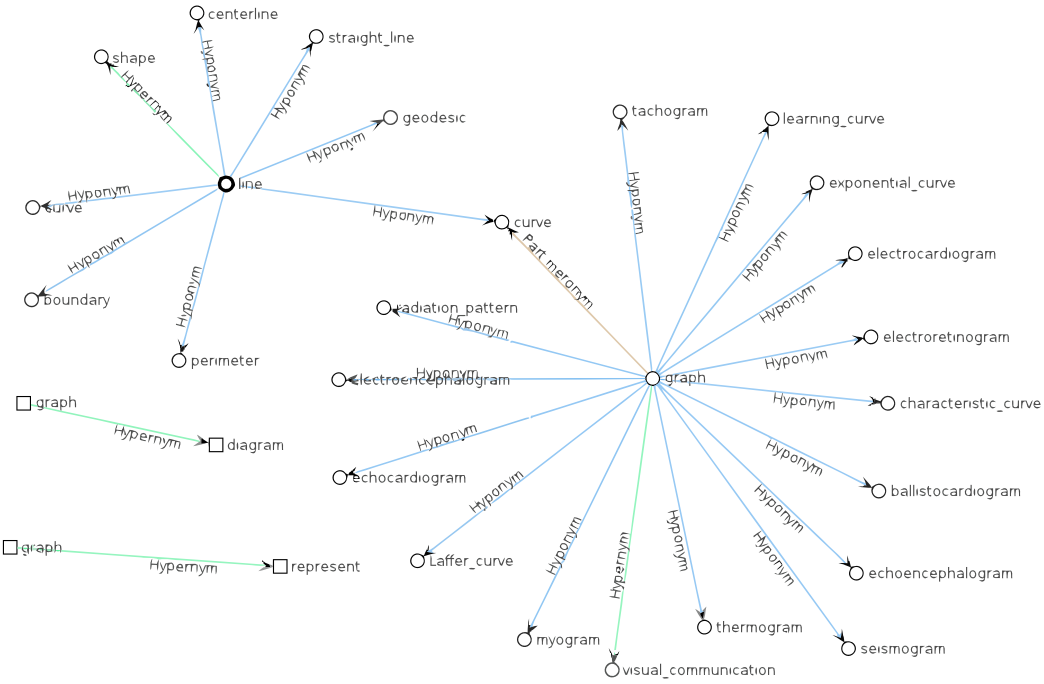


Figure 2.4 – An example of the relations in WordNet between synsets for the word *graph*

Due to the success of WordNet, other semantic networks have been developed for many other languages based on the original project, such as MultiWordNet (Pianta et al., 2002) for the Italian, or EuroWordNet (Vossen, 1998) which consists in a set of WordNet's that includes, among others, Spanish, Dutch, German and French.

Currently, the last version is WordNet 3.0, and it contains more than 150,000 words grouped in more than 115,000 synsets. WordNet is a resource widely used in many NLP applications, such as word sense disambiguation (*WSD*) (Agirre and Soroa, 2009; Li et al., 1995; Seo et al., 2004), textual

entailment (Micol et al., 2007) and correference resolution (Ponzetto and Strube, 2006).

Another important milestone in the use of graphs to deal with NLP tasks is Mihalcea and Tarau's TextRank (Mihalcea and Tarau, 2004) already discussed in Section 2.2.4. TextRank has constituted the basis for multiple graph-based applications in different NLP tasks, and also the seed of an increasing interest on graph theoretical approaches to computational linguistics, materialized in the *TextGraphs Workshops*[6], held yearly since 2006 in the context of the *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (ACL/HLT). This is one of the most interesting discussion forums on grah-based techniques for NLP.

There are several works that apply TextRank or similar ideas to other NLP problems. For example, TextRank scores are used in (Gamon, 2006) as one of the features included in a system intended to detect the novelties introduced by a text, given a previously acquired background information.

The same author of TextRank applied it to two main tasks: word sense dissambiguation (Sinha and Mihalcea, 2007) and extractive text summarization (Mihalcea, 2004). The first tasks consists in finding the correct sense of each word in a text. It is tackled by building a graph where the nodes represent each possible sense of the words in a sentence, and weighted edges are drawn using a semantic similarity measure (obtained from WordNet), computed for all pairs of senses for words found within a certain distance. The graph-based ranking algorithm obtains a score for each sense of the words, so the highest ranked sense for each word are chosen as the correct ones.

Another work related to this task (Aguirre et al., 2009) calculates the semantic relatedness of two given words by computing the personalized PageRank (Haveliwala, 2003) over WordNet separately for each word, producing a probability distribution over WordNet synsets. Then they compare how similar these two discrete probability distributions are by encoding them as vectors and computing the cosine between those vectors. The same authors propose in (Agirre and Soroa, 2009) a WSD method using the personalized PageRank, as well. In this case, they use the graph formed by the synsets in WordNet. The inference of the correct sense of a given term in a context is done by adding all the content words in the context of the term into the graph of WordNet, in such way that the words are linked to their corresponding synsets with a directed edge. The initial probability mass is concentrated over the recently added nodes. In this way, the personalized PageRank will give more weight to those synsets related to the words.

[6] `http://www.textgraphs.org/`

On the other hand, the *extractive text summarization* consists in building a summary of a given document by choosing the most relevant sentences of the text. This contrasts with *abstractive summarization*, where the information in the text is rephrased. In (Mihalcea, 2004) a graph is built by creating a node for each sentence in a document, and the edges are weighted according to the co-occurrence of words between each pair of sentences. The graph-based ranking algorithm computes a rank for each sentence, so the final summary is built by choosing the highest ranked sentences.

Another graph-based ranking algorithm for the extractive text summarization task is introduced in (Erkan and Radev, 2004). This algorithm, namely LexRank, also computes the sentence importance based on a graph representation of the sentences in a text. In this model, a connectivity matrix based on the cosine similarity between the sentences is used as the adjacency matrix of the graph.

In (Hassan and Banea, 2007) it is described a new approach for estimating term weights in a text classification task. The approach uses term co-occurrence as a measure of dependency between word features. A random walk model is applied on a graph encoding words and co-occurrence dependencies, resulting in scores that represent a quantification of how a particular word feature contributes to a given context. They show in the experiments certain improvement achieved by the random-walk based approach, which outperforms the traditional term frequency approach to feature weighting.

In reference to this work, our first contact with graph-based ranking algorithms consisted in the study of possible applications of graph-based, supervised machine learning schemas to NLP classification tasks. This interest was the origin of the work presented in (Ortega et al., 2011b), based on a supervised version of TextRank (STR) intended to deal with classification tasks, such as Part-Of-Speech (POS) Tagging. We review in depth this work in Section 3.2.

Following this line, we have also participated in the development of PolarityRank (Cruz et al., 2011b), another work on Natural Language Processing using graph-based algorithms. It computes the semantic orientation of the words in a text by processing a weighted graph. The semantic orientation of a word is a measure of the emotional implications, positive or negative, of that word when being used in a subjective context. It is usually represented by a real number whose sign indicates the polarity of the opinion involved, and whose magnitude indicates its intensity. PolarityRank processes a graph whose edges have positive and negative weights, representing the relations between the opinionated words in a text. We discuss deeply this contribution in Section 3.3.

2.4.3 Social Network Analysis

A social network can be defined as a set of individuals linked between them through different kinds of relationships. The analysis of these networks has been conducted from the 50's in diverse research areas such as Anthropology, Psychology, Biology and Economics. Nowadays, the appearance of on-line networks has caused an increasing interest in this field, due to their popularization and massive use with different purposes like social news sites, blogs, micro-blogs, networks for question-answering or product reviews, etc.

In Computer Science, the use of graph-based techniques for the social network analysis has proved to be very effective. These works cover many aspects of social network analysis, like the following:

- **Detection of communities**: clustering the users of a social network according to their relationships or any other aspect of their behavior (Ishida, 2005; Kumar et al., 2003). An example of the use of graph-based ranking algorithms to deal with this task can be found in (Tseng, 2005), where a blog community detection method is presented. The proposal in this work relies on the observation that blogs linked by a highly-ranked blog appear more connected than blogs linked by a lower-ranked blog. Thus, they develop a ranking-based connectivity analysis of the blogs. The proposal first identifies the relevant entries of the blogs with respect to a user query if the query term exists in the entry. Then the impact scores of these relevant entries are computed with a graph-based ranking algorithm. Finally, the ranking scores for the blogs are derived from the entry scores.

- **Influence of users**: studying the users whose opinions are more influent or get more propagation over a social network. The objective of this task is to determine the *opinion leaders* within a social network. In (Agarwal et al., 2008) they propose an intuitive model for evaluating the influence of blog posts and, hence, the influence of the author of the blog. It is based on four parameters: Recognition (proportional to the in-links), Activity Generation (proportional to the number of comments), Novelty (inversely proportional to the out-links) and Eloquence (inversely proportional to the posts length). The influence score is iteratively computed, similar to the PageRank definition, but applying a positive reinforcement from in-links and negative reinforcement from out-links to each blog post. Other graph-based work on this task is introduced in (Akritidis et al., 2009), where two metrics are developed in order to obtain the influence of bloggers taking into consideration the time as a relevant factor in the computation. Both metrics include

the publication dates of the blog posts and their comments, in addition to the number and dates of the in-links received.

- **Trust and reputation**: evaluating the trustworthiness of users in a network, regarding their actions and the opinions from the rest of users (Jø sang et al., 2007). This is a key point for on-line marketplaces (Chau and Faloutsos, 2005; Dong et al., 2009), where a buyer will choose the most trustworthy seller, and vice versa. Graph-based ranking algorithms are commonly applied to this task in order to obtain a score for each user according to their trustworthiness in the network, such as in (Guha et al., 2004; Kamvar et al., 2003b; Xiong and Liu, 2004; Zhou and Hwang, 2007), etc. One of the main aims of these systems is to detect suspicious actions of users who can try to disturb the normal behavior of the social network. These works will be more deeply reviewed in Chapter 6, together with an analysis of the main vulnerabilities and difficulties that these systems must overcome.

- **On-line Reputation Management**: it consists in monitoring the opinions from users in on-line networks about a specific entity (Amigó et al., 2010) (organizations, businesses, companies, political parties, etc). Reputation management systems collect this information and also facilitate the interactions between the entity and its "customers". For example, in (Jin et al., 2009), they present an approach to rank entities from a social network that has been mined from the web. They extract different kinds of relational data from the web in order to build a social network using several relevance measures in addition to text analysis (based on the co-occurrence of words). They integrate several types of relations into a network and use graph-based ranking algorithms to obtain the ranking model. This ranking is used in addition to some centrality scores of the companies on the network as features for ranking.

- **Hot-trends discovering**: many social networks are based on the publication and discussion of news, this task is aimed to detect the most relevant trends in a specific period of time in accordance with the interests of the users (Glance et al., 2004; Macdonald et al., 2009; Platakis et al., 2008). In (Qamra et al., 2006) it is proposed a graph-based approach intended to discover the trending topics on the blogosphere. It is based on the construction of a Content-Community-Time model that can leverage the content of the entries of blogs, their timestamps, and the community structure of the blogs (including the relations established when a blogger comments other blog, the links among blogs,

etc.), to automatically discover the hot-trends in a specific period of time.

Furthermore, the study of the main properties and characteristics of online social networks have made possible the implementation of a number of methods for the generation of random networks that are intended to model their topology of nodes and edges. We have reviewed some of them in Section 2.3. These techniques are very useful, not only for the generation of synthetic datasets, but also for the study of the normal actions and interactions of the users in a social network, helping to detect suspicious behaviors or even potential risks for the systems.

2.4.4 Recommender Systems

Recommender systems constitute an emerging research area which has gained popularity due to its straightaway applicability to e-Commerce, and also its inter-disciplinary character, influenced by techniques from many other fields such as information retrieval, machine learning and data mining. The basic aim of a recommender system is to offer a number of items (diverse products, web pages, images, videos, etc) that are likely to be of interest to a specific user. The objectives in the implantation of a recommender system into a web site are the increase in the number of sales, building a stronger customer bond, and also to make a difference with respect to the competitors. Recommender systems are usually classified as follows (Adomavicius and Tuzhilin, 2005):

- **Content-based recommendations (Lops et al., 2011)**: these systems are based on the experience of each user in order to recommend him items related to the ones he preferred previously. The interests of the users are usually encoded by building *user profiles* regarding the products that each user has visited, voted or bought. The pros of these systems are: the independence between users (we only need information about the user of interest), the possibility of easily justify the recommendations to the user, and finally these systems are not affected by the addition of new items (cold-start). In contrast, the cold-start for users is a problem for content-based systems, due to the lack of information about the newcomers. The other drawback of these systems is the high dependence on the information included in the user profiles, that entails the risk of offering too obvious recommendations or even not been able to provide any recommendations at all, in the case that the information is not useful to discriminate the items that the user will prefer.

- **Collaborative filtering (Koren and Bell, 2011)**: a user is recommended items preferred by other users with similar tastes. There exists two basic approaches to determine the similarity between users: memory-based techniques, where the similarity is computed as the correlation between the ratings provided by the users, or model-based techniques that use more complex patterns and machine learning methods such as graph-based techniques, bayesian networks and latent semantic analysis.

- **Hybrid systems (Adomavicius and Tuzhilin, 2005)**: these approaches combine ideas from the previous ones in order to improve the performance of the recommender system. In this group are included the content-based techniques that use features from collaborative filtering (and vice versa) in order to avoid their weaknesses, and also other existing methods for the combination of systems (Burke, 2007).

Graph-based techniques have been applied mostly to collaborative filtering approaches like in (Aggarwal et al., 1999). Some works are focused on the improvement of these kind of recommenders by including diverse information into the system. In (Clements et al., 2009) they combine in a single content ranking the results from two graphs, one based on positive and the other based on negative preference information. The resulting ranking contains a lower amount of false positives than a ranking only based on positive information. So negative information appears to have a valuable predictive ability for relevant content, in such way that discounting the negative information removes the irrelevant content from the top of the ranking.

Trust-based recommender systems combine the item ratings provided by the users and also the information about the trusted neighbors of each user (trust network of a user). In (Golbeck, 2005) a trust graph is processed, constructed by the relations between the users of the system. This work is based on the assumption that the users in a social network develop social connections with people who have similar preferences. The algorithm computes the trust score for each pair of users, $t_{u,v}$, relying on the direct experience of all the nodes that constitute the shortest path from u to v in the network of trust. They test their approach on FilmTrust (Golbeck and Hendler, 2006), a website that integrates Semantic Web-based social networks, augmented with trust, to create predictive movie recommendations.

Another trust-based recommender system, TrustWalker, is proposed in (Jamali and Ester, 2009), combining the trust-based and the collaborative filtering approach for recommendation. It consists in a graph-based ranking algorithm that allows to define and also to measure the confidence of each

recommendation. In the construction of the graph, they consider not only the ratings of the target item, but also those of similar items. This feature is intended to overcome one of the difficulties of this type of systems: the cold-start problem for the items (the lack of user feedback about new items in the system). So it improves the coverage in terms of ratings of items. Furthermore, this model improves the precision of recommendations by preferring raters at a nearer distance.

On the other hand, other works are focused on the *serendipity problem* of the recommender systems. This problem consists in an "overfitting" of the system to the tastes of the users, providing the users with recommendations that do not add any novelty or are not useful because they are too close to the user tastes. Many times it is also desirable for a recommender system that it offers diverse products that cover many aspects of the users preferences, similarly to diversification tasks in IR (Santos et al., 2011). In this case the recommendation problem could be re-formulated as: offer products that are interesting for the users, and also different from each other. This problem can be also tackled with graph-based approaches, like in (Zhu et al., 2007) where an absorbing random-walk algorithm is proposed in order to obtain a set of diverse items in the system by eliminating the similar occurrences in the ranking.

2.5 Summary

The objective of this work is to apply graph-based ranking algorithms to the detection of dishonest behaviors in on-line networks. In this chapter we have provided the necessary background on the graph-based techniques that constitute the basis for our research work, to wit: the random generation of graphs and the graph-based ranking algorithms. We think that it is worth to review these techniques in order to clearly understand the basic tools and nomenclature used in this dissertation.

With this aim, first we have reviewed some relevant ranking algorithms on graphs, intended to obtain ordered lists of the nodes in a given graph according to their relevance in the network. Then, we have explained three methods developed in order to randomly generate graphs. These generation methods are intended to create graphs that emulates the topology and properties of real world on-line networks. In our work, we will use them to generate the datasets for the evaluation of our proposals by creating random networks which emulate the behavior of the users in real on-line systems. Finally, we have reviewed some applications of these graph-based techniques to different research areas, such as Information Retrieval, Natural Language Processing,

Recommender Systems and Social Networks, giving an overview of the wide variety of fields where these methods have been successfully applied.

Chapter 3

Previous Research Works

Caminante, no hay camino
se hace camino al andar.
(Traveler, there's no road,
the road is your traveling.)
Antonio Machado
Cantares, Campos de Castilla

3.1 Introduction

The path followed in our research work, that has led us to this work, and the works carried out during that period of time, previous to the development of this dissertation, are interesting in terms of understanding the underlying ideas that constitute the foundations of our work on the detection of dishonest behaviors in on-line networks with graph-based methods. In this sense, we think that it is worth to highlight the milestones of this path, due to their relevance and also their influence over the next steps in our research work.

Therefore, in this chapter we discuss the two main contributions that were the seeds of our research work. The first one, reviewed in Section 3.2, consists in a graph-based tagger for Natural Language Processing (Ortega et al., 2011b). The main novelty in this work was the inclusion of a graph-based ranking algorithm into a supervised machine learning schema for classification tasks. We proposed a method intended to build graphs from texts in such way that the information from the text (co-occurrences of words, frequencies of the terms, transition and emission probabilities, etc.) could be included into the graph, characterizing the nodes and the edges of the graph in order to obtain a richer model. Thus, this new enriched graph model can

take advantage of the information extracted from the textual content, in addition to the information of the relations between the different elements in the text.

The second milestone in our way is the developing of PolarityRank (Cruz et al., 2011a,b), a graph-based ranking method which processes a graph with positive and negative edges. This algorithm has been successfully applied to the computation of the semantic orientation of the expressions in opinionated texts. PolarityRank consists in a propagation algorithm that takes advantage of the information about some relevant nodes (namely *seeds*) in the graph in order to spread it over the network following the positive and negative links between the nodes. The objective is to compute a score for all the nodes in such way that they are influenced not only by their neighbors (nodes connected) but also by the set of *seeds*, because their information is propagated over the network with more strength. This algorithm was applied to the automatic expansion of opinion lexicons. The main aim was to compute the semantic orientation (positive or negative) of the expressions in a text, given the semantic orientations of a set of known opinionated expressions. We discuss in detail this work in Section 3.3.

3.2 STR: A Graph-based Tagger Generator

Graphs have been proved to be one of the most natural representation methods in many NLP applications (Biemann et al., 2007), as we mentioned in Chapter 2. In fact, since a text can be seen as a group of words with some kind of relationship among them, we have a graph representation of the text. In the past few years, some proposals highlighted the usefulness of the graphs in machine learning, and many workshops and conferences were coming up with their attention focused on the application of graph-based algorithms in NLP tasks.

An interesting proposal in this area is TextRank (Mihalcea and Tarau, 2004), already discussed in depth in Section 2.2.4. TextRank is an unsupervised graph-based ranking algorithm conceived to deal with some NLP problems. It is based on the PageRank algorithm but taking as input a text instead of web pages. TextRank obtains a ranking of the nodes of a text graph, that is a graph built from a text. One of the main contributions of this work was the adaptation of the original PageRank in order to deal with weighted graphs, so the relationships among vertices can be evaluated taking into account their strength.

One of the limitations of TextRank is that it cannot offer a general solution to text processing. Applying it to a task makes it necessary to find a suitable definition of the task in terms of graphs. Trying to overcome this limitation, we proposed STR (*Supervised TextRank*) (Ortega et al., 2011b), a method intended to help in the definition of graphs from a tagged corpus to deal with different NLP tasks. We implemented these ideas in a tagging tool with two main capabilities:

- Applying the TextRank algorithm in a supervised learning schema, using a tagged corpus to build the model.
- Allowing to easily define different graph structures that could be adapted to the features of the input corpus and/or the task that we are dealing to.

In previous works (Cruz et al., 2006a,b) we also showed the feasibility of using TextRank in a supervised machine learning schema, covering the first goal of our work. With the methodology proposed, it is possible to build text graphs using the information extracted from a training corpus. This a priori knowledge helps TextRank to improve its discriminative ability, and makes it flexible enough to deal with different NLP tasks, provided a corpus for each specific task. This covers the first point.

The second capability has been achieved with the definition of a graph description language. It allows to describe the graph structure (nodes and edges) of the model to be processed, and the way that the information extracted from the corpus is used for characterizing the graph. Using these features, it is possible to train taggers with a text corpus and experiment with different graph topologies in order to find the graph structure that fits the best with the information in the corpus and the goal of the application.

The organization of the rest of the section is as follows. First we introduce the foundations of our approach in Section 3.2.1. Then in Section 3.2.2 we specify the settings of the experiments performed to test our ideas. The results obtained by our proposal and a brief comparison to other techniques is shown in Section 3.2.3. Finally, in Section 3.2.4 we point out some useful conclusions extracted from this work.

3.2.1 Supervised TextRank

In order to successfully apply TextRank to an NLP task, the resolution of the problem must be close to a graph representation. At this point the question is: is it possible to apply the TextRank algorithm to other NLP tasks?

At the moment when this work was finished, we had not found TextRank-based applications to other classical NLP problems like POS-tagging, syntactic analysis, or information extraction. The primary objective of our work was, precisely, to explore other possible areas of application of the TextRank algorithm and, at the same time, trying to use it in a supervised schema, so the information available in a training corpus could be used by the algorithm. In this sense, we performed a set of preliminary experiments to assess the method feasibility. The results of these experiments were exposed in (Cruz et al., 2006a,b). Both works showed the way to apply the TextRank algorithm to any task that could be specified through a tagged corpus.

The method is as follows: once we obtain a corpus collected for solving a specific task, our proposal extracts the information from that corpus and builds a text graph, using the corpus-based information to characterize the weights of the edges in the graph, in such way that the edges represent the relations between the elements in the text and their weights correspond to the intensity of these relations.

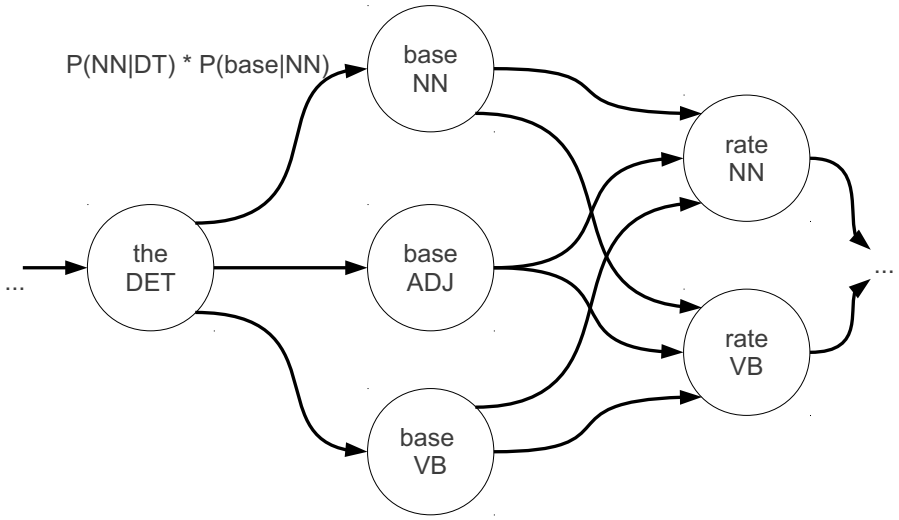


Figure 3.1 – Graph portion from the text *"...the base rate..."*.

Foundations

There may be more than one method to represent an NLP task as a graph model. We chose one generic way that allows us to apply it to any problem in which a set of tags is related to a set of words. All the vertices in our graphs have, at least, two attributes, $V = (w, t)$, where w is a word and t is its associated tag. If there is an ambiguous word (a word with more than one related tag), one vertex will be created for each tag of the word. The idea behind our approximation is to build a graph with this type of vertices for each sentence, then we apply TextRank to the graph and, finally, a tag is assigned to each word in the sentence according to the node of that word with the highest score. If there are many occurrences of a word in a sentence, we will create one vertex for each occurrence to avoid interferences between them. Figure 3.1 shows a portion of the graph built from the sequence of three words belonging to a sentence. In order not to overload the graphic, only the probability related to one of the edges has been shown.

The edges are created depending on the co-occurrence of the words in each sentence. For example, let $V_i = (w_i, t_i)$ and $V_j = (w_j, t_j)$ be two nodes in the graph. An edge connecting them is created only if the word w_j appears in the sentence just before the word w_i. Finally, the information extracted from the training corpus is used to decide the weights of the edges in the graph. After trying some formulas, we achieved the best results with a combination of HMM (Hidden Markov Model) emission and transition bi-gram probabilities, $P(w|t)$ and $P(t|t-1)$, respectively. These probabilities are estimated by calculating the frequencies of tags and words in the training corpus:

$$P(w|t) = \frac{C(w,t)}{C(t)} \qquad P(t|t') = \frac{C(t',t)}{C(t')}$$

where $C(t)$ is the count of tag t in the training corpus, $C(w,t)$ is the number of occurrences of the word w tagged with t, and $C(t',t)$ is the number of times that a tag t appears just before tag t'. According to Markov Model, the probabilities are used to obtain the best assignment of tags to words in a sentence, maximizing the following equation:

$$P(t_{1,n}|w_{1,n}) = \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

where $P(w_i|t_i)P(t_i|t_{i-1})$ is the weight for the edge created between vertices $V_{i-1} = (w_{i-1}, t_{i-1})$ and $V_i = (w_i, t_i)$. Instead of that, the TextRank algorithm calculates the importance of each vertex. Intuitively, an edge between two nodes represents the relation between two elements in the text,

and the weight is the strength of this relation. In the case of POS tagging, an edge from a word to the next one encodes the influence of the first word in the classification of the second one.

Once the graph is built, its nodes are sorted according to their importance computed by the TextRank algorithm. In order to obtain the final tagged text, we just have to take one node for each word in the text, starting from higher to lower ranked.

Given that a graph is built for each sentence in the text, STR is scalable with regards to the size of the input text, because the size of the graph only depends on the size of the sentences in terms of the number of words contained in each sentence. Furthermore, since the classification of each occurrence of a word in a sentence is independent of the tag of other occurrences of the same words in other sentences, the process can be easily parallelized by simultaneously running the computation of the algorithm corresponding to different sentences of the input text.

Experimental framework

We developed an experimental framework that allowed us to easily perform the experiments with STR. It consisted in three components: *trainer*, *tagger* and *graph-builder*. All the components use a graph specification file in order to process an input text with the graph model described in it.

The trainer component takes as input a labeled corpus and retrieves the information needed by the specified graph model. The graph-builder component builds a text graph corresponding to an input text. It takes as input the text and the graph specification and creates a graph that can be processed by the next component. The tagger obtains the tagging of an input text by processing the related text graph using the STR algorithm.

We identified two main parametrization points in our tool: smoothing algorithms for probability distributions, and unknown-words treatments. There are many methods to deal with these problems, so we developed several components to solve each of them. We can choose one of them by setting up the appropriate parameters of our tool. Finally, other parameters that can be adjusted in STR are those related to the TextRank algorithm: the "random surfer model" damping factor, d, and the threshold that will stop the iterative process. We will be able to change each step of our supervised classifier, and specify all the characteristics of our experiments by modifying these parameters.

Graph description language

In this section we introduce the graph description language that was intended to specify the topology of the graph models that will be processed by our tool. A description of a graph consists in a text file with three sections: in the first one the input corpus features are defined, node structure is specified in the second section, and finally, the third part contains the definition of the expressions to calculate the weights of the edges. For example, the textual description of the graph in Figure 3.2 is shown in Figure 3.3.

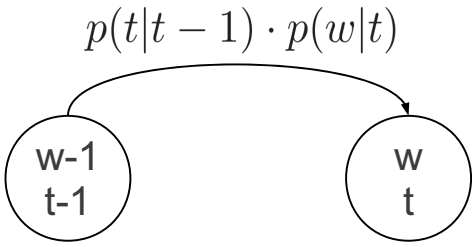


Figure 3.2 – Graph representing a bi-gram model.

We can define as many kinds of nodes as needed. A type of nodes is described with an identifier, separated by a colon from a list of slots. A slot is a relative position of one of the corpus features, and it is defined by the name of the feature and the offset (distance) from the current word. For example, in the specification shown in Figure 3.3 node `n` is described by a combination of the slots `w` (current word), `t` (tag assigned to the current word) and `t-1` (tag of the previous word). The graph edges are defined by indicating the source and target node types, and the expression used to obtain the weight of the edge.

```
// Bigram model. Two-column corpus
Corpus
    [w,t]
Nodes
    n*: w, t, t-1;
Edges
    n-1 -> n : p(t|t-1)*p(w|t);
```

Figure 3.3 – Textual description of a graph (graph of Figure 3.2).

3.2.2 Experimental settings

In this section we present the experiments made to prove the accuracy of Supervised TextRank method. We decided to prove the performance of STR in the POS Tagging task, a well-known classification problem that has been widely studied in the bibliography. There are also a number of techniques that deal with this task with a really high accuracy. If STR achieves the level of these state-of-art techniques, we can conclude that it is a useful classification method.

The first set of experiments with STR are based on classical bigrams and trigrams models. We also designed experiments using stacking. This technique has demonstrated to be a good mechanism to combine different methods. Stacking is used in order to build a classifier that takes into account the decisions of various graph models.

Stacking

Stacking has been often applied in machine learning and NLP(Florian et al., 2003; Halteren et al., 2001). This technique consists on applying machine learning methods to combine the results of different models. The main idea is to build a system that learns the way in which each model is right or makes a mistake. Therefore the final decision is made according to a pattern of correct and wrong answers. In order to be able to learn the way in which every model is right or wrong, we use a training database. Each example in the training database includes the tags proposed by the constituent models for a given word together with the right tag (Fig. 3.4)(Troyano et al., 2007). From this point of view, deciding the tag given the tags proposed by several models is a typical classification problem.

Apart from the outputs of each STR model, we can include more information in a stacking model, such as the context of each instance, or any other feature. For example, in our framework it is useful to combine various STR graph models in a single training database, and include not only the tags proposed by each model, but also the ranking of the instances.

In accordance with our experiments, the use of this technique to combine different graphs models in a single classifier achieves better results than the definition of a more complex graph.

Parameters

In the experiments exposed here we wanted to realize the influence of different graphical models in the classification task. This is the reason why all the parameters of STR has been fixed, except for the graph topologies. We used

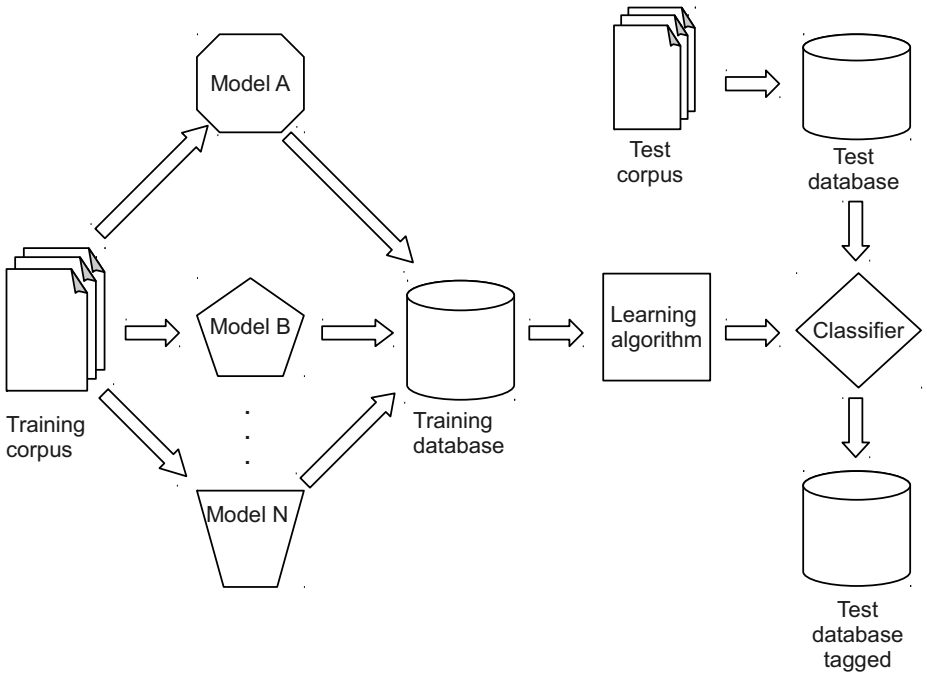


Figure 3.4 – *Stacking* schema.

the *Add-One* technique to smooth the probabilities distribution. In order to solve the unknown-words problem we opted for applying a method that assigns a set of possible tags to a word according to its suffix, in such a way that this set of tags is obtained from all the words in the corpus with the same suffix than the given word. We took the last two letters of a word as a suffix. The parameters of the TextRank algorithm were also fixed for all the experiments: the factor d was set to 0.85 (as in PageRank original description) and the threshold is 0.001.

The experimental design has been structured according to the type of graph topologies and their complexity. We detail them in the next section.

Dataset

The experiments of this work were executed over a partition of the Penn TreeBank corpus (Marcus et al., 1993), one of the most popular and widely used datasets in NLP for English.

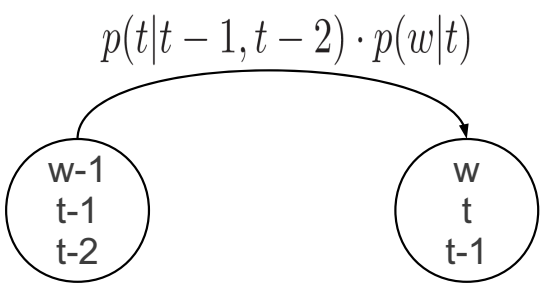


Figure 3.5 – Trigram-based graph.

Penn Treebank Project (1989-1992)[1] carried out a compilation of English texts of diverse sources, such as the Wall Street Journal and the Brown corpus, in order to build a comprehensive corpus for English. The aim of this project was to serve as research tool for researchers in NLP and Speech Recognition, as well as in theoretical linguistics

The complete dataset was automatically annotated with the POS (Part-Of-Speech) information of each word occurrence and, subsequently, reviewed manually. In addition, over half of it has been also annotated for skeletal syntactic structure using a hierarchical schema.

The version of Penn Treebank that we processed in these experiments was obtained by taking the words and their POS tags from the Wall Street Journal subset of the corpus. We used one section for testing and fifteen sections for training. The resulting corpus is formed by $766,463$ words for training and other $46,461$ words for testing. The POS tagset used in the annotation contains 36 POS tags and 12 other tags (for punctuation and currency symbols).

3.2.3 Experimental results

As the first stage of our experiments, we studied *STR* behavior with the classical bigram and trigram models (graphs in Figures 3.2 and 3.5). We also made some attempts by changing the direction of the edges in the graph, taking the input text from left to right and vice versa. Finally, we implemented a linear interpolation method between unigram, bigram and trigram models, adjusting automatically the parameters of the interpolation, using the algorithm shown in (Brants, 2000).

Results obtained with these basic models are shown in Table 3.1. Models ending with *"R"* corresponds to the original graph models with their edges

[1] `www.cis.upenn.edu/~treebank`

directed to the opposite direction (*Reverse* versions of the original models). We noted with *"B"* those experiments with bidirectional graph models (edges in both directions).

Graph Models	**Accuracy**
Bigrams	95.17%
BigramsR	94.40%
BigramsB	95.02%
Trigrams	95.15%
TrigramsR	94.46%
TrigramsB	94.48%
Interpolation	95.22%
InterpolationR	94.44%
InterpolationB	94.56%

Table 3.1 – Results of the experiments using STR with basic Markov-based graph models.

According to the Table, the best results are reached with the linear interpolation model. We also obtain a good performance with bigram and trigram graph models, with their edges directed from left to right.

Stacking experiments

In this section we show the experimental results obtained by using a *Stacking* schema within our framework. In order to check out the influence of TextRank scores in the classification task, two stacking schemas were designed. The first one involves only the graph models. In the second schema the TextRank scores of each tag were included as another feature.

Model	**Tags**	**Tags+Ranking**
Bigrams+R	95, 53%	96, 07%
Trigrams+R	95, 64%	96, 10%
Interpolation+R	95, 59%	96, 02%

Table 3.2 – Results obtained by STR in a stacking schema with basic models and their reverse versions.

We developed two experiments. In the first one the models were built by stacking the basic graphs with their reverse versions (Table 3.2). In our

second experiment the stacking database was created incrementally by considering the models in the previous experiment, from the most simple to the most complex one. That is, we took firstly bigrams, then added trigrams, and lastly the linear interpolation (Table 3.3). The effect of adding more columns to the stacking model can be supported by the corpus because of its size. The number of instances in the corpus is big enough to offer many examples for each possible features combination, therefore a classification algorithm can be applied without losing confidence in the performance. The algorithm used to classify the instances in all cases is the *J48* algorithm in Weka (Witten and Frank, 2000), that implements a decision tree.

As we can see in both experiments, the behavior of this stacking-based classifier has been improved thanks to the TextRank scores for each tag. This information has been supplied by our tool, running a pre-process over the corpus with each graph model.

Models	**Tags**	**Tags+Ranks**
Bigrams+R	95, 53%	96, 07%
Bigrams+R & Trigrams+R	95, 72%	96, 15%
Bigrams+R & Trigrams+R & Interp.+Interp.R	95, 74%	96, 16%

Table 3.3 – Experiments results using STR by stacking the bigram, trigram and interpolated versions together with their reverse models.

Results in Tables 3.2 and 3.3 show that the TextRank scores are a useful information for classification tasks. Its positive influence in this task is based on the fact that TextRank assesses the importance of all the possible tags for each word.

Comparison with other techniques

In order to compare the performance of our framework with other tools, we reproduced the tagging experiments with the following taggers:

- TreeTagger (Schmid, 1994), based on decision trees. It estimates the transition probabilities with a binary decision tree, recursively built from a training set of trigrams. The probability of a given trigram is determined by following the corresponding path trough the tree until a leaf is reached.

- Memory-based tagger (Daelemans et al., 1996) (MBT), it is an efficient implementation of *KNN* method. In the training step a set of patterns

is presented in an incremental fashion to the classifier, and added to memory. The tagger is based on the computation of the distance between the new examples and those previously stored in the training phase. The label of the least distant instance is assigned to the new examples.

- TnT (Brants, 2000), it is a supervised tagger that implements the Viterbi algorithm for second order Markov Model. The system incorporates a linear interpolation method for smoothing the probability distributions, and a suffix-based technique for handling the unknown words.

- FnTBL is an efficient implementation of TBL method presented in (Brill, 1995). It is based on the error-driven generation of transformation rules. The method succesively transforms the data to correct the error that gives the biggest win in terms of error rate. It is intended to obtain a small set of meaningful transformation rules, in order to improve the performance of the technique.

- MaxEnt (Baldridge et al., 2005), based on Maximum Entropy models. It uses a set of constraints in order to include in the model the information of the training corpus. The inference of the tags for new intances is made following the maximum entropy assumption, trying to maximize the entropy of the tags over the examples in order to achieve the best distribution of tags.

The accuracy of these tools is shown in Table 3.4. We can see that our framework outperforms (96, 16%) two of them, TreeTagger and MBT, and has similar results than the ones obtained by TnT tagger and FnTBL. These experiments show that STR technique achieves state-of-the-art results in POS-tagging task.

Taggers	Accuracy
TreeTagger	95, 52%
MBT	95, 67%
STR	96, 16%
TnT	96, 21%
FnTBL	96, 23%
MaxEnt	96, 41%

Table 3.4 – Accuracy percentage of some classifiers with Penn TreeBank corpus

3.2.4 Conclusions

STR constituted our first research work and, obviously, our first approximation to the graph-based algorithms as well as to the natural language processing area. STR is a technique to apply a graph-based ranking algorithm to an NLP classification task in a supervised machine learning approach. We also developed a tool to easily describe and build many graph topologies from a text. The variability of the graph structures can be specified with a graph description language. In order to prove the feasibility of our technique, STR has been applied to a classical corpus-based POS-tagging task over Penn TreeBank corpus.

The main aim of this work was to find out a reliable way of including information contained in a tagged corpus into a graph-based algorithm in order to deal with NLP tasks. Our hypothesis was that a well-known graph-based technique like TextRank, could take advantage of the large amount of corpora compiled for the diverse tasks in NLP, in order to improve the results achieved by the use of the existing classification methods on this field. The affirmation was confirmed by the results of the experiments, conducted in a specific and challenging task, the POS-tagging. There are a wide variety of research works intended to tackle this problem, with really good performances (around the 95% of accuracy), so the fact that our proposal achieved even slightly better results on this task shows the reliability of the technique.

This work motivated us to carry on our research in this direction: the enrichment of graph-based methods with different sources of information to deal with different tasks. These ideas have been applied mainly in the detection of dishonest behaviors in the WWW, that is the detection of web spam, that we explain in depth in Chapter 5.

3.3 PolarityRank: Automatic Expansion of Opinion Lexicons

Opinion Mining constitutes an emerging field in NLP which deals with subjectivity, affects and opinions in texts. One of the most common problems that have came up in this field is the classification of opinionated documents (or texts, sentences, words, etc. depending on the granularity of the specific problem). The aim of this task is to determine whether a text expresses a "positive" or "negative" opinion about the topic being discussed. It is usually tackled by studying the individual opinionated expressions contained in the text. An opinionated expression is a word, or more frequently a group of words, that contains an opinion.

As a key point in solving this and other common problems in *Opinion Mining*, the *semantic orientation* of the opinion expressions should be computed. It is a numeric value, usually between -1 and 1, referring to the negative or positive affective implications of a given word or phrase. These values can be collected in an *opinion lexicon*, so this resource can be accessed when needed.

In (Cruz et al., 2011a) we proposed the use of PolarityRank, a method to automatically induce domain-specific opinion lexicons from an annotated corpus. In other words, PolarityRank computes the semantic orientation of the opinionated expressions contained in texts about some specific topic (a product, a movie, a company, etc.). As we are committed to reduce the time and effort employed on this task, we research about the automatic expansion of this kind of lexicons, so we keep the number of required annotated documents as low as possible.

The method consists of two main steps. First, a small lexicon is built in the induction step, taking as input a small set of manually annotated documents. Then, in the *expansion* step, the induced lexicon is automatically expanded using a set of unannotated documents. This second step is carried out by PolarityRank over a graph of opinion expressions, according to their relations in the documents, propagating the information of the manually annotated opinion expressions through the graph in order to compute the semantic orientation of the unannotated ones.

The organization of the rest of the section is as follows. First we detail the two main steps of our method in Sections 3.3.1 and 3.3.2. Then we show the experiments performed with our proposal in Section 3.3.3. Finally, in Section 3.3.4 we point out the conclusions drawn from this work.

3.3.1 Induction

In order to generate an opinion lexicon to be used as seed for our expansion algorithm, we collect a set of textual reviews on a particular domain and manually annotate all the opinions in them. Each opinion is a tuple $(polarity, f, opW)$, where $polarity$ is $+$ (positive) or $-$ (negative), f is an opinable feature of a product, and opW is a set of opinion words from the text about the feature f. Each annotated opinion includes information about the semantic orientation of the opinion words.

Most of the times, the polarity of the opinion implies the polarity of the opinion words, but sometimes the opinion words include some *special expressions* that have to be studied in order to induce the polarity of the rest of opinion words. We distinguish two types of special expressions:

- *Negation expressions*: which invert the polarity of the rest of opinion words. For example: barely, hardly, lack, never, no, not, not too, scarcely.

- *Dominant polarity expressions*: they completely determine the polarity of an opinion, no matter which other opinion words take part. For example: enough, sufficient, sufficiently, reasonably, unnecessarily, insufficient, insufficiently, excessive, excessively, overly, too, at best, too much.

For each opinion term observed (individual words or phrases included as opinion words, once the special expressions have been removed), the final semantic orientation for a given feature is the mean of the semantic orientations suggested by each annotated opinion on that feature containing the opinion expression (we take 1.0/-1.0 for each positive/negative annotation).

3.3.2 Expansion

Starting from a big set of unannotated text reviews, we use the information provided by some annotated expressions and the conjunctive constructions of the text to expand the previously induced lexicon. The intuition behind this method is stated in (Hatzivassiloglou and McKeown, 1997): two opinion terms appearing in a conjunctive construction tend to have semantic orientations with the same or opposite directions, depending on the conjunction employed. Based on this principle, we build a graph using the nodes to represent the terms appearing in a conjunctive expression, so an edge between two nodes represents the conjunctive expression that links both terms. We compute the semantic orientation of each expression by spreading the information provided by the terms in the initial induced lexicon through the graph.

Building the graph

The first step on our proposal is the creation of the graph to be processed by PolarityRank. This graph is built by searching for conjunctive constructions between terms. Two terms participate in a conjunctive construction if they appear consecutively in the text separated by a conjunction *and* or *but*, or the punctuation mark *comma* (,). There are two types of conjunctive constructions, *direct* and *inverse*, depending on the conjunction and the negation expressions participating. In a direct conjunctive construction, both terms are supposed to share the same semantic orientation; in a reverse one, they might have opposite semantic orientations. Some examples are shown next:

- **Direct conjunctive constructions**

  The camera has a **bright *and* accurate** len.

  It is a **marvelous**, really **entertaining** movie.

  ...**clear** *and* **easy to use** interface.

  ...**easy to understand**, **user-friendly** interface.

- **Inverse conjunctive constructions**

  The camera has a **bright** *but* **inaccurate** len.

  It is a **entertaining** *but* **typical** film.

  The driving is **soft** *and not* **aggressive**.

The terms appearing in conjunctive constructions (in bold type in the previous examples) are the nodes of the graph. If two terms participate in a conjunctive construction, the corresponding nodes are linked by an edge. The edges are assigned a weight equal to the number of direct conjunctive constructions minus the number of inverse conjunctive constructions observed between the linked terms.

PolarityRank

We propose a new random-walk ranking algorithm, namely PolarityRank. It is based on PageRank (Page et al., 1999). In summary, PageRank computes the relevance of each node in a graph based on the incoming edges and the relevance of the nodes participating in those edges; an edge is seen as a recommendation of one node to another. PolarityRank generalizes the concept of vote or recommendation, allowing edges with positive and negative weights. A positive edge still means a recommendation, more strongly the greater the weight of the edge. By contrast, a negative edge represents a negative feedback, more strongly the greater the absolute value of the weight.

PolarityRank computes two scores for each node: a positive one and a negative one (PR^+ and PR^-, respectively). Both scores are mutually dependent: the positive score of a node n is increased proportionally to the positive scores of the nodes linked to n with positively weighted edges; in addition, the positive score of n is also increased in proportion to the negative scores of the nodes linked to n with negatively weighted edges. Analogous principles apply to the computation of the negative scores of the nodes.

The formal definition of the algorithm is as follows. Let $G = (V, E)$ be a directed weighted graph where V is a set of nodes and E a set of directed

edges between pairs of nodes. The edges in E have an associated real value or weight, distinct from zero, being p_{ji} the weight corresponding to the edge going from node v_j to node v_i. Let us define $Out(v_i)$ as the set of indices, j, of the nodes which have an outgoing edge from v_i. Let us define $In^+(v_i)$ and $In^-(v_i)$ as the sets of indices, j, of the nodes which have an incoming edge to v_i, whose weight is positive or negative, respectively. We define the positive and negative PolarityRank scores of a node v_i, $PR^+(v_i)$ and $PR^-(v_i)$, respectively, as shown in the Equations 3.1 and 3.2:

$$\begin{aligned} PR^+(v_i) &= (1-d)e_i^+ + d(\\ &+ \sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PR^+(v_j) + \\ &+ \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PR^-(v_j)) \end{aligned} \tag{3.1}$$

$$\begin{aligned} PR^-(v_i) &= (1-d)e_i^- + d(\\ &\sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PR^-(v_j) + \\ &+ \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PR^+(v_j)) \end{aligned} \tag{3.2}$$

where the values in e^+ and e^- are greater than zero for certain nodes acting as positive or negative seeds, respectively. The sum of the values in e^+ and e^-, separately, must be equal to the number of nodes in the graph. The parameter d is a damping factor that guarantees the convergence of the algorithm; in our experiments we use a value of 0.85 (as recommended in the original definition of PageRank). The computation of PR^+ and PR^- is performed iteratively as described by (Page et al., 1999). The algebraic proof and convergence of PolarityRank can be consulted in Appendix B.

Extending the lexicon

Based on a seed lexicon L_D containing a set of annotated opinionated expressions, and a set of unannotated reviews R'_D, the expanded lexicon L'_D is obtained following these steps:

1. Build a graph $G = (V, E)$ representing the conjunctive relations observed in R'_D, taking the expressions as the nodes, and the conjunctive relations between them as the edges, with a positive or negative weight depending on the type of conjunction.

2. For each feature of the product to be considered:

   (a) For each expression, v_i, that is associated to this feature, initialize vectors e^+ and e^- according to the values in the seed lexicon L_D.

   (b) Linearly normalize the values of e_i^+ and e_i^-, so that the sum of the values is equal to $|V|$.

   (c) Compute PR^+ and PR^-.

   (d) For each expression v_i from V, assign $SO(v_i)$ as its semantic orientation in the expanded lexicon, L'_D, where:

$$SO(v_i) = \frac{PR^+(v_i) - PR^-(v_i)}{PR^+(v_i) + PR^-(v_i)}$$

   Note that these values are contained in the interval $[-1.0, 1.0]$.

3.3.3 Experiments

In this section we report the results of some experiments intended to evaluate the quality of the feature-level opinion lexicons obtained by our method.

Data

We used a set of reviews of three different domains (*headphones*, *hotels* and *cars*). We retrieved them from Epinions[2], a website specialized in product reviews written by customers. Some reviews from the dataset were labeled, including the polarity, the feature and the opinion words of each individual opinion found. The information about the sizes of the dataset is shown in table 3.5, including the number of annotated reviews (the amount of unannotated reviews appears in parenthesis), and the number of opinions and features found in these reviews.

[2] `www.epinions.com`

Domain	**Reviews**	**Opinions**	**Features**
Headphones	587 (2591)	3897	31
Hotels	988 (6171)	11054	60
Cars	972 (23179)	8519	91

Table 3.5 – Information of the dataset. The number of unnanotated reviews available for each domain is shown in parenthesis.

		Induced Lexicon			Expanded Lexicon		
Domain	$\|R_D\|$	p	r	F_1	p	r	F_1
Headphones	9	0.9941	0.4479	0.6176	0.9193	0.7332	0.8158
	45	0.9821	0.7011	0.8181	0.9440	0.8179	0.8764
	108	0.9665	0.8038	0.8777	0.9525	0.8562	0.9018
	531	**0.9554**	0.9062	0.9302	0.9526	**0.9185**	**0.9352**
Hotels	9	0.9875	0.3333	0.4984	0.9416	0.8131	0.8726
	117	0.9823	0.7964	0.8796	0.9716	0.8802	0.9236
	324	0.9822	0.8732	0.9245	0.9775	0.9128	0.9440
	891	**0.9801**	0.9449	0.9622	0.9792	**0.9507**	**0.9647**
Cars	9	0.9894	0.4687	0.6361	0.9536	0.8262	0.8853
	117	0.9868	0.8008	0.8841	0.9712	0.8915	0.9296
	279	0.9849	0.8799	0.9294	0.9786	0.9116	0.9439
	882	**0.9847**	0.9300	0.9566	0.9831	**0.9408**	**0.9615**

Table 3.6 – Results of expansion of lexicons induced from different numbers of annotated reviews. The second and third experiments for each domain are done selecting the number of annotated reviews needed to achieve F_1 scores for the induced lexicon similar to the F_1 scores for the expanded lexicon from the previous experiment.

Experimental setup

All the experiments were done using 10-fold cross-validation. Each annotated dataset was randomly partitioned into ten subsets. The results reported for each experiment are the average results obtained in ten different runs, taking a different subset as testing set and the remaining nine subsets as training set (to induce seed lexicons). To evaluate the lexicons, we compute the recall and precision over the terms participating as opinion words in the annotated opinions of the testing set. Recall is the proportion of terms which are contained in the lexicon; precision is the proportion of terms with a correct sentiment orientation in the lexicon.

Experimental results

Table 3.6 shows the results of the evaluation of the induced and expanded lexicons. In order to figure out the improvement in precision and recall obtained by our expansion method, we induced lexicons for each domain using different numbers of annotated reviews and expanding them using the whole set of unannotated reviews. For each domain, we show the results of the experiments using only nine annotated reviews (one from each subset of reviews of the cross-validation process), and using all the available annotated reviews. The second and third experiments for each domain are those where F_1 scores for the induced lexicon is similar to the F_1 scores for the expanded lexicon from the previous experiment. Thus, we can measure the number of additional annotated reviews needed to obtain similar results without expansion. Using only nine annotated reviews, the expanded feature-level opinion lexicon achieves 0.8158 of F_1 for the *headphones* domain, 0.8764 for the *hotels* domain and 0.8853 for the *cars* domain, a far better result than using a domain-independent opinion lexicon[3]. To achieve similar F_1 scores without using the expansion method, you should annotate between six and thirteen times more reviews.

3.3.4 Conclusions

PolarityRank constitutes a reliable method to propagate the information contained in a small set of seeds through a graph with positive and negative edges. Specifically, we have tested it by tackling the task of automatically inducing domain-specific, feature-level opinion lexicons from partially annotated datasets. One of the aims of this work was to evaluate whether an opinion lexicon can be expanded using a graph-based method that, taking as input a graph built from the conjunctive relationship between the opinionated expressions and a set of manually annotated semantic orientations, computes the semantic orientations of the rest of unannotated expressions in the dataset. The results confirmed our hypothesis, showing a really good performance in terms of the quality of the expressions automatically annotated by PolarityRank.

The knowledge acquired in the previous work, to wit, the methods for enriching graph models with heterogeneous information in order to improve the performance or even to expand the applicability of the graph-based methods to other tasks, has been successfully applied in this work.

[3]We perform some experiments using the domain-independent opinion lexicon SentiWordNet (Baccianella et al., 2010), obtaining F_1 values equal to 0.7907, 0.8199 and 0.8243 for the *headphones*, *hotels* and *cars* domains.

On the other hand, the main novelty of this algorithm is the ability of processing graphs with positive and negative edges. The main idea consists in the computation of two random-walks on the graph, in order to obtain two scores for each node within it. The positive and negative scores of a node are influenced by the scores of their neighbors, depending on the kind of relations existing between the nodes.

In this dissertation, we have applied these ideas mainly in the detection of dishonest behaviors in on-line social networks (see Chapter 7), where the users can establish not only relationships representing their affinity, but also representing their disagreements. Both kinds of relationships between the users of a social network can be processed with similar ideas than those introduced in this section, in order to infer their relevance and/or their trustworthiness in the system.

3.4 Summary

This chapter contains a thorough review of two crucial research works on the course for the development of this work. The first one is STR (Ortega et al., 2011b), a graph-based supervised classification technique which processes a weighted directed graph built from each sentence in a text in order to solve a classification problem. Both, the graph topology and the weights of the edges in the graph are computed following a description encoded using a specific graph description language. The information included in the computation of the weights of the edges is extracted from a language model of the text, taking into account the co-occurrences of words (bigrams and trigrams), the frequencies of the terms, etc.

This work constituted our first serious approach to graph-based algorithms, presenting one of the founding ideas of later works: the enrichment of graph models using some a priori knowledge extracted from a corpus or any other suitable dataset that can be useful for solving the problem that we are dealing with. Furthermore, the use of a graph as the data structure allowed us to capture the different morpho-syntactic relations between the words in a sentence. This feature was very useful for successive works, where we needed to represent other type of interactions between the elements of a text.

The second research work reviewed in this chapter corresponds to PolarityRank (Cruz et al., 2011a,b). It consisted in a graph-based algorithm that propagates the information of some relevant elements to the rest of nodes in a weighted graph. The main novelty of this approach is the ability to process a graph whose edges can have negative weights. This algorithm has been

applied to compute the semantic orientations of opinionated expressions in a text. PolarityRank processes a graph built from the expressions and their relations in a text, weighting the edge between two expressions according to the type of the particles that appear between them.

This work applied similar intuitions than those in STR, enriching a graph model with features extracted from the text to be processed. But the most important novelty in PolarityRank is the inclusion of negative edges in the graph model. This feature allows us to model many kinds of problems where negative relations are allowed. Along this dissertation we show the usefulness of the information encoded in the negative relations in on-line networks.

Since every research work relies on previous knowledge and experiences which strongly influence and lead them to the attainment of their goals, the discussion presented in this chapter is very important in terms of providing an overview of the experiences that constitute our background knowledge. In this sense, the works reviewed in this chapter can help to understand the underlying intuitions and ideas that are the basis of the proposals presented in this dissertation.

Part II

Dishonest Behaviors on the WWW: Web Spam

Chapter 4

Web Spam Detection

*'cuz I'm one of the *good* guys!*
I only spam groups by accident!
Richard E. Depew
Usenet, July 13th 1993

4.1 Introduction

Maybe one of the most extended dishonest behaviors in on-line networks is the web spam, that is based on the creation of web pages for the purpose of making a search engine to rank these elements higher than they would otherwise (Najork, 2009). Spam have become a problem for Web search engines, causing negative effects in their retrieval results (Henzinger et al., 2002).

This phenomenon was firstly known in the literature as *search engine persuasion* (Marchiori, 1997). Later, it took the name of *spam* from the similar method that was being used for the massive sending of advertisements and fraudulent or simply annoying messages in newsgroups and e-mails. The origin of the application of the term *spam* to the problem of network abuse, meaning the annoying repetition of useless contents, is a skit by the Monty Python[1], the famous British surreal comedy group. In the sketch, a waitress tells a couple the menu of the restaurant whose dishes (all of them) contain spam, that is canned ham. The term spam is uttered more than one hundred times in the sketch (including a viking song repeating the word) in about 4 minutes. The repetitive and unwanted presence of the word in the sketch

[1] `http://en.wikipedia.org/wiki/Spam_(Monty_Python)`

accurately describes the bursting of hundreds of junk mails and messages that flood the mails and newsgroups. The epigraph of this chapter is one of the first uses of spam in this context, corresponding to an accidental spamming on Usenet [2] in 1993, caused by an error in an administration script implemented by Richard E. Depew. Apart from this accidental spamming, a few notorious attempts of using spam messages as a profitable tool came up in those years, such as the "Make.Money.Fast" chain message by Dave Rhodes (a nickname to hide his real identity) that encouraged people to send 1$ to every mails in the message and include their own mails in it, the "Green Card lottery" by Canter and Siegel that consisted in a fraudulent sale of green cards for immigrants in the USA, and the "Global Alert for All: Jesus is Coming Soon" message that also flooded the newsgroups, but without any known purposes.

Concerning the WWW, together with the appearance of the first search engines, many companies came up for the purpose of offering the clients not only the creation of a web site, but also the maximum visibility for them, by taking advantage of the mechanisms of the web search engines to rank the web pages according to their relevance in addition to their similarity to the user query. So this is how SEO (Search Engine Optimization) was born.

SEO techniques are those intended to improve the visibility of certain web pages in the search engines. These techniques are usually classified into two groups: *white-hat* techniques are methods that the search engines itself recommend as part of good design; on the contrary, *black-hat* techniques attempts to achieve their goal in ways that are disapproved by the search engines or even involve some kind of deception.

The first approaches to achieve this goal used *keyword stuffing* methods, based on the inclusion of some frequent words in order to get a web page to be highly ranked. Spam mechanisms have evolved from those firsts attempts becoming more sophisticated, in parallel to search engines counter-measures.

In this chapter we focus our attention on the web spam, and review some relevant state of the art web spam detection techniques. In Section 4.2, we present a possible classification of web spam and the detection methods. Then we discuss in Section 4.3 a case study in order to highlight the difficulties of this task, and also the fact that the web spam methods are constantly evolving. Finally, following the previous taxonomy of spam methods, we study separately the spam detection techniques reviewing the link-based techniques in Section 4.4, the content-based techniques in Section 4.5 and, finally, the hybrid approaches in Section 4.6.

[2] Usenet (**Use**rs **Net**work) is one of the oldest distributed discussion systems on the Internet. It was created in 1980, and it is still in use. It works similarly to an on-line forum,but in a distributed fashion.

4.2 Taxonomy of web spam and spam detection techniques

Basically, there are two forms of web spam: Self and Mutual promotion (Cormack et al., 2010). Self promotion tries to create a web page that gains high relevance for a search engine, mainly based on its content. This can be achieved through many techniques, such as word stuffing, in which visible or invisible keywords are inserted in the page, in order to improve the retrieved rank of the page for the most common queries. Mutual promotion is based on the cooperation of various sites, or the creation of a wide number of pages that form a *link-farm*, that is a large number of pages pointing one to another, in order to improve their scores by increasing the number of inlinks to them. This method is effective against search engines that employ co-citations between pages as features (e.g. PageRank (Page et al., 1999)). In the SEO's slang, the amount of visibility (or relevance) that a web page propagates to its neighbors through the hyperlinks is called *link juice*.

There are several methods intended to deal with the problem of web spam, each of them using various approaches and processing diverse information from the web pages. In this section we study a possible taxonomy for these techniques. This classification can be established in accordance to the mechanism used for the spam detection and the source of information taken as input. In this way we can distinguish the following types:

- **Ranking algorithms**: these methods take as input the web graph and compute a ranking of web pages, trying to demote the spam ones. They study the topology of the web graph (the links between the web pages) in order to decide whether a web page is spam or not. These methods aim to detect the mutual promotion strategies analyzing the structure of the network. Their objective is to avoid the appearance of spam web pages in the first positions of the ranking, given that the users of a web search engine usually pick the top pages of a result set.

- **Machine Learning algorithms**: these methods usually consist in a classification method that decides whether a web page is spam or not. They usually use the textual content of the web pages to classify them, analyzing the characteristics of the HTML of each page and extracting useful information, such as the amount of links to other web pages and the redundancy of the content. The retrieved set of statistics from the web pages are included into a database which is processed by a machine learning method (usually a SVM (Cristianini, 2001)) in order to obtain a classifier.

- **Hybrid techniques**: this group is formed by the proposals that combine the previous techniques, usually by including the results of a ranking algorithm into a classification method. In this way, they can take advantage of both, the link-based and the content-based features.

For better understanding, in the remainder of the chapter we follow this taxonomy in the study of the web spam detection techniques.

4.3 A case study: Google's no-follow and PR sculpting

The constant evolution of the web spam techniques is one of the main difficulties for the web spam detection task. In this section we illustrate this problem with a real example, studying the case of the Google's "no-follow" attribute, and its subsequent use for *PR Sculpting*, in order to improve some spam methods by tunning the distribution of PageRank over the web pages within the same web site.

In early 2005, a new form of web spam was growing together with the rise of the blogosphere. The new method was intended to take advantage of one of the basis of these websites: the user-generated contents allowed in the comments to the posts. The easiest way of exploiting this feature is to include in the comments hyperlinks pointing to the web page that we want to redirect web traffic to. In this way, the PageRank of a web page can be increased just by including links to this web page in the comments section of relevant blogs.

The *no-follow* attribute was suggested as an attempt to avoid web spam in the comments of the blogs (Splogs, from "Spam on blogs"). The proposal consisted in the automatic inclusion of the attribute "rel=no-follow" into the hyperlinks appearing in the comments of blogs. Google announced that the hyperlinks with the no-follow attribute would not influence the PageRank of the web page pointed by the link. In this way, spammers would be discouraged to place their contents in those web sites where their links could be tagged with the no-follow attribute.

This mechanism was adopted by the three major web search engines, though each of them implemented the influence of the "no-follow" attribute in slightly different ways. For example Google states that its bot does not even follow this type of hyperlinks so the pages linked by them are not indexed and do not count for the PageRank in any sense either, while Yahoo treats them as normal links except for the exclusion of their influence in the ratings

of the web pages, and Bing performs an intermediate approach between the restrictive treatment of Google and the permissive process of Yahoo.

On the other hand, some social news sites such as Slashdot.org, which allow their users to publish, rate and comment news (we make a thorough review of it in Section 6.4.1), also use this feature. At the beginning they automatically inserted "rel=no-follow" attribute into any hyper-link appearing in the comments of the users, but this caused lots of arguments in favor of removing this feature from those publications that can be proved to be honest and/or relevant for the community. The strategy followed by Slashdot was to insert the "no-follow" only in the hyperlinks appearing into comments published by user who do not achieve a certain level of trustworthiness in the social network.

However, even though the inclusion of links to web pages in order to gain high PageRank score can be discouraged with the no-follow attribute, it does not avoid the other primary aim of splogs: gaining web traffic by publishing the URL of certain web pages in visible places where the link can be visited by many users. So the mechanisms against this type of spam must not be only directed to avoid the effects of spammers' actions in the relevance of a web page, but also to impede the appearance of this kind of contents in the blogs or any other social networks, because they cause a negative image of the site.

Another drawback of the no-follow method is its unintended usefulness in order to perform *PR sculpting*. This technique consists in redistributing the internal flow of PageRank scores that are propagated through the hyperlinks (also known as *Link Juice*) within a web site, redirecting the relevance of the web pages to those that we attempt to make more visible in the web search engines. In Figure 4.1 we can see a simple example of how PR sculpting works using the no-follow attribute to obtain more relevance for certain web pages by redirecting to them more *link juice* from other web pages of the web site.

Soon after, Google (as well as the other web search engines) realized this situation and announced changes in the "no-follow"'s implementation policy. The main variation consisted in excluding the "no-follow"-ed hyperlinks from the link juice computation. Thus, given the web pages in Figure 4.1, the link juice corresponding to the main web page in case (b) would be 2 instead of 3. Anyhow, some SEO's claim that this statement is not completely true, due to the results observed by some tests run in order to assess the actual behavior of the Google bot managing this attribute.

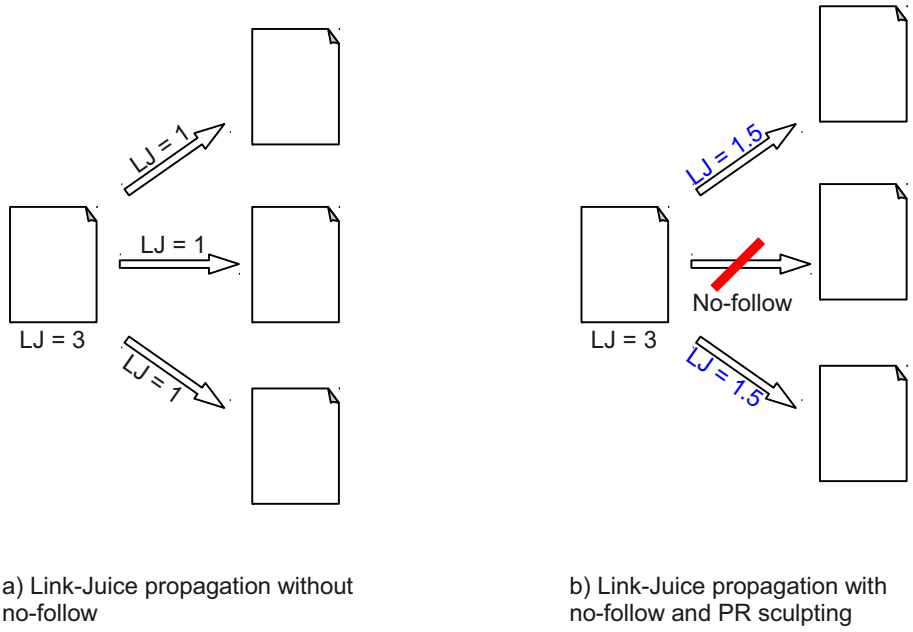


Figure 4.1 – Schema of the intuition behind PR sculpting with the no-follow attribute. The normal flow of link juice from a web page to its neighbors is shown in (a). In (b), the no-follow attribute is used for redirecting part of the link juice from the web page only to two of its neighbors.

However, PR sculpting do not necessarily needs the use of the "no-follow" attribute. Another common technique used in this context for the PR-sculpting is the use of obfuscated Javascript in order to hide the URL of the web pages that are required to be outside the link juice distribution. A simple example of this technique is shown in Figure 4.2. This method is addressed by the web search engines simply by making their bots to execute the Javascript code in a attempt of reconstructing the artificially split URL's.

As mentioned in the beginning of the section, these are some examples of the evolution of both, the web spam methods and the techniques intended to detect it. This case study illustrates some of the difficulties that emerge when facing this problem, specially those related to the continuous appearance of new threats and new mechanisms that overcome the existing detection techniques.

In the following sections we review the underlying ideas of some relevant works on web spam detection.

```
//...

function nofollowedlinks(link) {
    url = "'http://";
    piece1 ="window.";
    piece2 ="location.";
    piece3 = "href=";

    eval(piece1+piece2+piece3+url+link+"'");
}

//...
```

Figure 4.2 – PR sculpting sample using Javascript code in order to hide a URL that is not suposed to be visited by the bots of web search engines, emulating the "no-follow" attribute.

4.4 Link-based techniques

In this section we review the link-based web spam detection methods, which are intended to deal with the mutual promotion mechanisms. These techniques focus on the structure of the graph made up of the web pages and the hyperlinks among them. They study the relations of the pages in the web graph, aiming to detect linking patterns that do not correspond with the normal topology of the network.

One of these patterns is the link-farm of spam web pages (see Figure 4.3), that are formed by a set of web pages linking one to each other, causing an increased visibility due to the high amount of in-links received by all of them. The basic assumption to deal with link-farms is that similar objects are related to similar objects in the web graph (Jeh and Widom, 2002). In the context of web spam, it means that non-spam web pages are frequently related to other non-spam web pages, and vice versa.

Most of the techniques intended to deal with this problem study the *supporters* of the web pages. Given a web page, wp_i, a supporter, sp_ij, is a web page that links to wp_i and therefore the relevance of wp_i can be increased according to the relevance of sp_i.

The key assumption in (Benczur et al., 2005) is that supporters of a non-spam page should not be overly dependent on one another. In other words, if the supporters of a web page have a large numbers of links between them, they likely form a *link-farm*, and could be spam web pages. An example of

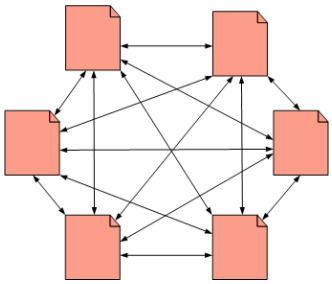


Figure 4.3 – Graphical representation of a link-farm of web pages: a set of web pages with a high number of links between them.

suspicion is the case of a page that receives its PageRank from a large number of very low ranked pages. The proposed algorithm obtains the supporters of each page, and then studies the distribution of their PageRank scores in order to compute a PageRank biased according to a vector of penalizations.

Truncated PageRank (Becchetti et al., 2006) also tackles the problem of the link-farms. It penalizes pages that obtain a large share of their PageRank from the nearest neighbors, avoiding the effect of the supporters that are topologically very close to a given node.

TrustRank (Gyöngyi et al., 2004) is based on the idea that a high PageRank score held on a huge amount of links from pages with low PageRank is suspicious of being spam. It means that a node with high PageRank and no relations with others pages with high PageRank is likely to be a spam web page. They obtain an estimator for this metric by calculating the *estimated non-spam mass*, that is the amount of PageRank received from a set of (hand-picked) trusted pages. In contrast, (Krishnan, 2006) proposes an algorithm with the same idea, but taking as input a set of spam web pages. This technique, called Anti-TrustRank, computes the *estimated spam mass* for each node.

An improvement of TrustRank is presented in (Wu et al., 2006b). In this work, they find out a weakness in TrustRank algorithm: the bias of the method towards highly represented topics on the dataset. In order to avoid this problem, they propose a topic-based partitioning of the seed set and then one computation of TrustRank for each seed set. This method obtains a trust score for each node with each seed set. The final score is calculated by combining these scores, introducing a bias in accordance to the seeds of each topic.

In (Wu et al., 2006a), an approach based on trust and distrust propagation is proposed. This work consists in an algorithm that computes two scores

for each node in the graph, indicating the levels of trust and distrust of a page. The process starts from two seed sets, trustworthy and spam pages, respectively.

Temporal link-based features are taken into account in (Shen et al., 2006). A SVM is trained with these features, in order to decide whether a web page is spam or not. They use two snapshots of a collection of web pages with a month of distance between them as datasets to evaluate the changes in the pages in that time period. Some of the temporal aspects included in the model are the in-link growth rate (IGR) of a web page and the in-link death rate (IDR), corresponding to the ratio of the increased number of inlinks and dead in-links pointing to a given web page, respectively. They also include statistics on these metrics, such as the average and variance of IGR and IDR.

4.5 Content-based Techniques

These methods are focused on determining whether a web page is spam or not according to its textual content. They usually examine the distribution of statistics about the contents in spam and not-spam web pages, such as the number of words in a page, the HTML invisible content, the most common words in a page in relation to the ones in the entire corpus, etc. (da F. Costa et al., 2005; Fetterly et al., 2004). In general, content-based methods tackle self promotion mechanisms.

In (Mishne et al., 2005) they propose the comparison of language models to classify texts as spam or non-spam. Other works apply machine learning techniques based on SVM (Kolari et al., 2006; Sculley et al., 2007), taking as features different heuristics, such as the anchor text of the links in a web page, the tokenized URL of the page, or the meta-tag text. In (Ntoulas et al., 2006) several spam detection metrics are studied. They compare the values of this content-based metrics for spam and not-spam web pages, and discuss the discrimination ability of each metric to detect spam. Some of the proposed heuristics are the number of words in the title of a web page, the average length of words, the amount of invisible content, the compression rate of the web pages, the fraction of anchor text with respect to the total amount of text in a page, etc.

A method using temporal information is proposed in (Dai et al., 2009). In this case, the temporal features are based on the textual content of the web pages, such as the average and deviation of the term weights between the web pages in different time points. They train a binary classifier for each feature, and finally combine all the decisions using logistic regression. In (Erdélyi et al., 2011) another combination schema is proposed, applying

ensemble selection (Caruana et al., 2004) in order to use a large collection of classification methods, amongst them a naive bayes classifier, decision trees, logistic regression and random forests (Breiman, 2001).

4.6 Hybrid approaches

A system that combines content-based and link-based features is proposed in (Castillo et al., 2007). They discuss three methods to include features related to the web graph topology into a classifier: clustering the host graph, assigning the label of all hosts in the cluster by majority vote; propagating the predicted labels to neighboring hosts; and using the predicted labels of neighboring hosts as new features, performing another classification with the augmented database. Some well known link-based algorithms are used in this work, such us TrustRank, PageRank, and Truncated PageRank, as features of the classifier.

Splog (*Spam on Blogs*) detection is studied in (Lin et al., 2007). They work with the assumption that spam messages exhibit highly stable content, temporal and link patterns. Their splog detection system includes some metrics intended to estimate the regularity of the messages in blogs. They define an autocorrelation heuristic that evaluates the similarity of the textual content of blog posts over time (content regularity). They also study the blog post intervals, assuming that splog messages are automatically generated, so they will be published at regular intervals. Finally, the link regularity is meassured by an adaptation of HITS in order to detect the blog posts that target a small set of websites, that are likely to be splogs.

The impact of spam in information retrieval systems, and the effects of some anti-spam filters are studied in (Cormack et al., 2010). They use three filters in this work and a naive Bayes classifier to combine all of them. The first filter is a classifier built from a labeled corpus with spam and non-spam pages. The second one consists of a set of documents retrieved by some of the most popular queries to a web search engine. And finally a set of documents extracted from the Open Directory Project[3]. They show the improvements achieved in some of the systems participating in TREC 2009[4] applying a spam detection technique.

WITCH (Web spam Identification Through Content and Hyperlinks) is presented in (Abernethy et al., 2008b). This method combines a set of content-based features with hyperlink data using a discriminative model, specifically a SVM. The link-based information is included in the SVM by

[3] `http://www.dmoz.org`

[4] `http://trec.nist.gov`

way of graph regularization. This method adds to the basic formula of a SVM an additional term intended to detect the similarity between two neighbors in the web graph. The assumption is that two web pages are linked if they are similar.

Other mechanism used for combining the information from the textual content and the links between documents is Latent Dirichlet Allocation (Blei et al., 2003). In (Bíró et al., 2009) they propose the application of an extension of this generative model to deal with web spam detection. The method includes in the LDA model an additional distribution that encodes the relation between the linked documents.

Recently, some hybrid methods for web spam detection have been presented in the context of the ECML/PKDD Discovery Challenge 2010 competition [5] on Web Content Quality. A dataset was specifically crawled for this task, and the organizers also provided a feature set consisting in tf-idf meassures, counts for sentences, tokens and characters, POS tags, bigrams, chunks, etc. The first place of the competition was shared by the works presented in (Geng et al., 2010) and (Sokolov et al., 2010). In (Geng et al., 2010) the set of features is used in the spam detection system, in addition to some link-based metrics, such as PageRank, Truncated PageRank and DomainPR (rank value of the domain corresponding to a host). The learning method used in this work is bagging (Breiman, 1996). In (Sokolov et al., 2010), three propagation methods are presented: a RankBoost method that trains a ranking model for each category of documents, a regularization-based method assuming that related documents must have similar rank values, and finally an iterative algorithm able to handle complex propagation schemes (such as propagation of scores between different classes of documents) by learning a discriminant without making any a priori assumptions.

Other works presented in this competition are (Nikulin, 2010) and (Lex et al., 2010). The first one propose an ensemble (linear combination) of two models, a random forest (Breiman, 2001) and a gradient boosting machine (Friedman, 2000). The second one also presents an ensemble-based approach using three methods: a decision tree (the J48 implementation of Weka (Hall et al., 2009)), a SVM and a centroid-based technique (Class-Feature-Centroid Classifier (Guan et al., 2009)). The ensemble is performed by way of a majority voting schema.

[5] `http://www.ecmlpkdd2010.org`

4.7 Summary

In this chapter we have discussed one of the most persistent problems in the WWW, the web spam. We have summarized the origins of the phenomenon and we have also provided a brief etymology of the use of the term *spam* in this context. Then, we proceeded to state a taxonomy of web spam mechanisms intended to illicitly promote a web page in a web search engine, distinguishing two basic types: the self-promotion through the inclusion of specific textual contents in the web pages, and the mutual promotion taking advantage of the link structure of the Web. Furthermore, a taxonomy has been proposed for the spam detection techniques in order to facilitate their study.

Then we have discussed a real case that clearly shows one of the main difficulties in the web spam detection task: the constant evolution of the methods intended to gain high relevance for certain web pages, and the necessity of developing a web spam detection method with a great adaptability in order to tackle new threats and new spam mechanisms. In particular, we have discussed the creation of the "no-follow" attribute in order to fight against the spam on blogs and other social networks, that was lately used by SEO's to perform PR sculpting in order to redistribute the PageRank scores of a web site towards those web pages that they want to promote.

Finally, following the classification of spam detection techniques previously mentioned, we have provided a background on the state-of-the-art web spam detection techniques focusing firstly on link-based methods, second the content-based ones, and finally the hybrid approaches that combines concepts and ideas from the two previous types.

Once we have reviewed the state-of-the-art on graph-based ranking algorithms and web spam detection, we have the necessary background to explain our proposal in this field, PolaritySpam (see Chapter 5), that consists in a hybrid system that includes content-based information into a link-based algorithm, unlike the majority of hybrid approaches reviewed in this chapter, that includes link-based features into content-based methods.

Chapter 5

Graph-based Ranking Algorithms Applied to Spam Detection

Two years from now,
spam will be solved
Bill Gates
World Economic Forum, 2004

5.1 Introduction

Together with the appearance of the earlier web search engines, intended to provide the users with a set of relevant web pages given a user information need or query, many companies came up for the purpose of offering the clients not only the creation of a web site, but also the maximum *visibility* for them. In terms of the World Wide Web, visibility means web traffic, and an increased web traffic brings valuable business results. In general, the higher ranking on the results and more frequently a site appears in the search results list, the more visitors it will receive from the search engine's users. So the main goal of these companies was to gain higher rank for their web pages in order to obtain a high visibility and, as a consequence, to improve their commercial benefits.

As we have explained in Chapter 4, SEO (Search Engines Optimization) techniques comprise those actions intended to achieve this goal. Some of these techniques are even encouraged by web search engines, such as the improvements in the design of the web pages to make them more easily nav-

igable. On the other hand, other methods are based on the diversion of the web traffic to undesired web pages through different mechanisms. Obviously, this type of methods, namely *black hat SEO techniques*, are disapproved because they deceive both, the web search engines and specially the users.

Basically, web spam is a phenomenon where web pages are manipulated for the purpose of obtaining some kind of benefits by illicitly gaining web traffic. The first approaches intended to achieve this goal used *keyword stuffing* methods, based on the inclusion of some frequent words in order to get a web page to be highly ranked. Web spam mechanisms have evolved from those first attempts becoming more sophisticated, in parallel to search engines counter-measures. Nowadays, spammers use a wide variety of methods intended to make a search engine delivering undesirable results and ranking these web pages higher than they would otherwise (Najork, 2009). There are two basic forms of web spam: Self and Mutual promotion (Cormack et al., 2010). Self promotion tries to create a web page that gains high relevance for a search engine, mainly based on its content. It can be attained through many techniques, such as the above mentioned keyword stuffing, in which visible or invisible keywords are inserted in the page, in order to improve its rank for the most common user queries. Mutual promotion is based on the cooperation of various sites in order to benefit each other.

There are several approaches intended to deal with the problem of web spam using different information sources to decide whether a web page is spam or not. We have reviewed some of the most important ones in Chapter 4. We have shown that some techniques are based on document classification approaches and try to identify the spam web pages according to some features. Other techniques are focused on the computation of a ranking of web pages, trying to demote the spam web pages in the last positions of the ranking. The intuition behind these methods is that a user who uses a web search engine will take mainly the top ranked documents retrieved by the system, so the goal is to avoid the appearance of spam web pages in those first positions.

In this chapter we present PolaritySpam as our proposal for web spam detection. It is a method that includes concepts from link-based and content-based techniques, combining the strongest points of them. Our intuition consists in giving a high reputation to those pages related to relevant ones, and giving a high spam likelihood to the pages linked to spam web pages. We introduce the novelty of including the content of the web pages in the computation of an a-priori estimation of the spam likelihood of the pages, and then propagate this information. PolaritySpam is based on a propagation algorithm that spreads over the graph this a-priori information, that is computed regarding the textual content of the pages. Unlike other web spam

detection systems, our method does not need any human intervention in the process. In spite of this, it achieves very good results in the task.

The chapter is organized as follows. In Section 5.2 we present PolaritySpam algorithm and its two main features: the selection of sources regarding the textual content of the web pages, and the propagation algorithm. The evaluation settings are shown in section 5.3, where all the aspects concerning to the setup of the experiments are detailed and discussed. Then, we show the experimental results for all the techniques in Section 5.4, comparing the results of our techniques to a very hard baseline (TrustRank algorithm) using some useful evaluation metrics. In this section, we study the performance of each component of our system. Finally, we provide a brief summary of the chapter in Section 5.5.

5.2 PolaritySpam

PolaritySpam (Ortega et al., 2010, 2012) is a web spam detection method intended to build a ranking of web pages according to their spam likelihood, demoting in the ranking those web pages that are likely to be web spam. Our proposal is based on similar ideas as Topic-sensitive PageRank (Haveliwala, 2003), a biased ranking algorithm over graphs. This graph-based algorithm is intended to obtain a ranking of documents from a search engine, giving more relevance to the documents whose topic is related to the user query. With this aim, a score evaluating the similarity of the topics of each document and a set of pre-established topics is computed before the processing of the query. Once the similarity of topics is obtained, the PageRank algorithm is biased according to those values in such way that the possibility of a document appearing in the first positions of the ranking is increased according to the similarity between its topic and the topic of the user query. The bias in the PageRank algorithm is achieved by setting up the e_i vector with higher values to those nodes which are required to be strengthened in order to slant the algorithm towards them, as we already explained in Section 2.2.1.

The idea of assigning a pre-computed score to each document in a collection according to certain criteria, in order to include a bias in the ranking algorithm can be easily adapted to the task of web spam detection. In our approach, we also propose a biased-ranking algorithm, but we evaluate the spam likelihood of each web page instead of the similarity of topics.

The intuition behind PolaritySpam is the propagation of spam and not-spam likelihood of web pages over the web graph. It relies on the *topical endorsement* assumption, previously stated: if a page, w, focused on topic T, points to page v, we can consider that page v is relevant for topic T. In

terms of web spam, this statement is translated into: every web page that is related to a spam-like web page, must be considered as spam, and vice versa. In other words, each page propagates its spam likelihood to their neighbors, so every web page will receive good or bad reputation regarding to the pages that are pointing to them.

Following this reasoning, we need some kind of a-priori knowledge about the web pages in order to determine their spam likelihood before starting the propagation. At this point, we propose the use of some heuristics to extract this information from the textual content of the pages. These metrics are the basis of an automatic process intended to obtain an a-priori score for each page, weighting their spam likelihood. Then, we compute a *spam-biased* random-walk algorithm to propagate this a-priori information through the web graph. We take advantage of the links in the graph to spread the content-based information over the network. In this way, we can promote those pages related to relevant web pages, because they will receive high positive scores. On the contrary, those web pages related to spam-like pages are demoted in the ranking because they will receive high negative scores.

Therefore, PolaritySpam consists in a graph-based algorithm that propagates certain a-priori information of the web pages through the graph, in order to build a ranking of them. Unlike other ranking algorithms, PolaritySpam computes two scores for each node: a positive score representing the authority of a web page, and a negative score which represents the spam likelihood of a page. The difference between both scores is taken into account in order to build a ranking of web pages. Intuitively, this value represents the overall relevance of a web page. In this way, web pages with high negative scores are demoted in the final ranking, because they are likely to be spam.

In Figure 5.1 we show the general schema of our system. It consists of three main components: the Content Evaluator that processes the textual content of the web pages in order to obtain a score representing their a-priori spam likelihood; the Selector of Sources that automatically selects two sets of spam and not-spam like web pages, respectively, regarding the a-priori information previously computed; and the Propagation Algorithm that processes the web graph, propagating the a-priori information of the sources according to the links between the web pages.

Moreover, we propose three variations for our system, each of them presenting different mechanisms for the characterization of the web pages according to their textual content. In this way, we specify three different *Selectors of Sources* that automatically select and characterize a number of web pages (henceforth called *sources*) to spread their positive or negative influence to the rest of the network. They establish the weight of each source to be propagated regarding their a-priori information. Since we compute two scores

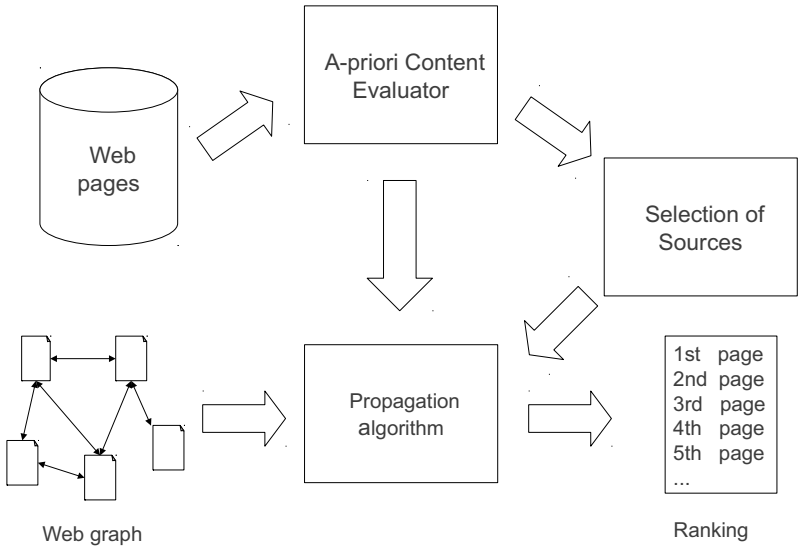


Figure 5.1 – Structure of our Spam detection System.

for each web page, we need two sets of sources, each of them intended to reinforce the positive or negative relevance of each type of web pages in the graph. Thus, the first set of sources must contain a group of non-spam pages, and the second one consists in a group of spam pages.

In the remainder of this section, we discuss the content-based metrics and their role in our system (Section 5.2.1), the automatic methods to obtain the sets of sources(Section 5.2.2), and finally the algorithm proposed to propagate the content-based information of the web pages over the network (Section 5.2.3).

5.2.1 Extracting a-priori information

The intuition behind the use of the content-based metrics in conjunction with a random-walk algorithm lies on the propagation of the spam likelihood of the web pages over the graph. We propose the use of the information extracted from the textual content of the web pages as spam likelihood indicators. In this way, we provide some a-priori information to our system in order to start the propagation graph-based algorithm.

The content-based metrics that we use in the experiments of this work have been chosen according to their discriminative ability, distinguishing be-

tween spam and not-spam web pages, as identified by (Ntoulas et al., 2006). In this work, they downloaded a big corpus (about 105 million web pages) from the Internet using the MSN Search crawler [1]. According to that work, "the MSN Search crawler is biased towards well-connected, important, and high-quality pages". This is an important quote, due to the fact that the resulting corpus studied in that work has suffered some filtering processes, so the results obtained are not completely extensible to the Web in general. However, it is a comprehensive analysis of content-based features for web spam detection, for this reason we decided to take it as source of knowledge when it comes to select two simple content-based heuristics that also must be in principle as effective as possible.

When we say that the metrics must be *simple*, we are talking in terms of their computational complexity. Following these criteria, we have implemented two heuristics:

- **Compressibility**: is defined as the fraction of the sizes of a web page, x, before and after being compressed.

$$Compressibility(v_j) = \frac{UnZippedSize(v_j)}{GZippedSize(v_j)}$$

  A web page with a very high compressibility value, is likely to be a spam. This heuristic is intended to detect repeated content or words in a web, because more redundant content leads to a higher compression ratio.

- **Average length of words**: non-spam web pages have an average word length around 5-6 (in English), while malicious pages have much higher values of this metric. Hence, this heuristic penalizes the use of word stuffing mechanisms.

In the next section, we explain the different ways of including these heuristics into our system, initializing the graph-based algorithm.

5.2.2 Selection of Sources

PolaritySpam uses the content-based heuristics to automatically assign a-priori scores to some specific pages (*sources*), regarding their spam or not-spam likelihood. A given page will be demoted or promoted in the final ranking in accordance to its relations with these sources. We use sources

[1] `http://search.msn.com`

sets to ensure that negative scores of negative pages will be propagated over the graph, and analogously for the positive sources. The sources sets are represented in our approach by two spam-biased vectors, e^+ and e^-. The vectors contain the spam and non-spam likelihoods of the web pages taken as sources in our algorithm, giving higher positive or negative scores to those nodes that have higher e^+ or e^- (see Equations (5.3) and (5.4)).

In this section, we introduce three different ways to characterize the spam-biased vectors, given the heuristics of each web page.

Most Spamy/Not-spamy Sources (S-NS)

This first method chooses as sources the N most spam-like and not-spam like pages in the graph according to their metrics. Formally, given a page v_i, let $M(v_i) = m_{i1}, m_{i2}, ..., m_{ik}$ be a vector with the k content-based metrics for v_i. The spam likelihood of v_i will be determined by the norm of $M(v_i)$, as shown in Equation (5.1):

$$Spaminess(v_i) = \sqrt{\sum_{m_{ij} \in M(v_i)} m_{ij}^2} \tag{5.1}$$

where v_i is a web page, and m_{ij} represents the heuristics which $M(v_i)$ contains.

In this way, we take the N nodes with highest *Spaminess* as negative sources, and the N nodes with lowest *Spaminess* as positive sources. The spam-biased vectors can be defined as follows:

$$e_i^+ = \begin{cases} 1/N & \text{if } i \in S^+ \\ 0 & \text{otherwise} \end{cases} \qquad e_i^- = \begin{cases} 1/N & \text{if } i \in S^- \\ 0 & \text{otherwise} \end{cases}$$

where N is a parameter that specifies the number of sources that will be taken. S^+ and S^- are the set of the N nodes with lowest and highest *Spaminess* in the graph, respectively. The formula is analogous for vector e^-.

Content-based weighted Spamy/Not-spamy Sources (CS-NS)

Following the previous schema, we can take advantage of the content-based metrics by including the specific scores directly in the computation of the weights of the sources, as shown in Equations (5.2):

$$e_i^+ = \begin{cases} \frac{Spaminess(v_i)}{\sum_{j \in S^+} Spaminess(v_j)} & \text{if } v_i \in S^+ \\ 0 & \text{otherwise} \end{cases}$$

$$e_i^- = \begin{cases} \frac{Spaminess(v_i)}{\sum_{j \in S^-} Spaminess(v_j)} & \text{if } v_i \in S^- \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

where $Spaminess(v_i)$ is the norm of the vector built from the metrics of the page v_i (see Equation 5.1).

Content-based Graph Sources (C-GS)

The last method consists in applying the Equation 5.2 to every nodes in the graph. We can rely on the thresholds proposed in the study in (Ntoulas et al., 2006), and use them to determine whether a page must be a negative or a positive source. The thresholds for the metrics considered are shown in Table 5.1.

Heuristics	Threshold
Compressibility	6.0
Average length of words	9.0

Table 5.1 – Thresholds for the content-based metrics.

Given a web page, if any of its metrics is above the corresponding threshold, we include the page in the set of negative sources, and in other case it will be taken as a positive source. Once the sets of nodes have been defined, we apply the same formulas shown in Equations (5.2).

5.2.3 Propagation Algorithm

As mentioned above, we propose a propagation algorithm in order to spread the information extracted by the content-based metrics over the graph, and to compute a ranking of the web pages according to their relevance. This algorithm is intended to demote the spam web pages in the overall ranking by computing two scores for each page, PS^+ and PS^-. Given a page A, it is desirable that its positive score, PS^+, depends on the good pages pointing to A, and analogously for the negative score, PS^-. In other words, we want the spam web pages to propagate their negative scores to their neighbors, and the positive pages do the same with their positive scores. With this aim, two vectors, e^+ and e^-, are built based on a set of content-based metrics from each page. These spam-biased vectors are used in the computation of our random-walk algorithm, representing the non-spam and the spam likelihoods of a web page, respectively. They can be thought of a reinforcement for the

positive and negative scores of each page. Having said that, the proposed scores are obtained as shown in Equations (5.3) and (5.4) below:

$$PS^+(v_i) = (1-d)e_i^+ + d \sum_{j \in In(v_i)} \frac{PS^+(v_j)}{|Out(v_j)|} \tag{5.3}$$

$$PS^-(v_i) = (1-d)e_i^- + d \sum_{j \in In(v_i)} \frac{PS^-(v_j)}{|Out(v_j)|} \tag{5.4}$$

where v_i is a node of the graph (a web page), $In(v_j)$ is the set of nodes pointing to v_j, and $Out(v_j)$ is the set of nodes which v_j points to. Both scores, PS^+ and PS^-, are obtained with a PageRank-like algorithm. The algorithm iterates over the nodes in the graph, applying Equations (5.3) and (5.4). This process is performed a fixed number of times or until the maximum difference between the scores in one iteration and the previous one, is lower than a given threshold.

Once the propagation algorithm is finished, a ranking of web pages is built regarding the difference between PS^+ and PS^- for each node, as shown in Equation (5.5).

$$score(v_i) = PS^+(v_i) - PS^-(v_i) \tag{5.5}$$

5.3 Evaluation

In this section, we show the experimental design that has been defined to show the performance of PolaritySpam, as well as the dataset used. We also detail the values of the parameters for each set of experiments, and the different variants proposed in this work.

The aim of the experiments is to show the performance of PolaritySpam in terms of the demotion of spam web pages in the resulting ranking, in order to asses its usefulness in the web spam detection task. The results of PolaritySpam are compared to a state-of-art web spam detection technique, TrustRank (Gyöngyi et al., 2004), which achieves very good results in this task.

Since PolaritySpam does not classify the web pages between spam or non-spam, it does not make sense to perform an evaluation in terms of classification accuracy. On the other hand, we use in our experiments two methods intended to evaluate ranking techniques: the Normalized Discounted Cumulative Gain (nDCG), and thePR-buckets which is the same evaluation method followed in other works on the application of graph-based algorithms to the spam detection task (Benczur et al., 2005; Gyöngyi et al., 2004). This

kind of spam detection techniques are based on the fact that a user usually looks only at the top ranked documents of a result set, ignoring the rest of lower-ranked documents. So the goal of ranking algorithms in the web spam detection task is to demote the spam web pages in such way that they became less visible to the users due to their low rankings. Therefore, the evaluation methods applied in this work take into account the positions obtained by the spam web pages in the rankings computed by the implemented techniques, in order to measure their performance.

The organization of the rest of this section is as follows. The dataset used in the experiments is presented in Section 5.3.1. The evaluation methods, *PR-buckets* and *nDCG* metric, are explained in Sections 5.3.2 and 5.3.3, respectively. Finally, in Section 5.3.4 we present the TrustRank algorithm, the method taken as baseline in the evaluation of our approach.

5.3.1 Dataset

The corpus used in the experiments is the WEBSPAM-UK2006 Dataset (Castillo et al., 2006), a huge collection of web pages specifically crawled for researching on spam detection. It contains more than 98 million pages. The collection is based on a crawl of the .uk domain performed in May 2006. It was collected by the Laboratory of Web Algorithmics, Università degli Studi di Milano, with the support of the DELIS EU-FET research project. The collection was labeled by a group of volunteers and by domain-specific patterns such as .gov.uk or .ac.uk. Of the 11,402 hosts in UK2006 dataset, 7,423 are labeled as spam or non-spam. For the evaluation purposes, we have considered as spam any web page that belongs to a host labeled as spam. Following this procedure, there are about 10 million spam web pages in the collection.

5.3.2 Evaluation with PR-Buckets

Since the position of spam web pages in the ranking of documents is the key for our spam detection technique, we focus on this feature in order to evaluate the performance of our approaches. In this section we explain the PR-buckets method, introduced in (Gyöngyi et al., 2004) to evaluate their proposal. The aim of this method is to easily get a qualitative evaluation by determining the number of spam web pages detected mainly in the highest positions of the ranking of web pages, that are the most pernicious for the performance of a web search engine.

This evaluation method is implemented as follows. First, a list of pages is generated in decreasing order of their PageRank score. This list is segmented into N buckets. The size of each bucket depends on the PageRank of the

web pages within it. Given a bucket, b_i, let us define the total PageRank of the bucket, $PR_{total}(b_i)$, as follows:

$$PR_{total}(b_i) = \sum_{\forall j \in b_i} PR(j)$$

We must ensure that the sum of the PageRank scores of all the pages within each bucket is the same. So given a bucket b_i, its size is limited according to:

$$PR_{total}(b_i) = \frac{\sum_{\forall k \in C} PR(k)}{N}$$

where C is the collection of all the web pages in the dataset and N is the number of buckets to be created. We show in Figure 5.2 an example of the computation of 4 PR-buckets, given a collection of 12 web pages. As it is shown, the first bucket contains only one web page, while the forth bucket contains 6. It can be seen that the property mentioned before is accomplished, because $\forall 1 \leq i \leq 4, PR_{total}(b_i) = 0.25$.

The remaining web pages at the end of the ranking are included in the last bucket, which will contain the less important web pages regarding their relevance in the collection. Since this last bucket is not relevant in terms of evaluating the demotion achieved by a web spam detection technique, this decision does not affect the analysis of the experimental results.

In the experiments in Section 5.4, the sizes of the PR-buckets are taken in order to build a similar set of buckets using the rankings computed by each web spam detection method. In this way, given N_1, the size of the first PR-bucket, we take the N_1 top ranked web pages for each technique in order to build the first bucket corresponding to each result set, and so forth.

The number of spam web pages per bucket is our evaluation metric. It is obtained by counting the number of pages in each bucket that are labeled as "spam" in the dataset. The aim of a spam detection technique is to avoid spam web pages into the first buckets (top positions of the ranking), demoting these pages in order to put them into the last buckets, so the lower is the metric for the first buckets, the better is the result of the technique.

5.3.3 Evaluation with nDCG

Another method intended to evaluate the results of ranking algorithms is the *Normalized Discounted Cumulative Gain* (Najork, 2007)($nDCG$), a well-known metric widely used in Information Retrieval for the evaluation of a set of retrieved documents given a query. This metric measures the quality of a ranking of documents, penalizing the appearance of top-ranked irrelevant

- Size of the Collection: 12 - Number of Buckets: 4

- Ordered List of PageRanks:

WP's	PR's
WP1	0.25
WP2	0.13
WP3	0.12
WP4	0.11
WP5	0.08
WP6	0.06
WP7	0.06
WP8	0.055
WP9	0.055
WP10	0.045
WP11	0.020
WP12	0.015
Sum	1

$$PR_{total}(b_i) = \frac{\sum_{\forall k \in C} PR(k)}{N}$$

$$PR_{total}(b_i) = \frac{1}{4} = 0.25$$

WP's	PR's	Buckets
WP1	0.25	1
WP2	0.13	2
WP3	0.12	2
WP4	0.11	3
WP5	0.08	3
WP6	0.06	3
WP7	0.06	4
WP8	0.055	4
WP9	0.055	4
WP10	0.045	4
WP11	0.020	4
WP12	0.015	4
Sum	1	

Figure 5.2 – Schema of the computation of 4 PR-Buckets for a collection of 12 web pages. The sizes of each bucket are, respectively: $\mid b_1 \mid = 1$, $\mid b_2 \mid = 2$, $\mid b_3 \mid = 3$, $\mid b_4 \mid = 6$.

documents in the results. It is based on the Discounted Cumulative Gain (DCG), computed as follows:

$$DCG = Relevance_1 + \sum_{i=2}^{N} \frac{Relevance_i}{\log i}$$

where N is the total number of documents in the collection, and $Relevance_i$ is defined as follows:

$$Relevance_i = \begin{cases} 1 & \text{if } \textit{document in position } i \textit{ is Relevant} \\ 0 & \text{otherwise} \end{cases}$$

In our case, a relevant document is a not spam web page. As we can see, the higher are the positions of irrelevant documents in the ranking, the lower is the DCG score of the ranking. $nDCG$ is obtained by normalizing the previous score using the $IDCG$, that is the best DCG that can be achieved with the given dataset:

$$IDCG = 1 + \sum_{i=2}^{|Relevance|} \frac{1}{\log i}$$

where $|Relevance|$ is the total number of relevant documents in the collection. The normalized score is obtained as follows:

$$nDCG = \frac{DCG}{IDCG}$$

where $0 \leq nDCG \leq 1$, corresponding $nDCG = 1$ to the perfect system that gets all the irrelevant documents ranked in the last positions of the ranking. So, with $nDCG$ we can evaluate the overall performance of each technique in terms of the demotion of spam web pages in the ranking. The higher is the value of $nDCG$, the better is the demotion of spam web pages achieved by the given technique.

5.3.4 Baseline: TrustRank

TrustRank (Gyöngyi et al., 2004) is a link-based algorithm that takes as input the web graph and a set of non-spam web pages, chosen in a semi-supervised way, that are the sources for the algorithm. The output is a ranking of web pages according to their relevance, in which the spam web pages are demoted. TrustRank computes a score for each web page, similarly to PageRank, using the source set to include a bias in the random-walk algorithm.

In order to build the source set, they propose an *inverse PageRank* algorithm, that computes a ranking of web pages, but taking into account the out-links of the web pages instead of their in-links. Once the inverse ranking has been built, they choose by hand the N top-ranked non-spam web pages from that ranking. In this way, they try to take as sources the N good pages that reach as many nodes as possible, trying to maximize the propagation of the information from these sources. The drawback of this method is that human intervention is required, so it is expensive to consider a high number of web pages as sources due to the manual effort needed to select the non-spam web pages from the top ranked web pages according to the inverse PageRank.

We test this technique with the UK2006 dataset, and it achieves very good results. The results shown in Figure 5.3 have been obtained with a damping factor of 0.85, as suggested in (Gyöngyi et al., 2004). We have manually built a source set with more than 500 non-spam web pages. We can see in the chart that there are less than 3% of spam web pages into the

two first buckets. However, more than the 10% of pages in the third bucket are spam.

5.4 Experimental results

PolaritySpam is composed of two main components: the propagation algorithm, and the selector of sources. In this section we evaluate both of them. The ranking algorithm is evaluated in Section 5.4.1, where their results are compared to those obtained by TrustRank technique. On the other hand, the performance of the selector of sources is studied in Section 5.4.2, where we discuss the influence of the content-based metrics in the overall performance of our system. These experiments follow the settings specified in Section 5.3.

5.4.1 PolaritySpam evaluation

In this section we show the results obtained by our approach and its three methods for the selection of sources, comparing their results with the TrustRank algorithm. We have used the same value for the damping factor as other techniques (0.85).

For the S-NS and CS-NS methods we have selected the 5% of the most positive and negative nodes as the positive and negative source sets, respectively. The results for the first ten buckets are shown in Figure 5.3, where TR corresponds with TrustRank algorithm; S-NS (Spamy/Not-spamy Sources) is our first approach, that takes the N most positive and negative pages as sources and assigns an a-priori weight of $1/N$ to each of them; CS-NS (Content-based weighted Spamy/Not-spamy Sources) is our second approach, based on the previous one, but setting the weights of the sources according to the content-based metrics; and finally, C-GS (Content-based Graph Sources) corresponds to our third approach that takes every node as a source for the random-walk algorithm.

In Figure 5.3 we can see that TrustRank performs very well, demoting a high number of spam web pages in the ranking, and allowing only a few of them to appear in the top positions of the ranking. Our techniques present a really good behavior as well, avoiding the spam web pages in the first bucket (except for S-NS). CS-NS shows the best results of all the techniques, successfully avoiding the spam web pages in the first and second buckets, and achieving a very low number of errors in the rest of them. Our third approach, C-GS, also outperforms the baseline, showing a really good performance in these first buckets.

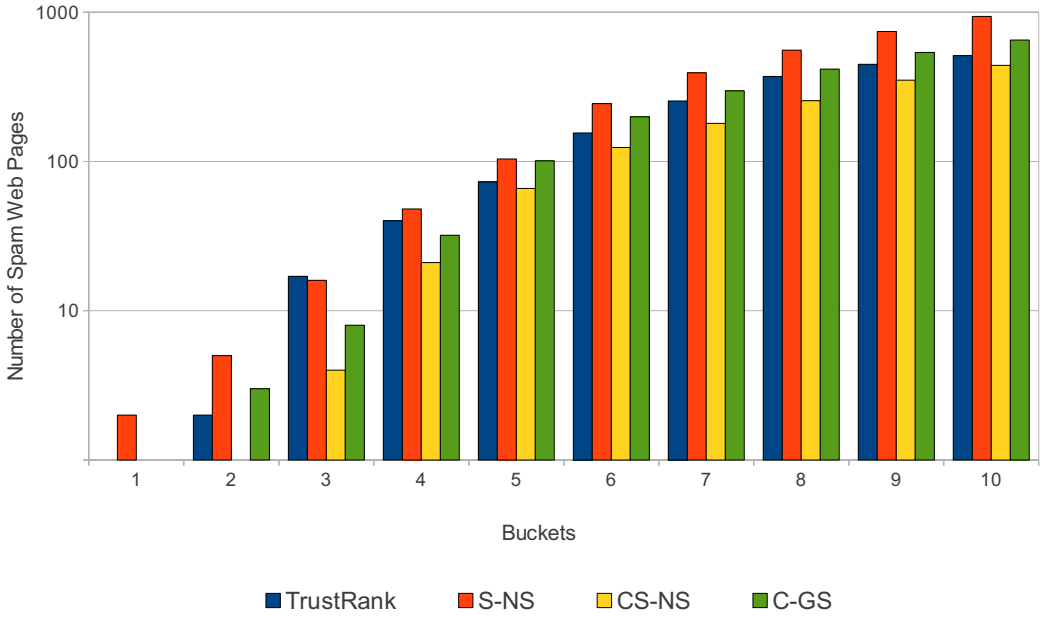


Figure 5.3 – Number of spam web pages per bucket for each technique in the first ten buckets.

In Table 5.2, we show the number of spam web pages per each of the twenty buckets. The first column represents the bucket order and the second one contains the total number of web pages from the first bucket to the current one, inclusive. The rest of the columns show the accumulated number of spam web pages for each technique, that is the total number of spam web pages from the first bucket to the current one, inclusive.

We can see that CS-NS achieves the best results outperforming TrustRank. In contrast, the first approach does not improve the TrustRank algorithm. This fact, in addition to the results of CS-NS, highlights the relevance of including the content-based metrics in the weights of the sources, and also gives us the idea of the need to improve the mechanism for the selection of the sources. Finally, the C-GS method also presents good results, with only 32 spam web pages in the first 649 pages, outperforming TrustRank in those first buckets as well.

Buckets	Pages	TR	S-NS	CS-NS	C-GS
1	14	**0**	2	**0**	**0**
2	68	2	5	**1**	3
3	212	17	16	**4**	8
4	649	40	48	**21**	32
5	1719	73	104	**66**	101
6	3849	155	244	**124**	199
7	6513	254	392	**180**	297
8	9291	371	557	**255**	416
9	12102	448	742	**350**	537
10	14914	511	937	**440**	650
11	17726	653	1112	**543**	781
12	20538	860	1292	**543**	957
13	23350	942	1460	**698**	1103
14	26162	1237	1885	**888**	1299
15	29673	1532	2310	**1060**	1495
16	38002	1827	2735	**1615**	1992
17	16103632	1554162	**693105**	940855	1810597
18	44043924	4657553	**3221681**	3383010	4589582
19	71984216	8138745	**6607552**	7379871	8080963
20	98812333	**10181905**	**10181905**	**10181905**	**10181905**

Table 5.2 – Accumulated number of spam web pages for each method: TrustRank (TR), Spamy/Not-spamy Sources (S-NS), Spamy/Not-spamy Sources with metric-based weights (CS-NS) and Content-based Graph Sources (C-GS). The best results are shown in bold.

In Figure 5.4, we show a graphical comparison of the performance of all the techniques. For each bucket b, we compute the proportion of good pages present in the ranking up to that bucket, according to the formula:

$$Precision(bucket_b) = \frac{\sum_{\forall bucket_i <= bucket_b} NumberOfGoodPages(bucket_i)}{\sum_{\forall bucket_i <= bucket_b} NumberOfPages(bucket_i)} \tag{5.6}$$

This metric highlights the evolution of the performance of each technique as we advance in the ranking of web pages. We must focus our attention again in the first buckets where we try to avoid the appearance of spam web pages, although we plot in Figure 5.4 the precision of each technique for all the buckets to show the global behavior of all the methods.

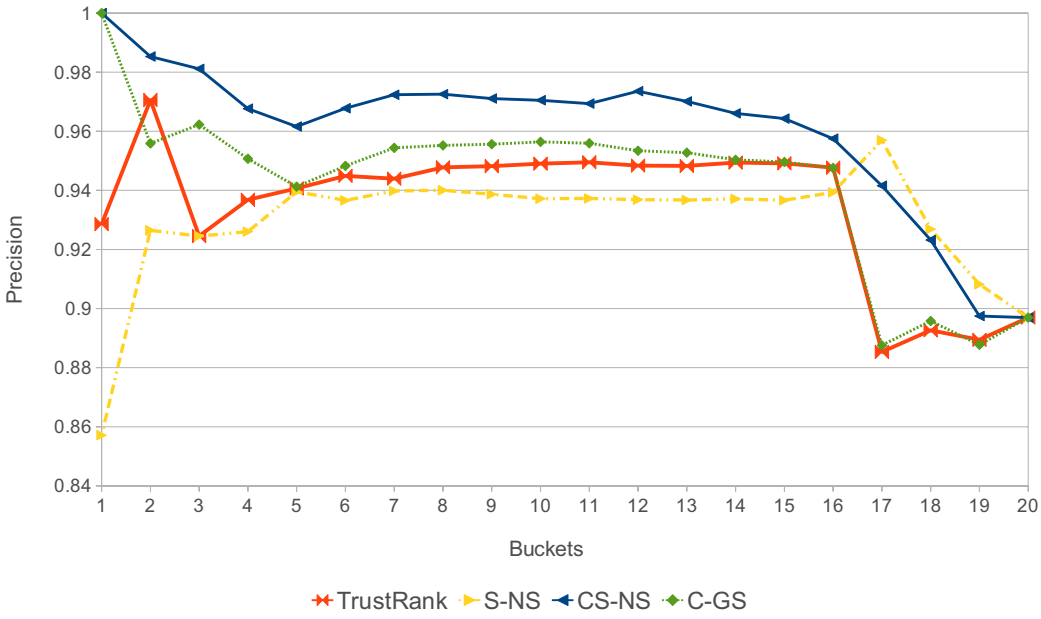


Figure 5.4 – Precision computed for each bucket as shown in Equation 5.6

In Figure 5.4 we can see that our first approach, S-NS, under-performs the TrustRank algorithm in the first buckets, making some mistakes in the top positions of the ranking, and it is near the baseline in the rest of the collection. On the other hand, the proportion of good pages per bucket for CS-NS is always better than TrustRank, successfully demoting the spam web pages into the last positions of the ranking. Our third approach, C-GS, also presents a good behavior for all the collection, avoiding a higher number of spam web pages than TrustRank in the first positions, except for the second bucket. In this last case, although the inclusion of metrics in the link-based algorithm has proved to be effective, the selection of the sources using specific thresholds seems to slightly penalize the accuracy of the method, causing some errors in the technique because of relying too much on the two simple content-based metrics.

Finally, as a summary of the different performances of each technique, we show in Table 5.3 the $nDCG$ obtained by each method. This metric highlights the positions that spam web pages have reached in the ranking computed by each technique.

Table 5.3 confirms the results showed by the PR-buckets evaluation, showing that our approaches CS-NS and C-GS clearly outperform the results of TrustRank, while the most simple S-NS has the lowest *nDCG* as expected according to the previous results.

	nDCG
TrustRank	0.7381
S-NS	0.4230
CS-NS	**0.8621**
C-GS	**0.8648**

Table 5.3 – nDCG scores obtained by TrustRank and PolaritySpam with its three variations

According to *nDCG*, both CS-NS and C-GS are equivalent in terms of the demotion of spam web pages. This metric is more accurate to evaluate the performance of ranking-based techniques than PR-Buckets, because the last one groups the web pages into buckets, considering all the web pages contained in a bucket to have the same position, so it does not take into account the actual ranking of each web page.

5.4.2 Evaluation of the impact of the selection of sources

In the previous section we have tested the performance of the PolaritySpam technique in comparison to TrustRank, a very good method for the detection of web spam. In order to make a comprehensive study of our proposal, we carry out in this section an analysis of the impact of the component intended to select and characterize the positive and negative sources of our algorithm. With this aim, we focus our attention in the content-based metrics used in the experiments.

As mentioned before, we have performed the evaluation of PolaritySpam including only two very simple content-based metrics: the *compressibility* of a web page, that measures the repetition of words in the content of a web page because a high compressibility implies a higher amount of repeated content in the document; and the *average length of words*, that detects the phenomenon of keyword stuffing and other methods that alter the language model of a web page, because this kind of spaming usually generates longer words, and hence a higher average of lengths, in the documents. These metrics have been used in many works on web spam detection (Abernethy et al., 2008a; Castillo et al., 2007; Ntoulas et al., 2006) in addition to a huge set of other features.

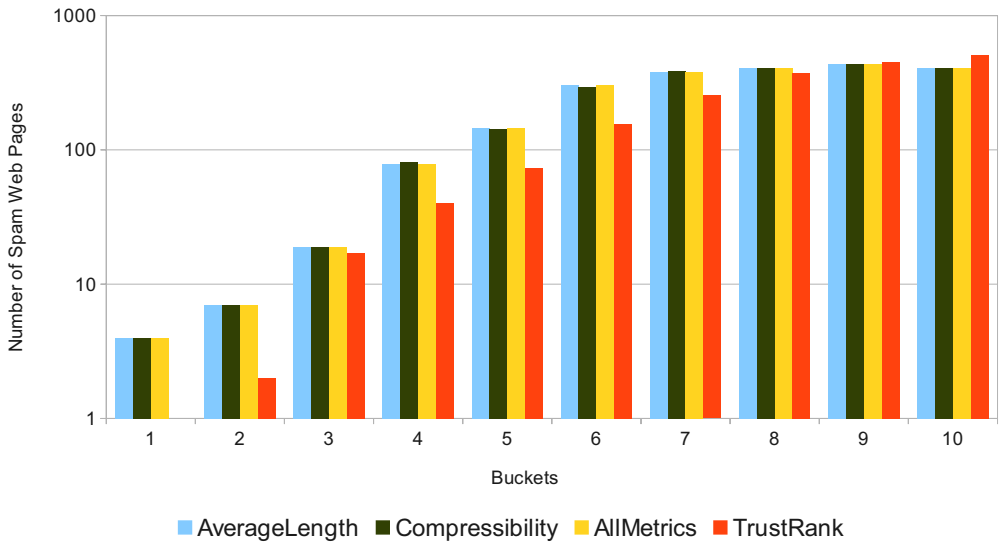


Figure 5.5 – PR-buckets for the rankings built using the content-based metrics, in comparison to the results of the TrustRank algorithm.

With the aim of studying the impact of the inclusion of the content-based metrics into our system, two different rankings of web pages have been built by assigning to each page the value of their metrics separately, and another ranking has been created with the combination of both heuristics. These three rankings are compared to TrustRank algorithm in Figure 5.5. The results obtained by the metric-based ranking methods are clearly worse than those computed by TrustRank algorithm, though in some buckets (the fifth one, for example) are slightly close.

In this case the use of just the content-based metrics is not enough to obtain a reliable ranking method for the web spam detection task. Both metrics, which presented really good performances with the corpus studied in (Ntoulas et al., 2006), do not perform very good with the UK2006 dataset. This can be caused by the different characteristics of the datasets. As mentioned before, the corpus used in the work by Ntoulas et al. was crawled from the MSN Search crawler, that applies some filters in order to retrieve high quality and well-connected web pages. On the contrary, UK2006 dataset (crawled from the .uk domain in May 2006) has been built with the aim of having a representative sample of the WWW in that domain, so it has not been focused on the quality or the connections of the webs downloaded. This

is why we find a high amount of low-quality web pages (regarding their textual content) in UK2006 dataset, and even a lot of isolated web pages (the average degree of the nodes in the web graph is significantly lower in the UK2006 dataset).

In Figure 5.6 we study the impact of the inclusion of the content-based metrics into our system. We plot in this figure the results obtained by a simplified version of our proposal, implemented without including any information about the content of the web pages. The results of the complete version of PolaritySpam are also included, in addition to the metric-based rankings computed just with the heuristics.

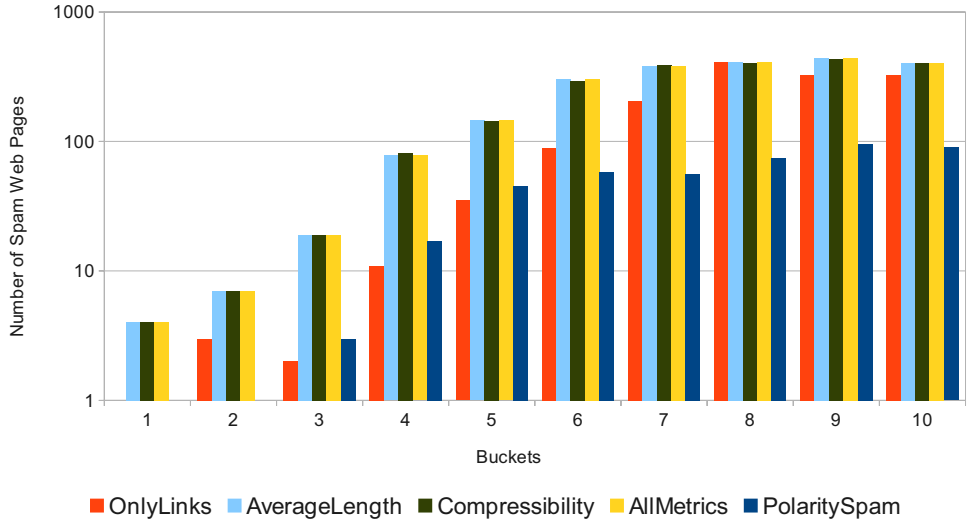


Figure 5.6 – Performances of five rankings: the first one using the PolaritySpam algorithm without metrics; the next three have been built with the values of each metric and a combination of them; and the last one corresponds to the results obtained by PolaritySpam with the metrics.

We can see that the ranking computed without the use of any content-based information obtains better results in general than the rankings based just on the metrics. It means that, in this dataset, the knowledge provided by the links between the web pages is more useful for the detection of web spam. However, this method under-performs the hybrid system consisting in the link-based algorithm enriched with the heuristics. Thus we can confirm that the combination of both content-based and link-based information results in a more reliable method for dealing with the web spam problem.

5.5 Summary

In this chapter, we have introduced a new method, PolaritySpam, to deal with the web spam detection problem. It combines concepts from link-based and content-based techniques to avoid the negative effects of spam web pages in a web search engine. Our approach consists in a graph-based algorithm that uses a set of automatically chosen web pages as sources whose information will be propagated over the network. The information to be propagated is obtained by the aggregation of two simple content-based metrics, that are also used for the selection of the sources. Other methods for web spam detection, such as TrustRank, use similar ideas but the selection of the sources is a manual process which limits the number of sources that can be taken into account. Our proposal avoids this limitation without compromising the performance of the system, as it is shown in the experiments.

Furthermore, we have proposed three methods for the selection and characterization of the sources in the algorithm, showing the flexibility of our system, because it allows not only the inclusion of new content-based heuristics to improve the a-priori information extracted from the web pages, but also the implementation of different techniques for the propagation of this information. In fact, the three methods shown in this work have proved to be very reliable in the spam detection task.

Regarding to content-based techniques which include link-based features in their training data, PolaritySpam takes into account this information implicitly, because the ranking algorithm follows the topology of the web graph in the propagation step. It means that our system implements a lower number of metrics (only the content-based metrics need to be computed, not the link-based ones), so the off-line computation of them is a lighter process.

In relation to other link-based spam detection techniques, the inclusion of information about the textual content of the web pages into the algorithm has been shown as a good method avoiding the problem of isolated pages, i.e. pages without any in-links or out-links. In many cases, this situation constitutes a difficult case for link-based web spam detection techniques, due to the lack of information about the spam likelihood of these web pages if just the link-based information is taken into account. Indeed, despite not being able to obtain any feedback from their neighbors, the textual content itself provides useful criteria to make a decision about the spam likelihood score for these pages.

On the other hand, PolaritySpam can be easily extended to include other content-based heuristics in the propagation algorithm, or to combine different methods for the selection of sources such as the inverse PageRank method proposed in (Gyöngyi et al., 2004). These features can be useful to improve the performance of our technique, and can also make our system more flexible in terms of dealing with new possible spam strategies.

Part III

Dishonest Behaviors on Social Networks: Trust and Reputation

Chapter 6

Trust and Reputation in Social Networks

It is an equal failing to trust everybody, and to trust nobody
English Proverb

6.1 Introduction

Social networks have experienced a great expansion in the past few years, covering a wide variety of themes and functionalities, and allowing their users to share many kinds of contents and to establish different types of relationships between them. One of the first on-line social networks is the Blogosphere, a network formed by the web-logs and the links between them. The users of blogs share their opinions about many topics, and the relationships can be seen as affinities on these opinions. Nowadays, we can find social networks focused on sharing very diverse contents, such as images, music or videos (Flickr, Youtube, Grooveshark). There are also a number of *social news sites* where users share interesting news on many topics (Slashdot, Digg). Usually all these systems provide their users with *social functionalities*, for example voting the contents shared by other users. Microblogging services (Twitter, Tumblr) were born primarily out as social networks for sharing short text messages. Although, they are actually being used for sharing all kinds of on-line contents and news through URL's to other sites. This fact has provoked the proliferation of a number of related services in order to make it easier to share all types of contents, even between different social

networks. On-line marketplaces (eBay, Bizrate, eBid) can be also considered as social networks, as long as they allow their users to establish relationships between them, and to give opinions about other users, expressing their level of satisfaction on the buying/selling experience. This is a very useful feature, because it helps buyers to choose the most trustworthy sellers, and vice versa.

One point in common for the majority of the social networks mentioned before is the necessity of qualifying their own contents in order to provide a better service and to improve the user experience. In social news sites and other content-sharing networks it is very useful to take advantage of the user opinions in order to give more relevance to some contents over others. On the other hand, in on-line marketplaces it is crucial to distinguish untrustworthy sellers or buyers, so these systems usually allows their users to evaluate their transactions.

Most of the systems provide the users with the ability of giving their opinions about other users or the contents generated by them in order to evaluate their reputation in the network. Hence, in this context, the *reputation* of a user can be defined as the assessment of the trustworthiness of a user in a social network, in accordance to his behavior in the system and the opinions of the rest of the users in the network. From other point of view, the reputation of users can be viewed as the evolution of the concept of *relevance* (of web pages) from the Web 1.0 to the Web 2.0.

The problem comes out when a user or a group of users take advantage of the system in order to gain any kind of benefits. For example, in an on-line marketplace, a dishonest seller would want to gain high reputation in order to increase his sales. Or maybe a user in a social news site would try to give as much visibility as possible to news with a specific biased opinion about some topic. All these actions can provoke negative consequences in the services provided by these sites, disturbing the normal behavior of the social networks. *Trust and Reputation Systems* (from now on *TRS*'s) are intended to deal with this problem, avoiding the effects that users with dishonest behaviors can cause in a social network.

In this chapter we give an overview of the state-of-the-art in the field of trust and reputation in social networks, analyzing the vulnerabilities and possible ways of exploiting the weaknesses of trust and reputation systems in Section 6.2. We also discuss a case study in Section 6.3, focusing our attention in Digg.com and some of the problems, related to the trust and reputation system, that it has suffered. Finally, the main techniques intended to tackle this task are reviewed in Section 6.4.

6.2 Robustness of TRS's

The correct performance of a trust and reputation system can be at risk due to many different causes. In this section we classify these dangers into two groups: the first one is formed by the inherent vulnerabilities of TRS's due to the nature of the environment and the information to be processed (Section 6.2.1), and the second group contains the threat models of malicious behaviors that users can carry out in order to harm the performance of a TRS (Section 6.2.2).

6.2.1 General Vulnerabilities of Trust and Reputation Systems

The required information to have an idea about the trustworthiness of a user in a social network comes from the rest of the users in the system, in other words: it is a user-generated data. Since this is an external resource that is not produced by the system itself, there exist some problems or risks that must be taken into account in a TRS, or at least the system must be aware of them.

In this section we review some of the most common vulnerabilities of the Trust and Reputation Systems (Cheng and Hurley, 2010; Hoffman et al., 2009; Jø sang and Golbeck, 2009; Kerr and Cohen, 2009; Marti and Garcia-molina, 2006) caused by the nature of the input data that is processed by these systems.

- **Incentives for feedback provision**: the first problem that appears when the data to be processed is a user-generated content is precisely the lack of this feedback information in the system. It usually happens when the users are not encouraged enough to the (annoying) task of providing their feedback to the system. Some on-line systems propose giving incentives to those users who are more active in this sense. These incentives depend on the main topic of the social network or the services that it offers to its users. In this way, content-sharing networks can provide a higher download rate to users who interact with the system, or contents with more quality (for example, videos with higher resolutions). Other incentives can be extra features or functionalities, unlimited use of the resources of the system, or even money.

  Recently, gamification (Deterding et al., 2011) has also proved to be a reliable method to include in the social networks a really useful set of tools for providing incentives in order to encourage users to interact with a system. Gamification is a set of techniques based on applying

the ideas of the games in order to engage the audience to behave in some specific ways. Common mechanisms of gamification are the use of points (or stars, or levels) that users can obtain by carrying out some actions in the system. These points can be also used for filling up leader boards or progress bars in order to add a competition ingredient in the system. A good example of the use of gamification to provide incentives to users in social networks is FourSquare[1], a location-based social network where users are encouraged to share the different places that they visit (using the terms of the site: "to check in"). The incentive consists in the appointment of the user who provides the highest number of "check-in's" as the *Mayor* of the place.

- **Opinion bias**: some works on social network analysis point out the existence of a bias in the majority of the ratings provided by users, towards giving positive or negative scores, depending on the topic of the network. This bias must be taken into account due to the relevance of the opinions that are opposed to the majority in the network. Although giving more relevance to the outliers can lead the system to strengthening some votes made by mistake or for revenge. The other solution in these situations is to smooth the weight of the opinions whose polarity, positive or negative, constitutes the majority in the network.

- **Cold-Start Problem**: when a user arrives to a social network there is no information about him on the system because he has not performed any actions and other users have not interacted with him yet. This common phenomenon constitutes the cold-start problem. It is a demanding situation for TRS's because it is hard to estimate the trustworthiness of the newcomer due to the lack of information about him. Some TRS's tackle this problem by not allowing the newcomer to perform specific (maybe threatening) actions until he receives a certain amount of ratings from other users. The cold-start problem leads to the next vulnerability, the re-entry problem.

- **Re-entry Problem**: as mentioned before, the cold-start problem comes up every time that a new user account is created in a social network. The re-entry problem is a *reputation whitewashing* mechanism that takes advantage of the cold-start problem in order to allow a user to start from fresh. This vulnerability is based on the creation of a new account by a user that has been identified as a malicious one in the TRS due to his inappropriate behavior. In this way, the malicious

[1] `foursquare.com`

user gets a new identity that does not suffer any consequences for the wrong behavior performed with his previous account.

- **Exit Problem**: the malicious behavior of a user may damage his reputation, so the rest of the social network will react in a specific way against future actions of this user. This reaction can consist in limiting the features that the system provides to the user, a decrease in the interactions that other users perform with him, or even the suspension of the user account. The problem comes up when the user is planning to leave the social network, so he has no further need for his reputation. Thus, he can behave as he wants without consequence or, to say it better: without worrying about it. This is a extremely hard problem for TRS's.

6.2.2 Threats for Trust and Reputation Systems

In this section we study some common tactics that can be adopted by users who want to gain some kind of benefits by manipulating the TRS of a network. Several threat models are presented in (Donato and Stefano Leonardi, 2008; Kamvar et al., 2003b). They take the example of a P2P network for file sharing, in order to explain the methods used by malicious users to achieve their goal. In many senses, a social network can be viewed as a P2P network, in terms of a decentralized network where users can share different resources (texts, videos, images, etc).

It is assumed that good users, in terms of trust and reputation, will always be honest, so they will receive positive links from the rest of good users. In this situation, the main threat models to interfere in the overall ranking of trust can be described as follows:

- **Individual Malicious Peers**. Malicious users always present a bad behavior, so they receive negative links from good users. In fact, this model represents the absence of attacks against the network, because the behavior of each type of user is just as expected, so the ranking of trustworthiness is not affected. An example of this threat is shown in Figure 6.1. Good users are represented by green circles, and malicious users are the red ones. Green solid lines represent positive votes, and the negative votes are highlighted by red dashed lines.

- **Malicious Collectives**. Also known as *orchestrated attacks*, they exploit the possibility for bad peers to assign positive trust values to other malicious users (see Figure 6.2-1). These attacks are based on the same ideas behind the *link-farms* formed by spam web pages in

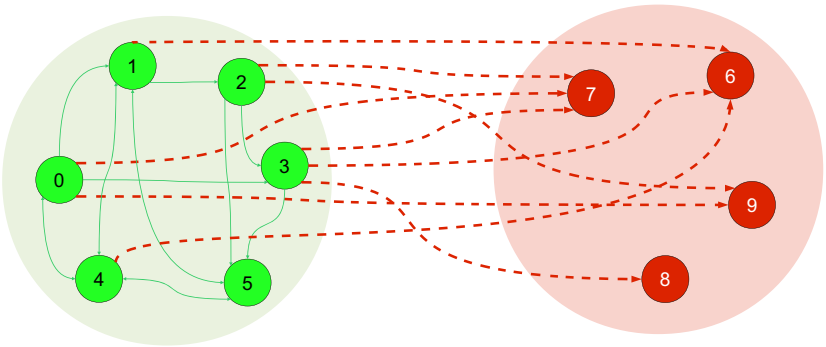


Figure 6.1 – Graphical representation of threat type A: good users (green circles) vote negatively (red dashed lines) for malicious users (red circles)

order to gain high relevance to the web search engines. This type of attacks is intended to take advantage of the TRS's that include in their computation the number of in-links to the users. In this way, the ranking of malicious users can be increased due to the amount of positive in-links received.

- **Camouflage behind good transactions**. Through this attack, malicious peers can make some good users vote positively for them. It is based on the idea of showing one face or another, depending on the user that they are interacting to in each moment. In this way, malicious users can obtain high reputation by showing a trustworthy face to some users in the network, who can vote positively for them. It is graphically represented in Figure 6.2-2, where positive (green solid) edges from good users to malicious ones can be observed. The effect in the network is that some bad users can receive sporadically positive votes from a good user.

- **Malicious spies**. There are two types of malicious users: some of them act as in any of the other threat models; another group of them, called *spies*, make good users to vote positively for them, and then assign positive trust values only to bad nodes. It is a more elaborate version of the camouflage behind good transactions, because the malicious spies attack needs to create different user accounts to work properly, while the camouflage behind good transaction is performed by the same user who wants to obtain high relevance. Node 5 in Figure 6.2-3 is an example of a malicious spy. It receives positive links from good users, and provide positive votes to malicious users.

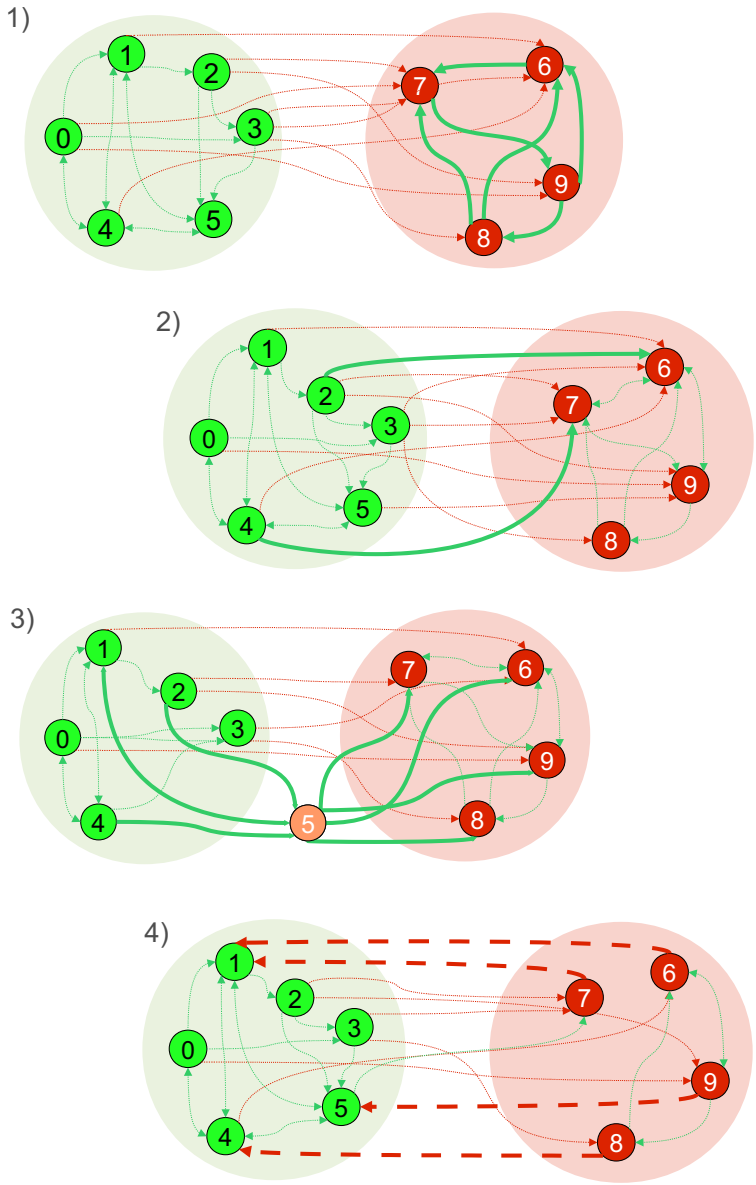


Figure 6.2 – Graphical representation of threat models. Green nodes represent good users, and the red ones are malicious users. Green lines represent positive votes and red lines are the negative ones. Threat models: (1) Threat model B. (2) Threat model C. (3) Threat model D (the node 5 is a spy). (4) Threat model E (red dashed lines represent negative votes).

- **Camouflage behind judgments**. In this model, malicious peers assign negative trust values to good users. This attack is shown in Figure 6.2-4, highlighting the negative votes (red dashed lines) from malicious to good users. This strategy can cause the decrease of trust for good peers and, as a consequence, the promotion of the malicious peers in the ranking of trust.

As we have already stated, these attacks can be combined in order to achieve better results, from the point of view of an attacker.

6.3 A case study: Digg.com

The problems and threats studied in previous sections give an overview on the possible ways of disturbing or cheating a TRS by a malicious user or a group of them. In this section we review the case of the popular social news site Digg.com, that has been the object of many attacks against its reputation system.

6.3.1 What is Digg.com?

Digg.com is a social network created in 2004, based on similar ideas as Slashdot.org (see Section 6.4.1), which consists in allowing their users to share interesting news and comment and rate them, in order to show the most relevant ones, according to the readers, in the main page of the site. The huge growth of this type of social networks caused the appearance of other similar websites, such as Reddit [2], Meneame, or Fark [3].

These web sites usually take advantage of gamification techniques (see Section 6.2.1) in order to encourage their users to contribute with publications, comments or votes. The most frequently used method is to define badges that are bestowed on those users who achieve certain amount of relevant contributions to the site. It is usually measured by counting the number of publications of a user that have appeared in the main page of the site, because these are the most relevant news according to the users of the social network. In particular, in Digg.com a positive vote is called *digg*, while a negative one is a *bury*.

Concerning the web traffic of this type of social network, the chart in Figure 6.3 shows the statistics of use of some of the most popular social news

[2] `www.reddit.com`

[3] `www.fark.com`

sites, according to Google Trends [4]. We can see that Digg has been the most popular news site, meaning its appearance the drop of users in Slashdot. Nevertheless the other sites, particularly Fark, have been slowly growing and attracting a higher number of users.

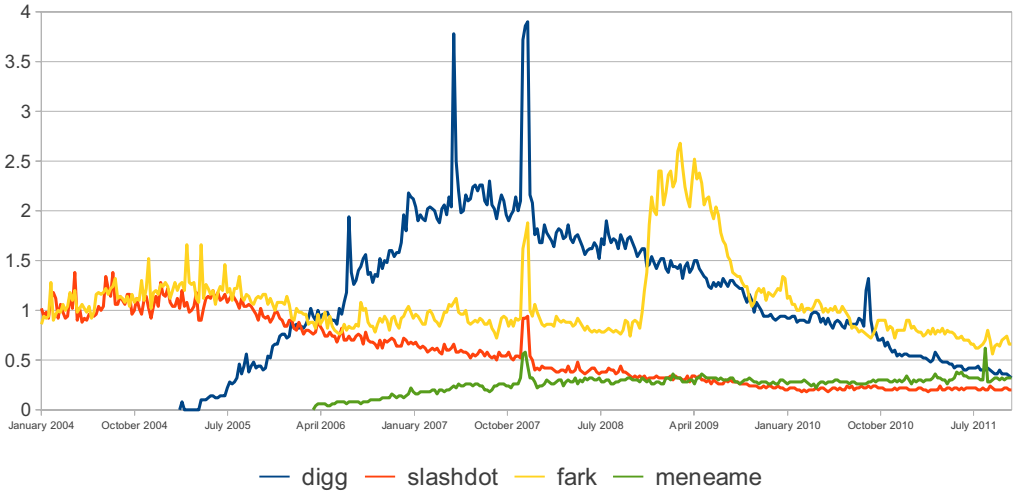


Figure 6.3 – Search volume index of Digg.com, Slashdot.org, Meneame.net and Fark.com, according to Google Trends. The peaks of all the sites in the chart probably corresponds to the US Elections in 2008.

Each of these social networks uses its own trust and reputation system in order to evaluate the trustworthiness of the users and the relevance of the publications. In particular, Digg.com implements a private algorithm that takes several factors into consideration, including among others the number and diversity of diggs, the user reports, the time that a story was submitted and its topic, etc. Though we cannot go into further detail about the algorithm, the administrators of the site claim that one of the keys to promotion is the element of diversity, that tries to ensure that a publication is relevant not only for the top "diggers" of the site, but also for the long tail of low-rated users.

Digg is in constant evolution, being improved with new features and designs. For example, one of the most recent versions of the site introduced the concept of personalized news page for each user. This personalized page is built from the most relevant news that have been published or commented or "dugg" by the friends or the trusted sources of a given user. This new

[4] `www.google.com/trends`

aspect makes Digg.com more similar to traditional RSS feeds aggregators[5], but with the social aspect of the user-generated content (news, comments, ratings for both of them). The personalization of the website has recently made another step further with the addition of the *Newsrooms*, that consist in specific clusters of publications grouped by topics, and rated according to the diggs and buries of the most relevant users in the social network, and also taking into account other features, such as the number of "Likes" from Facebook and the related tweets in Twitter. In this way, Digg tries to aggregate not only the opinions from its own users, but also the reactions produced in other social networks.

6.3.2 Gaming Digg.com

Even though Digg.com has experienced a number of reviews, adaptations and new versions (the current one is the forth one), it has suffered several attacks, some of them certainly notorious. As we mentioned previously, they have not publicly discussed the algorithm in charge of computing the relevance of the news, but it has not been proved to be a useful safety precaution. Actually, in these cases the exact specification of the algorithm is not as relevant as the intuitions implemented in it, such as the actions considered as bearers of a positive (digging) or negative (burying) consequence for a publication, or the reputation of the user who has posted it in terms of the relevance of his previous publications, etc.

One of the most popular incidents related to this matter was announced in 2010[6], when an orchestrated group of conservative users formed a coalition intended to slant the news appearing in the main page of the site was uncovered. This group, known as *"DiggPatriots"*, is formed by no more than 100 users that collude in order to vote up or down those publications that did not fit in with their ideology. In this way, a relatively small group of users was able to slant the news published in the main page of Digg.com, disturbing the normal performance of a social network with thousands of users.

In relation to the threat models previously studied, since many of the members of the *"DiggPatriots"* were power users of Digg.com, it is clear that they used a malicious spies strategy to obtain high reputation in the system, together with the formation of an orchestrated collective intended to perform a camouflage behind judgments to bury the undesired news.

[5]An RSS aggregator is an application that allows the user to subscribe to certain web sites in order to periodically search for new contents of the given sources. Social news sites can be seen as an evolution of these applications.

[6]`http://www.guardian.co.uk/technology/2010/aug/06/digg-investigates-claims-conservative-censorship`

In order to avoid (or at least just to hamper) these kinds of incidents, Digg.com announced changes in some of the characteristics of their algorithm, taking away part of the influence that power users exerted on the system and including some new features intended to minimize the effect of orchestrated attacks, such as the promotion of news with a high diversity of diggs and user reports.

Apart from these minor updates, it has been mentioned above the step towards a more personalized interface of the website. Actually the aim of these improvements is to give more relevance to the notion of local trust, over the global trust ratings that were computed in previous versions of Digg. In this way, the ratings and opinions of a user and its close relations become the key factors that affect the relevance of the publications in the personalized page of that user.

Despite of all these efforts, Digg.com and other social news sites are seen as *piece of cakes* in terms of possibilities to be easily exploited for illicit purposes, such as slanting the contents of the network towards certain ideology or point of view about some topic, or even to complement any black or white hat SEO techniques in order to obtain high relevance for a web page attracting web traffic by using a highly visited web site.

6.4 On-line Trust and Reputation Systems

The relevance of Trust and Reputation Systems has grown in the past few years, in parallel to the huge expansion of social networks. The problem of managing the trustworthiness of the users of an on-line community is not new, and it has been faced with diverse methods depending on the context and the aim of the network where it is applied. We analyze in this section some relevant examples of systems that have been created to deal with this task, in order to see the big picture of the state-of-the-art techniques on this field. First, we review in Section 6.4.1 the TRS's of some well-known social networks that implement different mechanisms to manage the reputation of their users into the network. Then we study in Section 6.4.2 some of the most important research works on TRS's, highlighting their strong and weak points in each case.

6.4.1 Commercial TRS's

One of the early on-line social networks were the *on-line forums*. These systems allow users to hold conversations by posting messages to the network. These messages are usually grouped in threads of discussion. A phenomenon

that became too common in on-line forums is the appearance of *trolls*, that are users who comment contents with the aim of focusing the attention on themselves diverting the topic of discussion, or simply to cause an argument. This is obviously a trust and reputation problem. The mechanism to deal with it was the creation of the *moderators*. Forums usually have this special group of users with the ability (and responsibility) of banning those users that they consider malicious or trolls. The main benefit of this system is the ability of determining whether a user is a troll or not regarding all their actions, including their comments in the network.

This mechanism is being used currently not only in on-line forums, but also in most recent systems such as social news sites. Slashdot.org [7] is a social network funded in 1997, focused on the publication and discussion of news mainly related to technology. The system provides the users with the ability of establishing among them two types of relationships: friend (positive link) or foe (negative link). In the beginning, Slashdot established a group of 25 moderators. This amount was increased to 400 due to the growth in the number of users. Nowadays, the moderator team is automatically selected. The Slashdot TRS consists in three layers: the first one is for rating the content of the network (comments and articles), while the second one moderates the raters. In addition, Slashdot staff members are able to moderate any element in the system (comments, articles and participants).

The inclusion of moderators in these systems has two main disadvantages: scalability and subjectivity. This supervised system can be a reliable mechanism in networks with a medium-small number of users, but it is not scalable to social networks with a high amount of members and contents. Some social networks that have adopted this kind of TRS try to overcome this problem by delegating some of the moderation task to the users. They are provided with a mechanism to send notifications to the moderators whenever they detect an abusive content or behavior in the network. This feature can help the moderators, but it also offers options to malicious users for cheating the TRS in other ways. Concerning the subjectivity problem, the decisions about whether a user is a troll or not, or whether a comment is a controversial contribution to a discussion or just trolling, depends on the judgment of the moderators.

Other social networks provide all their users with the ability of giving their opinions about other users, or even the contents generated by them. These voting systems are decentralized, so they avoid the subjectivity problem by delegating the decision to all the users of the network. It also overcomes the scalability issue because there is not an authority (or a group of them) that

[7] `slashdot.org`

has to make all the decisions. There exist many different techniques that process this information together with the topology of the network of relationships formed by the users in order to propagate their opinions through the system with the aim of computing a trust score for each user. For example, in eBay [8] the users are encouraged to give their opinions about each commercial transaction that they perform, using this information to obtain the trust scores of users.

The combination of all these mechanisms is frequent. In Meneame [9], a very popular social news site for Spanish-speakers (similar to Digg), is implemented a hybrid TRS with different levels of moderators and also positive and negative votes between the users. The system computes a score for each user, namely *karma*, taking into account many parameters about the activity and behavior of each user in the system. In this way, the users are rated in terms of the number of positive and negative votes to their comments and the news uploaded to the system, and also regarding to the ratings that they give to other users. The score of all the users is updated every 24 hours. A more comprehensive explanation of this metric can be found on `meneame.wikispaces.com/Karma` (in Spanish). In addition, the users in the moderator group can penalize other users by manually decreasing their *karma*, and also edit their news and comments, etc.

A new feature is presented in the Advogato [10] trust and reputation system (Levien, 2002; Levien and Aiken, 1998): the use of a small group of designated *seeds* or *authoritative nodes* in order to compute the trustworthiness of the rest of the users in the system from the point of view of these completely trustworthy *seeds*. Advogato is a social network that constitutes a community discussion board for free software developers. It was created with the aim of serving as testbed for research works on TRS. The system allows members to "certify" other users, forming a network of trust. The computation of each user rating is performed by a graph-based algorithm, similar to PageRank, that propagates the information about the relationships between the users and their relation with the *authoritative nodes*. Once the system has found out a bad user, b, the TRS will also search for all users that certifies b. The trust score is computed for all the users in the system, so it is a global trust algorithm.

[8] `www.ebay.com`
[9] `www.meneame.net`
[10] `www.advogato.org`

6.4.2 Research on TRS

Focusing on some of the most relevant research works on trust and reputation in on-line social networks, we find the EigenTrust algorithm (Kamvar et al., 2003b), that aims to reduce the number of inauthentic file downloads in a P2P network. In the EigenTrust algorithm, each user calculates the local trust value for all peers voting to him. The global trust value is obtained by aggregating the normalized local trust values with respect to a peer. EigenTrust claims to take into account both positive and negative feedback from the users, but it actually assigns a trust score of 0 to every user whose local trust score is negative. So the algorithm only works with graphs whose edges have always positive weights.

TidalTrust (Golbeck, 2005) is proposed to measure the trust between two users in a social network. It processes the trust graph, constructed by the relations between users. The algorithm computes the trust score for each pair of users, $t_{u,v}$, relying on the direct experience of all the nodes that constitute the shortest path from u to v in the network of trust. The algorithm follows the Equation 6.1, to recursively compute the trust between u and v in terms of $t_{w,v}$, where w is a node of the path from u to v.

$$t_{u,v} = \frac{\sum_{w \in T_u \cap t_{u,w} > \epsilon} t_{u,w} t_{w,v}}{\sum_{w \in T_u \cap t_{u,w} > \epsilon} t_{u,w}} \tag{6.1}$$

where ϵ is the threshold that limits the strength of the paths to be considered. The strength of a path $\{a, b, c\}$ is computed as the minimum between the rating between a and b and the rating between b and c. In the cited work they show the performance of the algorithm using a dataset extracted from FilmTrust [11], a social network formed by a system for user-generated movies reviews and a recommender system. The recommendation of movies is based on the combination of the user's trust network and the movie ratings created by trusted friends.

PowerTrust (Zhou and Hwang, 2007) is a TRS that aggregates the positive and negative opinions between the users into the *local trust scores*, similarly to EigenTrust. These scores are propagated through the network by a random-walk algorithm in order to obtain the *global trust scores* for each user in the network. The novelty in this system is that it establishes a method intended to select a set of *power* peers (most reputable users in a given instant). The opinions from these users will be propagated through the network with more strength than the opinions from the rest of users. This method has the advantage of automatize this process, but it presents a major disadvantage:

[11] `trust.mindswap.org/FilmTrust`

since it is an unsupervised process, it is susceptible to being used by malicious users for their benefits, so it constitutes a potential weak point in the TRS.

Furthermore, PowerTrust implements a *look-ahead* random-walk strategy that aims to accelerate the convergence of the random walk by aggregating into each node not only its own local scores, but also its neighbors first hand information.

The concept of *transitivity of distrust* is introduced in (Guha et al., 2004), representing the way in which negative opinions of users can be spread over the social network. There are different assumptions that can be adopted when we try to estimate the trustworthiness of a node in a network, depending on the transitivity of distrust.

- **Multiplicative distrust** follows the assumption of *"The enemy of my enemy is my friend"*, it implies that if user a distrusts b, and b distrusts c, then a trusts c.
- In an **additive distrust model**, given the previous example, a should strongly distrust c, because c is not trusted by b. In other words: *"Don't trust someone not trusted by someone you don't trust"*.
- **Neutral distrust** can be stated as: *"Don't take into account your enemies' opinions"*. It implies that if a distrust b, then we can not make any inference for neighbors of node b.

The transitivity models are graphically represented in Figure 6.4. The red lines represent distrust between the related nodes, the green edges represent trust, and the black ones are undetermined relations.

After the research work in (Guha et al., 2004), some approaches have been proposed as extensions of already existing solutions intended to include in those systems the negative opinions and the transitivity of distrust. This is the case of the proposal in (Donato and Stefano Leonardi, 2008), that extends the EigenTrust algorithm with some ideas intended to include into the algorithm the negative opinions of the users in a network. Another work that processes a social network with positive and negative opinions is presented in (Kunegis et al., 2009). They study the trust and reputation mechanism in Slashdot.org, where users can establish relations of friend or foe with other users in the network. Given this network as dataset, some centrality measures and metrics are compared with the proposals of the paper: *Signed Spectral Ranking* (SR) and *Negative Ranking* (NR), which takes into account positive and negative opinions in the social network. The first one is a popularity measure consisting in an extension of PageRank algorithm that includes in its computation the negative edges of the graph. Negative Ranking includes

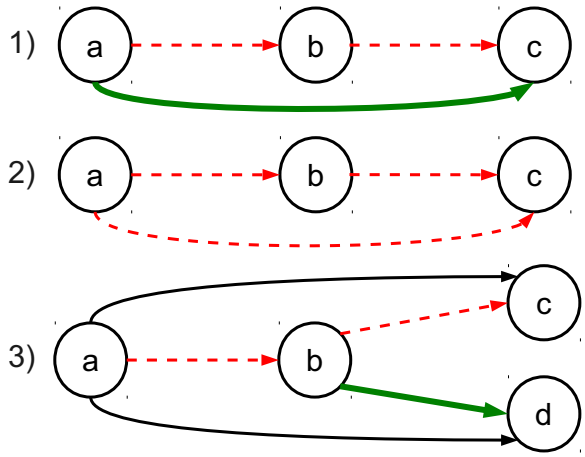


Figure 6.4 – Transitivity models: (1) Multiplicative distrust. (2) Additive distrust. (3) Neutral distrust.

both PageRank and SR in the computation. They also use in their experiments the heuristic *Fans Minus Freaks*. It is a simple metric that relies only in the direct experience of the users, taking into consideration the detractors and followers of each one to compute its score.

It is also interesting the inclusion of time in the computation of trust. In (Agudo et al., 2009), they introduce an evolutionary trust and distrust model, in which time is taken into account to obtain trust values. When an user (trustor) trusts another one (trustee) in an instant, a trust statement is built as (t, r, t_m), where t is the trust value, t_m is the time-stamp of the statement, and r represents the reliability of the trust statement. The reliability is the level of confidence that trust or distrust values will stay stable in the future. This work measures the influence of the time in the trust values. They propose a trust consensus mechanism in order to obtain a trust decision about one user. This algorithm takes as input multiple trust statements, and calculates the average of the statements about a user.

6.5 Summary

Trust and Reputation Systems constitute an essential part of many social networks due to the great expansion of these on-line communities in the past few years. As a consequence of this growth, some users try to disturb the normal atmosphere of these communities, or even to take advantage of them in order to obtain some kind of benefits. The most common tactic used by these malicious users is to gain high reputation in the network in order to

make their behavior more damaging to the system. Therefore, the concept of trust is a key point in the performance of on-line systems such as on-line marketplaces, review aggregators, social news sites, forums, etc.

In this chapter we present a study of the inherent vulnerabilities of trust and reputation in social networks. The background on TRS is rounded out with an analysis of the malicious behaviors that can lead to a distortion in the reputation of the users in the network, decreasing or increasing the trustworthiness of some users, harming or benefiting (depending on the case) some of them.

Furthermore, we have discussed a case study of one of the most popular social news sites: Digg.com, a social network based on the publication of news sent by their users, who can also comment and rate the contents of the site. In order to illustrate the vulnerabilities and problems of a TRS we comment some incidents related to the trust and reputation caused by some users in this site and the countermeasures applied in order to avoid the effects of these attacks. This situation is a good example of some of the problems that a TRS must face in a real environment.

Finally, we have reviewed some of the most relevant research works on this field, in addition to the trust and reputation systems implemented in some quite popular (at present) on-line social networks.

Chapter 7

A Trust and Reputation System Using Graph-based Ranking Algorithms

Keep your friends close
but your enemies closer
Michael Corleone
The Godfather Part II, 1974

7.1 Introduction

The relevance of Trust and Reputation Systems (TRS) has grown in the past few years in parallel to the huge expansion of Social Networks. The problem of managing the trustworthiness of the users of an on-line community is not new, and it has been faced with diverse methods depending on the context and the aim of the network where it was applied. For example, in on-line forums and social news sites it is not rare to encounter a *troll*, that is a user who comments contents with the aim of focusing the attention on himself diverting the topic of discussion, or simply to cause an argument. This can be just a problem of annoyance, or it can entail much more serious consequences for both, the social network and its users. A good example can be found in on-line auctions and on-line marketplaces, such as eBay [1], where the reputation of users is a crucial feature in order to provide a security mechanism for the commercial transactions to the potential buyers. The illicit manipulation of

[1] `ebay.com`

the reputation in these systems can be really harming due to the immediate benefit that can be obtained by a dishonest user, and the subsequent damage suffered by the user being subjected of the misbehavior. So, determining the trust and reputation of the users is an important task to be addressed by many social networks, since the concept of reputation is a key feature for those on-line systems whose functioning or even their business model strongly rely on the trust among their users.

Social networks often establish their own systems to ensure the trustworthiness of their users, or to inform to their users about the trustworthiness of the rest of them. This is the aim of *trust and reputation systems*. We have reviewed some of the most important ones in Chapter 6. Furthermore, we have also discussed some weaknesses and threats for these systems. In our study, we have confirmed that there are not many works that process a social network with positive and negative opinions between its users. They usually consider the reputation of a user to be in a positive range from 0 to $x > 0$, being 0 the worst reputation score.

The problem in these systems is the non-existence of a neutral value for the reputation, because a positive value always implies, subconsciously, a positive connotation, and analogously for negative values. For example, given a range of $[0, 5]$ for the reputation of the users in a social-network, a user with a reputation of 0 is perceived as a bad user, while a user with a reputation of 5 is considered as totally trustworthy. The problem lies in the values in the middle (in our example, 2 or 3), because they should mean that the system is not able to determine whether the user is trustworthy or not, or that a user cannot be clearly defined as trustworthy or untrustworthy. On the contrary, these scores can be perceived, unconsciously, as a positive rating due to their positive value.

On the other hand, when it comes to give your opinion about a user who is neither bad nor good from your point of view, the fact of assigning him a score of 2 or 3 would look like you are giving credit to someone who does not deserve it. Thus it is more common to resolve the situation by assigning a 0 anytime that you do not have anything good to say about someone, although nothing bad can be said either. But this rating actually means that you do not trust that user, so it is a negative vote for him in the system. This is why the fact of considering both positive and negative votes in a TRS can be helpful not only for the system itself, but also for the users who can provide their feedback in a more accurate and intuitive way.

In this chapter we present a novel TRS based on the ideas behind PolarityRank (Cruz et al., 2011b). Our approach, PolarityTrust, consists in a graph-based algorithm that computes a ranking of users according to their trustworthiness. We define a general method that takes advantage of the

positive and negative opinions in a social network in order to build the ranking of trust. Our proposal is intended to demote in the ranking those users who present a dishonest behavior in the system and, at the same time, to avoid the negative effects that the actions of these malicious users can cause in the reputation of the rest of users in the network.

The rest of the chapter is organized as follows. In Section 7.2 we present PolarityTrust, our trust and reputation algorithm that adapts PolarityRank in order to deal with the trust and reputation task, facing some common vulnerabilities in TRS's, such as the orchestrated attacks and the dishonest votes. The design of the experiments carried out to test our approaches are shown in Section 7.3. The results of all the experiments performed with the different techniques and datasets are discussed in Section 7.4. Finally, in Section 7.5 we do a summary of the main points of this chapter.

7.2 Our proposal: PolarityTrust

In this section, we present PolarityTrust (Ortega et al., 2011a) our proposal for the computation of the trust and reputation of the users in a social network. It is based on similar ideas as PolarityRank (Cruz et al., 2011a,b) discussed in depth in Section 3.3. It is worth to remember that PolarityRank is a graph-based ranking algorithm which can process a graph with positive and negative weights in its edges. For example, PolarityRank has been proved to be a reliable method to compute the semantic orientations of opinionated expressions in user-generated reviews of products, by the propagation of the information of some known expressions through a graph built from the conjunctive relations between the expressions in a text.

Concerning to our proposal, PolarityTrust propagates the information of some relevant users of a social network through a weighted graph with positive and negative edges, in order to determine the trustworthiness of the rest of users in the network. In this case, the graph to be processed is built by taking the users of the network as the set of nodes. The edges represent the relations between those users, expressed in terms of opinions among each other or different kinds of ratings among them or even the different types of relations, depending on the possibilities that the diverse social networks provide their users with.

Given a graph with those characteristics, PolarityTrust computes a ranking algorithm similar to PolarityRank in order to obtain two scores for each user: the first one represents the positive reputation of the user (PT^+), and the other one represents the negative reputation (PT^-), according to the relations among all the users in the network. Finally, we build the ranking

of users by ordering them by their $Trust$ score, computed for each user, i, taking into account both $PT^+(v_i)$ and $PT^-(v_i)$ as follows:

$$Trust(i) = \frac{PT^+(v_i) - PT^-(v_i)}{PT^+(v_i) + PT^-(v_i)} \tag{7.1}$$

PolarityTrust implements as trust and distrust transitivity model the multiplicative distrust assumption: *"The enemy of my enemy is my friend"*, graphically represented in Figure 6.4(1). The basic equations are formulated as follows:

$$\begin{aligned} PT^+(v_i) &= (1-d)e_i^+ + d(\\ &+ \sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^+(v_j) + \\ &+ \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^-(v_j)) \end{aligned} \tag{7.2}$$

$$\begin{aligned} PT^-(v_i) &= (1-d)e_i^- + d(\\ &\sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^-(v_j) + \\ &+ \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^+(v_j)) \end{aligned} \tag{7.3}$$

So a node i gains high trust score in two cases, when:

1. A highly trusted node votes positively for i, because the high trustworthiness score of the voting node increases the trust score of i (first sum in Equation 7.2). In other words, node i is related to a trustworthy user, so it must be considered trustworthy as well.

2. A highly distrusted node votes negatively for i, because the inverse relation between both nodes makes the high untrustworthiness of the voting node rise the trustworthiness of i (second sum in Equation 7.2). This situation can be caused by the fact that node i has an opposite position (opinion) of an untrustworthy user, so it must be taken as a trustworthy user.

Analogously, a node i gains high distrust score if:

1. A highly trusted node votes negatively for i, because the inverse relation between the nodes transforms the high trust score of the voting node into a high distrust score for the target. See the first sum in Equation 7.3.

2. A highly distrusted node votes positively for i, because node i is considered to be closely related to an untrustworthy user. See the second sum in Equation 7.3.

Furthermore, PolarityTrust includes two mechanisms intended to specifically deal with some difficulties in the task of trust and reputation in social networks, such as the dishonest voting attacks. We discuss them in depth later on.

The rest of the section is organized as follows. In Section 7.2.1 we discuss the inclusion of a bias in the ranking algorithm with the use of sources of trust and distrust. Then in sections 7.2.2 and 7.2.3 are explained the two propagation mechanisms implemented in PolarityTrust in order to deal with the diverse threats for a TRS. Finally, the complete formulation of PolarityTrust is stated in Section 7.2.4, including all the aspects explained in previous sections.

7.2.1 Sources of Trust and Distrust

As mentioned in Chapter 6, some social networks establish a group of authorities or moderators (Slashdot, Kuro5in[2], most of the on-line forums). The opinions of these users are more relevant than the opinions from the rest of users in the network, and they are allowed even to judge the behavior of other users and to decide the actions to be conducted in consequence. In our system, we can take into account this fact by considering a special group of nodes as *sources of trust*. The intuition behind this concept is that the opinions from these users must be propagated over the network with more strength than the opinions from the rest of the users. In this way, their opinions can be more influential in the propagation algorithm than the opinions from the rest of users in the network.

This idea is included in our algorithm by setting up one of the parameters in the original formulation of PageRank: the vector e, mentioned in Section 2.2.1. This vector is intended to insert a bias in the random-walk algorithm, giving more weight to some nodes over the rest, in other words: if

[2] `www.kuro5hin.org/`

we consider a surfer that randomly visits the nodes of the graph depending on their number of in-links and the relevance of their neighbors, the vector e is intended to artificially increase the probability of a node to be visited by the random surfer.

In terms of our proposal, an adaptation of the idea behind the vector e in PageRank is included in PolarityTrust in order to characterize some users who are considered as totally trustworthy a priori (before the computation of the algorithm), the so called *sources of trust*. Thus, we propose the use of the vector e^+, which contains a score for each user that is considered a *source of trust*. The vector is initialized as follows:

$$e_i^+ = \begin{cases} \frac{1}{|Sources^+|} & i \in Sources^+ \\ 0 & otherwise \end{cases}$$

where $Sources^+$ is the set of users taken as sources of trust. This intuition can be extended in order to obtain a set of *sources of distrust*, who can provoke in the algorithm the opposite effect of the sources of trust. In this way, the reputation of the users who are positively linked from a source of distrust, u, must be decreased, because the distrust score of u is propagated to them. Analogously to the previous approach, the users that are taken as sources of distrust, have an *a priori* score in the vector e^-. Given $Sources^-$, a set of users who are previously tagged as untrustworthy, the initialization of vector e^- is similar to the vector e^-:

$$e_i^- = \begin{cases} \frac{1}{|Sources^-|} & i \in Sources^- \\ 0 & otherwise \end{cases}$$

7.2.2 Non-Negative Propagation (PT_{NN})

It has been shown in Section 6.2.2 that malicious users have many ways to take advantage of the weaknesses of trust and reputation algorithms. In order to avoid the influence of bad nodes in the final ranking, we integrate in the PolarityTrust algorithm the capability of deciding whether the opinion of the users will be taken into account or not. Thus we can minimize the influence of malicious peers in the ranking by allowing only some users to propagate their opinions over the network.

In this sense, we adopt an hybrid model for the transitivity of distrust, graphically represented in Figure 7.1. It consists in avoiding the propagation of negative opinions from bad users, so they cannot harm the trust scores of good users. Since we take into account the trust scores of the nodes in order to decide whether a user is good or bad, the consideration of a "bad user"

is dynamic because these scores are changing during the computation of the propagation algorithm.

Non-Negative Propagation can help the system to deal with possible attacks based on the use of negative votes, such as in the *"Camouflage behind judgments"* threat model. As it is shown in the Figure 7.1, if user a distrusts user b, and b trusts d, then a distrusts d (multiplicative distrust). However, if b distrusts c, a does not take this opinion into account (neutral distrust), because this decision would be inferred by following a negative opinion from a negative user.

The effects of the positive opinions from bad users are already included in the basic mechanism of PolarityTrust, taking this information into account depending on the target of the links. In this way, Non-Negative propagation approach aims to protect the global ranking from certain actions of malicious users, because their negative opinions will not be propagated.

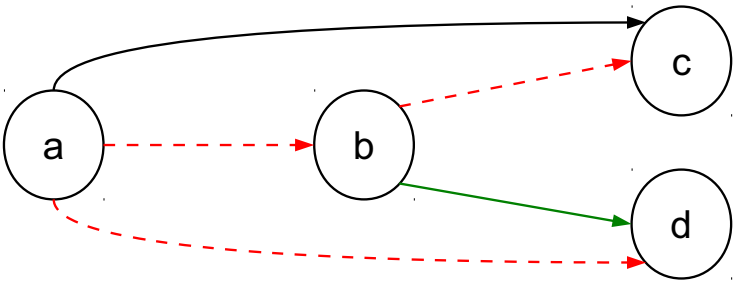


Figure 7.1 – Transitivity model adopted by the Non-Negative Propagation approach. Red dashed lines correspond to negative opinions, while green lines are positive votes. The black line represents a relation whose polarity cannot be inferred from the information provided by user b, in accordance with the transitivity model considered.

Given the scores PT^+ and PT^-, we can dynamically classify a user, i, as good or bad by checking the sign of $Trust(i)$, as follows:

$$Sign(i) = \begin{cases} -1 & Trust(i) < 0 \\ 1 & otherwise \end{cases}$$

So, the basic PolarityRank with the addition of Non-Negative propagation is computed as it is shown in the Equations (7.4) and (7.5).

$$
\begin{aligned}
PT^{+}_{NN}(v_i) &= (1-d)e^{+}_{i} + d(\\
&+ \sum_{j \in In^{+}(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^{+}_{NN}(v_j) + \\
&+ \sum_{j \in In^{-}(v_i), Sign(j)>0} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^{-}_{NN}(v_j))
\end{aligned}
\tag{7.4}
$$

$$
\begin{aligned}
PT^{-}_{NN}(v_i) &= (1-d)e^{-}_{i} + d(\\
&\sum_{j \in In^{+}(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^{-}_{NN}(v_j) + \\
&+ \sum_{j \in In^{-}(v_i), Sign(j)>0} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^{+}_{NN}(v_j))
\end{aligned}
\tag{7.5}
$$

This mechanism allows PolarityTrust to take a node as good or bad depending on the scores obtained in each iteration of the algorithm. It means that, in a given iteration t, we can take user i as a bad user (and consequently do not propagate its negative opinions), and the next iteration, $t+1$, the same node could be taken as a good user due to the scores of its neighbors in that iteration.

7.2.3 Action-Reaction Propagation (PT_{AR})

The main aspect of the behavior of malicious users is the set of relations that they establish intended to gain some kind of benefits in terms of good reputation in the network. Most of the attacks that these users perform against a TRS consist in providing incoherent votes to other users in the network. In this section we explain the last extension of our TRS, that addresses the dishonest voting problem. This vulnerability is based on the concept of *malicious spies* that are users who seem to be good but only assign positive trust values to malicious peers. This kind of attacks can be blocked by studying the coherence or incoherence of the opinions in the network. This problem is called *Dishonesty* in (Donato and Stefano Leonardi,

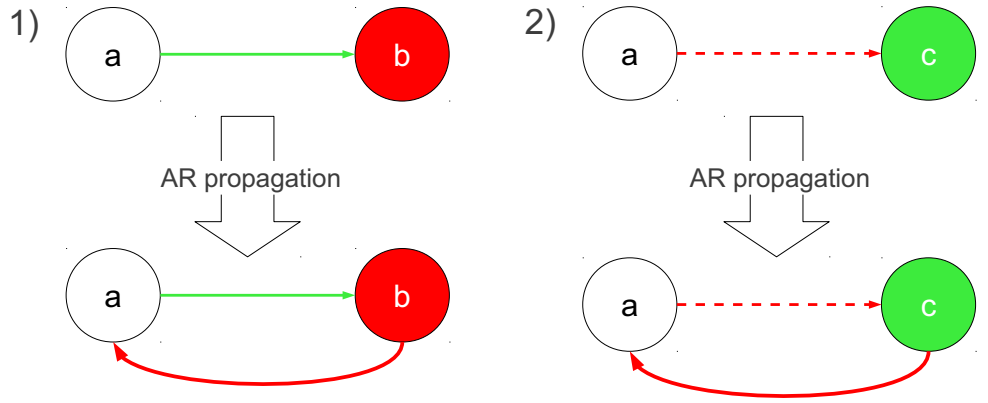


Figure 7.2 – Action-Reaction against incoherent judgments: node b is a malicious user, node c is a positive one, and node a is a dishonest user voting them. Red dashed lines corresponds to negative opinions, while green lines are positive votes. AR Propagation virtually creates a negative vote (red solid lines) from nodes b and c to node a, in order to penalize its incoherent judgments.

2008), defining as *Positive Dishonesty* the situation in which a user assesses positively malicious peers, and analogously for *Negative Dishonesty*.

Action-Reaction Propagation is the method that we propose to deal with *Dishonesty*. It is graphically explained in Figure 7.2. It consists in penalizing those users who have incoherent judgments, or *dishonest users*, by the dynamic inclusion of virtual negative votes against incoherent opinions. In order to judge a vote as dishonest or not, we define the *polarity* of the edge from user i to user j as:

$$Polarity(i,j) = \begin{cases} -1 & p_{ij} < 0 \\ 1 & otherwise \end{cases}$$

where p_{ij} is the weight of the edge from i to j. The penalization depends on the level of dishonesty of each user, in other words, the total number of incoherent judgments. The penalization score, $AR(i)$, is computed as follows:

$$AR(i) = \frac{\sum_{j \in Out(v_i), Sign(j) \neq Polarity(i,j)} Trust(j)}{\sum_{k \in Out(v_i)} Trust(k)} \tag{7.6}$$

where $Sign(j) \neq Polarity(i,j)$ is true if user i assigns a positive vote to a negative node, or if it assigns a negative vote to a positive node. This penalization only affects the negative score of each node. Thus the score PT^- is now obtained as shown in Equation (7.7).

$$
\begin{aligned}
PT^{-}_{AR}(v_i) &= (1-d)e^{-}_i + d(\\
&\quad \sum_{j \in In^{+}(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^{-}_{AR}(v_j) + \\
&+ \quad \sum_{j \in In^{-}(v_i)} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^{+}_{AR}(v_j)) + \\
&+ \quad \frac{AR(i)}{\sum_{k \in V} AR(k)}
\end{aligned}
\tag{7.7}
$$

Note that the AR model does not reward the coherent votes, but penalizes the incoherent ones. This observation is important due to the possibility of an attack consisting in a variation of threat model E, in which malicious users vote negatively for good users. If the coherent votes are rewarded, a malicious user could gain high reputation by voting positively for good users. This approach also avoids this possibility by only penalizing the incoherent votes.

7.2.4 PolarityTrust: Complete Formulation

Our proposal, PolarityTrust, combines both extensions, Non-Negative Propagation and Action-Reaction Propagation, in order to take advantage of their ideas. In this way, the opinions of bad users are not propagated to their neighbors, and all the nodes with a dishonest behavior are penalized in the ranking of trust. This combined model follows the Equations (7.8) and (7.9).

$$
\begin{aligned}
PT^{+}_{ALL}(v_i) &= (1-d)e^{+}_i + d(\\
&+ \quad \sum_{j \in In^{+}(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^{+}_{ALL}(v_j) + \\
&+ \quad \sum_{j \in In^{-}(v_i), Sign(j)>0} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \; PT^{-}_{ALL}(v_j))
\end{aligned}
\tag{7.8}
$$

$$
\begin{aligned}
PT^{-}_{ALL}(v_i) &= (1-d)e^{-}_{i} + d(\\
&+ \sum_{j \in In^{+}(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \, PT^{-}_{ALL}(v_j) + \\
&+ \sum_{j \in In^{-}(v_i), Sign(j)>0} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} \, PT^{+}_{ALL}(v_j)) + \\
&+ \frac{AR(i)}{\sum_{k \in V} AR(k)}
\end{aligned}
\tag{7.9}
$$

The final score for each user, $Trust(i)$, is obtained as follows:

$$
Trust(i) = \frac{PT^{+}_{ALL}(v_i) - PT^{-}_{ALL}(v_i)}{PT^{+}_{ALL}(v_i) + PT^{-}_{ALL}(v_i)}
$$

7.3 Experimental settings

The experiments are designed to evaluate the reliability of PolarityTrust in challenging situations that can show the performance of the technique in carrying out the task of computing the trustworthiness of the users in a social network. In the reminder of this section, we present the datasets used in this work in Section 7.3.1. They consist in a set of randomly generated graphs (see Section 2.3) and a real-world dataset extracted from Slashdot.org, a very popular social news site. In Section 7.3.2 we explain additional methods used in the experiments to compare the results. Finally, in Section 7.3.3 we briefly discuss the metrics used for the evaluation of the experimental results.

7.3.1 Datasets

The evaluation of our proposal requires the use of large real-world networks as datasets, in order to prove its usefulness in a real environment. Thus, we have used a set of randomly generated graphs, in addition to a real-world dataset extracted from the Slashdot social network. All these datasets have been augmented with a set of malicious users who perform some kind of attacks against the system, in such way that the reliability of PolarityTrust can be evaluated in challenging situations.

The randomly generated graphs, simulating the topology of real-world networks, have been created using one of the methods presented in Section 2.3: the Barabási-Albert method (Barabási and Albert, 1999). On the

other hand, the attacks performed by "our" malicious users are those explained in Section 6.2.2. A separate component has been implemented in order to reproduce the dishonest behaviors of malicious users, adding a group of nodes and a set of edges that reproduce possible attacks against a TRS.

Dishonest Behaviors Generator

Since we need to prove the robustness of our proposal in challenging situations, the datasets used in our experiments must include users with dishonest behaviors. With this aim, we implement a mechanism to include threatening behaviors in a given graph representing a social network.

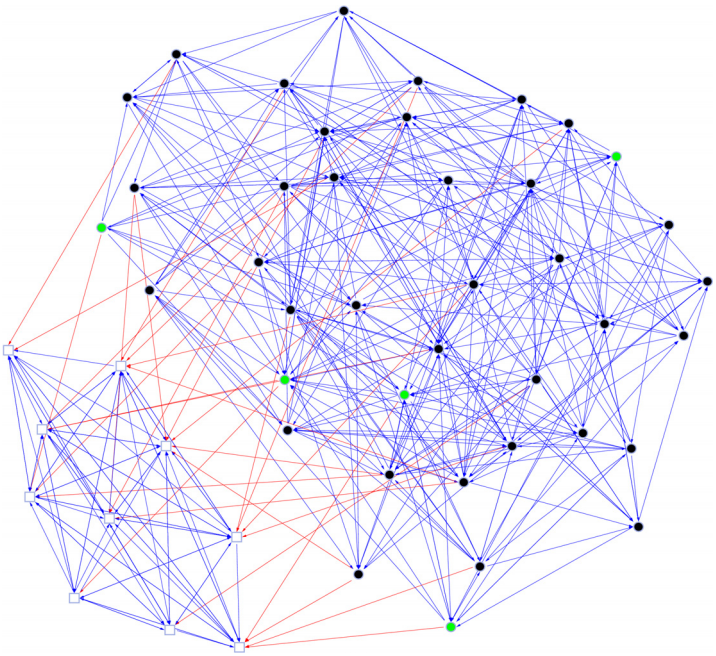


Figure 7.3 – Example of the random graph generated by the Barabási-Albert Model of Figure 2.3-(1) with negative nodes. White nodes represent malicious users, and the blue ones are good users. The green nodes represent the users taken as sources of trust.

Given the number of bad nodes that must be included in the dataset and the set of attack models that they will perform against the system, the component for the generation of dishonest behaviors adds the bad nodes in the random graph and creates the necessary edges to perform the required attack models. It works by iterating over the nodes in the graph and randomly choosing the targets of the edges, given a source node. The method also takes as input the type of attacks that will be simulated, and their intensities. In

other words, these parameters establish the probability of creating an edge intended to perform the corresponding attack. In addition, it also generates negative edges from good to bad users with a certain probability, simulating the normal behavior of these users. In Figure 7.3 is shown the random network of Figure 2.3-(1) with the addition of malicious users performing an orchestrated attack.

The pseudo-code of the algorithm for the random generation of the attacks against a social network is shown in Algorithm 1. According to this specification, it is the responsibility of the method *createAttackingEdge* (line 7) to analyze the type of *Attacks* that are required and randomly create an edge from node n to node *target* that helps to perform those attacks. As mentioned before, the algorithm takes as input a graph, G, the number of malicious users who will be added, b, and the set of attacks that they will trigger against the system, *Attacks*. The malicious spies attack is specified separately, by initializing the parameter *sp* with the number of spies that will be included in the dataset.

Finally, there is a set of additional parameters intended to change the density of the edges in the graph. Parameter d_{max} defines the maximum degree of the nodes in the graph, in the experiments this parameter has been set to 20. Parameter *bias* is used for the introduction of a slant towards creating a higher amount of edges between nodes of the same type (good or bad), specifically the 20% of the relations in the generated datasets are established between nodes of different types. The *it* parameter forces the algorithm to repeat the process that number of times, the value of this parameter has been set to 5 for all the datasets used in our experiments.

Slashdot Zoo dataset

In addition to randomly generated datasets, we have tested our proposal against a real-world dataset, namely *Slashdot Zoo*, presented in (Kunegis et al., 2009). It is a crawling of the Slashdot.org network that is a social network funded in 1997, focused on the publication and discussion of news mainly related to technology. Slashdot.org system provides the users with the ability of establishing among them two types of relationships: friend (positive link) or foe (negative link). In the beginning, Slashdot established a group of 25 moderators. This amount was increased to 400 due to the growth in the number of users. Nowadays, the moderator team is automatically selected. The Slashdot TRS consists in three layers: the first one is for rating the content of the network (comments and articles), while the second one moderates the raters. In addition, Slashdot staff members are able to moderate any element in the system (comments, articles and participants).

Data: it = iterations;
b = number of bad nodes;
sp = spies;
d_{max} = max. degree of nodes;
bias = prob. of choosing a node of the same type;
Attacks = set of attack models;
G = graph of good users;
Result: G = graph of good and bad users

```
1  generateBadNodes(G, b, sp);
2  while numberOfIterations < it do
3      forall the n ∈ G do
           // Random number of neighbors of opposite type to n
4          noPairNeighbors = random(0, ((1−bias)·d_max)/it);
           // Selecting nodes of the opposite type to n
5          nopairs = chooseNoPairs(n, noPairNeighbors);
6          forall the target ∈ nopairs do
7              createAttackingEdge(n, target, Attacks);
8          end
           // Checking if an orchestrated attack is required,
              and creating the corresponding edges
9          if n is a bad node & Attacks include orchestrated attacks then
               // Random number of neighbors of same type as n
10             pairNeighbors = random(0, (bias·d_max)/it);
               // Selecting nodes of the same type as n
11             pairs = choosePairs(n, pairNeighbors);
12             forall the target ∈ pairs do
13                 createPositiveEdge(n, target, Attacks);
14             end
15         end
16     end
17 end
```

Algorithm 1: Method that generates a random graph under a combination of attacks performed by b dishonest users and sp malicious spies.

The dataset crawled in (Kunegis et al., 2009) is formed by the users in the system and their friend-or-foe relations. It contains about 71500 users and more than $510K$ edges, being 24% negative links. The gold standard is given by a special user, called *No_More_Trolls*, who has all the known trolls of Slashdot in its list of foes. Since we want to follow the same experimental settings of (Kunegis et al., 2009), we have considered the same set of 96 trolls.

7.3.2 Baselines

In this section we present the techniques taken as baselines in the experiments (EigenTrust and Fans Minus Freaks) as well as two more sophisticated methods (Signed Spectral Ranking and Negative Ranking) that, like our proposal, use positive and negative links to compute trust values.

The first baseline method is the EigenTrust algorithm (Kamvar et al., 2003b). It aims to reduce the number of inauthentic file downloads in a P2P network. In the EigenTrust algorithm, each user calculates the local trust value for all peers voting to him. The global trust value is obtained by aggregating the normalized local trust values with respect to a peer. Formally, given C, a matrix where c_{ij} represents the opinion of i about j (local trust value), the EigenTrust algorithm computes the global trust values as:

$$\bar{t}_i = C^T \bar{c}_{ij}$$

where $\bar{t}_i$ is the vector of local trust values of i for each node in the network. Repeating this process, $\bar{t}_i$ will converge to a stable value, $\bar{t}$, that is the vector containing the EigenTrust values for each node. This vector is the *left principal eigenvector* of the matrix C.

The second baseline is the heuristic *Fans Minus Freaks*. It is a simple metric that relies only in the direct experience of the users, taking into consideration the detractors and followers of each one to compute its score. The trust score of node v_i is the difference between the number of positive and negative links pointing to i, obtained as follows:

$$FmF(v_i) = \mid In^+(v_i) \mid - \mid In^-(v_i) \mid$$

Finally, we have implemented the two methods proposed in (Kunegis et al., 2009), *Signed Spectral Ranking* (SR) and *Negative Ranking* (NR). The first one is a popularity measure consisting in applying PageRank algorithm directly to the graph, including negative edges. They consider the positive edges with a weight of 1, and the negative ones -1. Negative Ranking includes both PageRank and SR in the computation as it is defined below:

$$NR = SR - \beta \cdot PR$$

where $\beta \geq 0$ determines the influence of PageRank on the final ranking. In (Kunegis et al., 2009), the best results are achieved by setting $\beta = 1$, so it is done in our experiments.

Unlikely this method, in our proposal both positive and negative links are propagated in such way that they influence both positive and negative scores of the users, being able to implement the transitivity of trust and distrust in a social network.

7.3.3 Evaluation metrics

Since the aim of TRS's is to demote the bad users in the ranking and to promote the good ones, we use the *number of inversions* as one of the evaluation metrics. In other words, we evaluate the performance of the techniques in terms of the number of bad users that appear in the positions of the ranking corresponding to good users. The lower the metric value, the better is the performance of the technique. This metric is normalized by the total number of bad users in order to obtain values from 0 to 1. The error rate of each technique is computed as follows:

$$ErrorRate = \frac{NumberOfInversions}{TotalNumberOfBadUsers}$$

On the other hand, we have included in the evaluation of the experiments the standard $nDCG$ (Normalized Discounted Cumulative Gain) (Najork, 2007), already explained in Section 5.3.3. In this way, we do not only evaluate the fact of avoiding bad users in top positions, but also the specific ranking achieved by these malicious users. $nDCG$ is obtained as follows:

$$nDCG = \frac{relevance_1 + \sum_{i=2}^{N} \frac{relevance_i}{log_2 i}}{IDCG}$$

where N is the total number of users in the network. The *relevance* is 1 for all the good users, and 0 otherwise. Finally, $IDCG$ is the Ideal Discounted Cumulative Gain, that is the best DCG that can be obtained for the given dataset. In our case, this metric evaluates the appearance of bad users in the ranking, penalizing the results where these users reach high positions.

7.4 Experimental results

In this section we present the results of the experiments performed following the settings explained above. In Table 7.1 we can see a summary with the abbreviations used in showing the results of the experiments.

Abbreviation	Concept
A	Individual Malicious Peers
B	Malicious Collectives
C	Camouflage behind good transactions
D	Malicious spies
E	Camouflage behind judgments

Table 7.1 – Abbreviations for the threat models used in the rest of tables and figures of the experiments.

First, in Section 7.4.1 we test our proposal against an incremental combination of the basic attacks reviewed in Section 6.2.2. We also show the performance of our approaches with a real-world social network using the Slashdot Zoo dataset in Section 7.4.2. Then we test our proposal in Section 7.4.3 with a combination of real and synthetic data, by modifying the Slashdot Zoo dataset in such way that the known trolls of the system perform the attack models introduced in Section 6.2.2. Finally, we present in Section 7.4.4 some experiments intended to test the usefulness of the inclusion of the *sources of distrust* in our algorithm.

7.4.1 Basic attack models

The first set of experiments is intended to show the performance of our approaches against the basic attacks explained in Section 6.2.2, and incremental combinations of them. For the evaluation of our proposals, we have generated graphs with 10^3 malicious users performing different sets of attacks against the network. The resulting graphs have about 10^5 edges, of which 25% are negative links. The graphs implementing the threat model D additionally have 100 spy nodes. Finally, we have used 10 nodes as sources of trust for all the experiments.

The error rate for each technique is shown in Table 7.2 where we can see that our system obtains very good results, improving EigenTrust and Fans-Minus-Freaks. SR and NR results are between PT_{ALL} and PT_{AR}, improving EigenTrust and FmF as well. This fact shows the relevance of using the negative links in the computation of trust.

Threats	ET	FmF	SR	NR	PT_{NN}	PT_{AR}	PT_{ALL}
A	0.535	0.446	0.354	0.271	0.265	0.175	**0.087**
A,B	0.535	0.446	0.355	0.272	0.265	0.175	**0.087**
A,B,C	0.526	0.649	0.345	0.272	0.256	0.166	**0.106**
A,B,C,D	0.528	0.650	0.344	0.272	0.255	0.166	**0.116**
A,B,C,D,E	0.527	0.527	0.345	0.282	0.261	0.169	**0.110**

Table 7.2 – Error rate for each technique against incremental attacks: EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PT_{NN}), Action-Reaction approach (PT_{AR}) and PolarityTrust (PT_{ALL}). For the attacks including the model D, we insert 100 spy nodes in the graph.

In Table 7.3 we show the $nDCG$, measuring the demotion of bad users in the ranking of trust. In this case, both extensions Non-Negative and Action-Reaction propagation present better performance than the baselines, while the complete technique is the best method in accordance to these results.

Threats	ET	FmF	SR	NR	PT_{NN}	PT_{AR}	PT_{ALL}
A	0.833	0.843	0.599	0.749	0.876	0.906	**0.987**
A,B	0.833	0.844	0.811	0.920	0.876	0.906	**0.987**
A,B,C	0.842	0.719	0.816	0.920	0.877	0.903	**0.984**
A,B,C,D	0.823	0.723	0.818	0.937	0.879	0.903	**0.984**
A,B,C,D,E	0.753	0.777	0.877	0.933	0.966	0.862	**0.982**

Table 7.3 – nDCG for each technique against incremental attacks: EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PT_{NN}), Action-Reaction approach (PT_{AR}) and PolarityTrust (PT_{ALL}). For the attacks including the model D, we insert 100 spy nodes in the graph.

7.4.2 Slashdot dataset experiments

Up to this point we have performed experiments with a set of randomly generated datasets, using the Barabasi and Albert generation method in order to obtain graphs that model real-world networks. In this section we show the results of our proposals on a real-world dataset: the Slashdot Zoo (see Section 17). The results of each technique have been evaluated using the error rate and the nDCG. In both cases we have evaluated the number

of bad users in positions above the last 96 of the ranking. In Table 7.4 we show the results.

For these experiments, the set of sources of trust is formed by the user *CmdrTaco*, corresponding to the founder of Slashdot.org Rob Malda, and the users that he has tagged as friends. There are 6 sources of trust in total.

	ET	FmF	SR	NR	PT_{NN}	PT_{AR}	PT_{ALL}
Error %	0.990	0.901	0.881	0.881	**0.861**	**0.861**	**0.861**
nDCG	0.310	0.460	0.479	0.477	**0.593**	0.570	0.588

Table 7.4 – *nDCG* and error rate for each technique processing Slashdot Zoo dataset. EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PT_{NN}), Action-Reaction approach (PT_{AR}) and PolarityTrust (PT_{ALL}).

We show the results of these experiments in Table 7.4. The scores in the error rate for every technique are higher than those in previous section due to the proportion of bad nodes in the Slashdot Zoo dataset (0.0013) in contrast to the randomly generated datasets of previous section (10%). In this case, *nDCG* is a more appropriate metric to compare the techniques because their values are normalized. Furthermore, a high error rate implies that many malicious users have reached the positions corresponding to good users, whilst *nDCG* gives a deeper view of the results, because a low value implies that the attackers not only have gained the positions of good users, but also that they have achieved good positions in the ranking.

In accordance with the data in Table 7.4, EigenTrust algorithm does not demote any of the 96 trolls in the last positions of the ranking, while FmF and SR perform slightly better and NR gets a very good result. Finally, our approaches outperform NR, presenting the best performance with this dataset, achieving a 10% of improvement over the rest of methods. This experiment highlights the usefulness of our proposal in a real environment.

7.4.3 Trolling Slashdot Zoo dataset

In this section we adopt the role of trolls in Slashdot.org. We try to perform some attacks against the network through the 96 tagged trolls in the Slashdot Zoo dataset. In this way, we can measure the impact of the different trolling techniques discussed in Section 6.2.2, and compare the behavior of the implemented techniques.

In order to create the datasets for these experiments we apply the same mechanism explained in Section 7.3.1 for the generation of bad users and

their out-links and in-links. Instead of taking a randomly generated graph as the network under attack, we have used the Slashdot Zoo dataset. The different attacks have been applied to this graph by adding a number of edges modeling the required behavior of the bad nodes. No new nodes have been added to the original dataset, so the amount of good and bad users is the same as in previous experiments.

Threats	ET	FmF	SR	NR	PT_{NN}	PT_{AR}	PT_{ALL}
A	0.990	0.901	0.881	0.881	**0.861**	**0.861**	**0.861**
A,B	1.000	0.901	0.880	0.880	**0.861**	**0.861**	**0.861**
A,B,C	1.000	0.901	0.881	0.881	**0.861**	**0.861**	**0.861**
A,B,C,D	1.000	0.925	0.935	0.940	0.851	0.856	**0.841**
A,B,C,D,E	1.000	0.925	0.940	0.935	0.851	0.856	**0.841**

Table 7.5 – Error rate for each technique processing the Slashdot Zoo dataset with synthetic attacks: EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PT_{NN}), Action-Reaction approach (PT_{AR}) and PolarityTrust (PT_{ALL}). For the attacks including the model D, we insert 100 spy nodes in the graph.

The experiments are evaluated applying the same metrics previously mentioned. In Table 7.5 we present the error rate for each technique against each threat model, counting the number of good users that appear in the last positions of the rankings. We can see that the addition of the attacks provoke an increasing number of errors using any of the studied methods but our proposals, whose performances remain reasonably stable.

Threats	ET	FmF	SR	NR	PT_{NN}	PT_{AR}	PT_{ALL}
A	0.310	0.460	0.479	0.477	**0.593**	0.570	0.588
A,B	0.308	0.460	0.478	0.477	**0.593**	0.570	0.588
A,B,C	0.311	0.460	0.474	0.484	**0.593**	0.570	0.588
A,B,C,D	0.370	0.476	0.501	0.501	0.580	0.570	**0.586**
A,B,C,D,E	0.370	0.475	0.501	0.496	0.580	0.574	**0.588**

Table 7.6 – nDCG for each technique processing the Slashdot Zoo dataset with synthetic attacks: EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PT_{NN}), Action-Reaction approach (PT_{AR}) and PolarityTrust (PT_{ALL}). For the attacks including the model D, we insert 100 spy nodes in the graph.

Regarding the positions where bad users are placed in the ranking, we show in Table 7.6 the $nDCG$ values for each experiment. The results of our methods achieve an improvement of about 10% over the rest of techniques. It means that the misclassified bad users are demoted in the ranking so their ranks, according to our approaches, are lower than the ones computed by the other methods.

7.4.4 Experiments with Sources of Distrust

All the experiments presented above have been performed with a set of sources of trust, and no *sources of distrust*. In this section we test this feature by taking as input the same datasets of the previous section, the Slashdot Zoo with the addition of synthetic attacks. As long as we just try to test whether this feature can be useful in our schema, we have randomly chosen 5 known trolls (foes of the special user *No_More_Trolls*) as sources of distrust, without following any specific heuristic.

	Sources of Trust			Sources of Trust & Distrust		
Threats	PT_{NN}	PT_{AR}	PT_{ALL}	PT_{NN}	PT_{AR}	PT_{ALL}
A	0.861	0.861	0.861	0.475	0.505	**0.465**
A,B	0.861	0.861	0.861	0.475	0.505	**0.465**
A,B,C	0.861	0.861	0.861	0.475	0.505	**0.465**
A,B,C,D	0.851	0.856	0.841	0.657	0.677	**0.642**
A,B,C,D,E	0.851	0.856	0.841	0.652	0.677	**0.637**

Table 7.7 – Error rate for our approaches using some known trolls as sources of distrust, taking as input the Slashdot Zoo dataset with synthetic attacks. Results for Non-Negative propagation approach (PT_{NN}), Action-Reaction approach (PT_{AR}) and PolarityTrust (PT_{ALL}) using sources of trust only, and sources of trust and distrust.

The results of our approaches with the inclusion of the sources of distrust are shown in Table 7.7 and 7.8, corresponding to the error rate and the $nDCG$, respectively. For better understanding, we include in these tables also the scores of the techniques using only sources of trust.

The improvement achieved by all the approaches in relation to the same techniques without the sources of distrust is evident, outperforming all the previous results. Regarding these results, this feature can be very useful in a system where some malicious users have been identified, penalizing those users who interact with them (creating a positive edge to any of them, for

	Sources of Trust			Sources of Trust & Distrust		
Threats	PT_{NN}	PT_{AR}	PT_{ALL}	PT_{NN}	PT_{AR}	PT_{ALL}
A	0.593	0.570	0.588	**0.846**	0.790	**0.846**
A,B	0.593	0.570	0.588	**0.846**	0.790	**0.846**
A,B,C	0.593	0.570	0.588	**0.846**	0.790	**0.846**
A,B,C,D	0.580	0.570	0.586	0.775	0.739	**0.782**
A,B,C,D,E	0.580	0.574	0.588	0.774	0.741	**0.781**

Table 7.8 – nDCG for our approaches using some known trolls as sources of distrust, taking as input the Slashdot Zoo dataset with synthetic attacks. Results for Non-Negative propagation approach (PT_{NN}), Action-Reaction approach (PT_{AR}) and PolarityTrust (PT_{ALL}) using sources of trust only, and sources of trust and distrust.

example). It can be also applicable together with some simple mechanisms that help to (vaguely) identify users who can be considered *a priori* as bad users according to their links, their comments or their behavior in general.

7.5 Summary

In this chapter we have presented PolarityTrust, a Trust and Reputation System based on the propagation of the positive and negative opinions of the users in a social network in order to compute their trustworthiness. Our proposal uses this information in order to obtain two scores: a positive one indicating the goodness of a user, and a negative one corresponding to its badness. Unlikely other approaches, in our method both positive and negative links influence both positive and negative scores, being able to implement the transitivity of trust and distrust in a social network.

The reliability of PolarityTrust has been proved by testing its behavior under some common attacks against TRS's, and also with a real-world dataset from the social news site Slashdot.org. The attacks consist in dishonest behaviors of malicious users who can take advantage of the vulnerabilities of a TRS in order to gain a high reputation in the network. We have also introduced a extension to the basic model of PolarityTrust intended to penalize those users who present a dishonest behavior, according to the intensity of this behavior. The experiments on synthetic datasets show that the performances of our approaches are not affected by the implemented attacks, clearly outperforming the results of other systems. Finally, the experiments with the Slashdot Zoo dataset show the reliability of PolarityTrust in a real-

world social network, even when new edges are added in order to perform a set of more sophisticated attacks against the network.

The concept of *sources of trust* allows to include in our TRS the different levels of users existing in social networks (administrators, moderators, normal users, etc). With PolarityTrust we can easily assign different weights to the opinions of the users regarding their category in the network. Otherwise, the use of *sources of trust and distrust* have proved to be a highly reliable method.

Finally, the flexibility shown by PolarityTrust when it comes to select the sources of trust and distrust and also the possibility of easily extending its functionalities are two really useful properties of our proposal. As we mentioned previously in this dissertation, since the methods for taking advantage or manipulating the existent TRS's present a constant evolution, the methods intended to deal with them must be as adaptable and flexible as possible in order to allow quick and easy changes in a highly variable and challenging environment.

Part IV

Conclusions

Chapter 8

Final remarks

Just one more thing...
Peter Falk
Columbo

8.1 Introduction

The analysis of dishonest behaviors in on-line networks as one of the most important challenges for both: the well-established WWW (or Web 1.0) with its on-line web search engines, and the astonishingly rising Web 2.0 with its user-generated content, motivated our work on these two closely related tasks.

Our background knowledge, acquired with our previous research works on graph-based techniques, STR (Cruz et al., 2006a; Ortega et al., 2011b) and PolarityRank (Cruz et al., 2011b), led us to tackle the dishonest behaviors problem from the point of view of the graph theory, together with natural language processing techniques. We have discussed these previous works in Chapter 3.

Given this framework, with those motivating real-world problems and our background knowledge, we formulated the following hypothesis:

The detection of dishonest behaviors in on-line networks can be carried out with graph-based techniques, flexible enough to include in their schemes specific information (in the form of features of the elements in a graph) about the network to be processed and the concrete task to be solved.

This dissertation presents the work developed in order to demonstrate the previous hypothesis through a research work divided in three main steps: the study of state-of-art systems in each field, the proposal of novel approaches intended to tackle the problems based on the premises stated in the hypothesis, and finally a thorough evaluation intended to verify the validity of the proposals.

According to the discussions and the results shown by the different proposals for each task (see Chapters 5 and 7), we can state that the hypothesis has been succesfully verified. In next sections we discuss in depth the conclusions extracted from both proposals: the web spam detection technique, PolaritySpam (see Section 8.2), and the trust and reputation system, PolarityTrust (see Section 8.3).

8.2 Web Spam Detection

Web spam is one of the first massive dishonest behaviors observed on the Internet, successor of the spam mechanisms intended to disturb the normal behavior of other on-line systems such as Usenet or the e-mail. In fact, it adopts the same (sur)name as its predecessors and, more important, also the main aims of the spam on Usenet and the e-mail, to wit: including information, (not just) commercial offers and/or hyperlinks to other sites in order to obtain web traffic to them, or just to insert annoying content without any other purposes.

In Chapter 4 we have shown a taxonomy of the common mechanism for web spam, and we also reviewed some of the most relevant web spam detection techniques, classifying them according to the type of information being processed into: content-based, link-based and hybrid systems. This last group is formed by those systems that tackle the web spam detection task combining both content-based and link-based information. They usually consist in a normal content-based system (a supervised classifier that contains a set of features characterizing the textual content of the web pages) with the addition of a number of link-based features, such as the amount of out-links and in-links of a web page, or even more elaborate (and expensive) metrics like the PageRank score, for example.

In this work we propose PolaritySpam (a deeper discussion can be consulted in Chapter 5) as a new hybrid web spam detection technique. In contrast to the other hybrid methods, PolaritySpam is a graph-based ranking algorithm that is enriched with content-based information. The underlying idea is to use the textual content within the web pages in order to extract some a priori knowledge that can be then propagated through the web graph.

In this way, the spam likelihood of a web page is computed according to its textual content and its proximity to spam or not-spam web pages. Our system follows the *topical endorsement assumption*, that can be summarized as follows: pages that are linked must belong to the same class. In our case this assumption can be translated as follows: pages that link to a (not-)spam web page are likely to be also (not-)spam.

The evaluation of our proposal have been carried out by comparing its results with those obtained with a well-known web spam detection technique, TrustRank (Gyöngyi et al., 2004). The results achieved by PolaritySpam are really good, not only in terms of pure comparison to the TrustRank algorithm, but also taking into account the fact that we have included only two content-based metrics in order to extract the a priori content-based knowledge. Furthermore, TrustRank uses a manual mechanism intended to choose a set of web pages whose information will be propagated through the web graph, while PolaritySpam implements an automatic method to perform this step.

In accordance to the results detailed in Section 5.4, we can state that the hypothesis formulated for the web spam detection task has been verified, so we can confirm that:

The inclusion of knowledge about the textual content of the web pages in addition to the relations between them, implicit in the web graph topology, into a graph-based model can improve the performance of a web spam detection system.

Finally, we would like to highlight the flexibility of our technique, that can make it possible the easy adaptation of PolaritySpam to new possible threats in this field in terms of new web spam methods or even new environments where those methods can be applied, such as the spam on the blogosphere (namely, *splog*) and also the rising spam on social networks.

8.3 Trust and Reputation

The rising of on-line social networks, with the focus on the user-generated contents, has constituted a change in the paradigm of the Web, commonly stated as the Web 2.0. It is said that the Internet has evolved from a (more or less) static collection of information, where the focus was on the web pages, to a dynamic system where the main actors are the users who create and share different kinds of contents through a wide variety of systems that usually forms on-line communities or social networks. In other words, the focus has

moved from the information encoded in the web pages, to the users who are not only the consumers but also the principal providers of information.

This step towards the Web 2.0 has brought also the evolution of some tasks that were already common in the WWW. This is the case of the web search problem, that is the search for relevant web pages about a certain topic, avoiding those sites without useful information. In the Web 2.0, one of the main problems is the search for relevant users, due to their leading role in the new paradigm. As the case of the web spam detection in the web search, this new task suffers for the problem of dishonest behaviors of users who try to gain high reputation in the systems in order to obtain some kind of benefits. This problem has been studied in Chapter 6, in addition to the mechanisms intended to deal with it, namely the Trust and Reputation Systems (TRS), and the vulnerabilities and problems shown by these TRS's.

We present in Chapter 7 PolarityTrust, our proposal intended to tackle the problem of evaluating the relevance of users in a social network. PolarityTrust processes a social network modeled as a graph whose nodes corresponds to the users of the social network and the edges represent the relations between those users. Given a set of trustworthy users in the network (they can be the administrators of the site, or a group of moderators) PolarityTrust propagates their knowledge through the graph in order to compute two scores for each user: a positive score that represents its trustworthiness, and a negative one representing its untrustworthiness. Other novelty of PolarityTrust is the inclusion of the negative relations between the users in the computation of the iterative algorithm, in such way that both positive and negative relations in the network influence the two scores of each user, depending on the strength of the relations and the previous scores of the users. Finally, the other main novelty of our proposal is the development of two propagation mechanisms, Non-Negative Propagation and Action-Reaction Propagation, in order to spread the opinions of the users depending on their reputation and their actions in the network.

The evaluation of the system has been performed by using three different datasets: first, we have evaluated PolarityTrust with a set of randomly generated graphs, modeling a number of previously studied attacks; then a real-world dataset extracted from the social news site Slashdot.org has been processed; finally we have tested our proposal with a modified version of the Slashdot dataset, that included the same set of attacks mentioned in the first step. In each step of our evaluation we have compared PolarityTrust to other four techniques. As we show in Section 7.4, our proposal presents a very good performance in all the experiments, improving the results of the other four techniques.

In accordance to the discussion and the results, we can state that the hypothesis formulated for the trust and reputation task has been verified, so we can confirm that:

Taking into account the negative links (edges with a negative weight) in a network in addition to the positive ones improves the discriminative ability of a system intended to detect dishonest behaviors in on-line networks.

Furthermore, the use of various strategies when it comes to propagate the information of each node through the network, such as the Non-Negative Propagation and the Action-Reaction approach, shows the ability of PolarityTrust to be easily adapted depending on the nature of the social network to be processed and the capabilities that its users are provided with. This is a very important feature for a trust and reputation system, due to the constant evolution of the methods intended to overcome this kind of systems.

Chapter 9

Future work

La meta è partire
(The goal is to depart)
Giuseppe Ungaretti

9.1 Introduction

The research work shown in this dissertation presents two proposals intended to tackle two variants of the problem of the detection of dishonest behaviors in on-line networks. The first one is focused on the detection of web spam through the computation of link-based and content-based information from the web pages. The second work is focused on the detection of untrustworthiness users in social networks through the study of the positive and negative relationships established between them. The results obtained in both cases are good, successfully dealing with the difficulties that motivated our work.

Nevertheless, both proposals present some active fronts in their respective tasks, and they also open new lines of research in other contexts. In this section we detail the future work that is waiting to be explored following this dissertation. We discuss first the ones corresponding to the web spam detection in Section 9.2, and then in Section 9.3 we point out the upcoming challenges in the detection of dishonest behaviors in social networks.

9.2 Web Spam Detection

Concerning the spam detection technique, PolaritySpam, we plan to further our research by studying the relation between the improvement achieved by the inclusion of new heuristics and the time complexity of our algorithm. In principal, the addition of new metrics can improve the a priori information to be processed, but a low time complexity is strongly required in this kind of systems so we must evaluate carefully the benefits introduced by the new metrics in relation to the penalization in the performance of the algorithm in the step of computing the a-priori information.

Speaking of the improvements in the time complexity of our system, it would be interesting to study the parallelization of our algorithm, maybe with similar methods to those introduced in (Desikan et al., 2006; Haveliwala, 1999; Kamvar et al., 2003a; Kohlschütter et al., 2006; Wicks and Greenwald, 2007). These works propose different parallelization schemas for random-walk algorithms, by *divide and conquer* or similar strategies. Since PolaritySpam is also based on a random-walk algorithm, the parallelization of the algorithm can result in a really useful work due to the huge amount of information to be processed in a web spam detection system. Likewise it is necessary to evaluate the costs and benefits that PolaritySpam and this adaptation would entail in a real-world environment.

It would be also interesting to find out the influence of the positive and negative source sets in the overall ranking of nodes, in order to be able to improve the source selection methods in different aspects, such as determining the minimum number of sources needed to obtain good results in the spam detection, or the maximum amount of sources before the method gets overfitted. In this respect, we also plan to research additional methods for the selection of sources, taking into account not only the content-based features from the web pages, but also their capability for the propagation of information through the web graph.

Regarding the development of new features, we think that some of the ideas presented in PolarityTrust can be adapted to the detection of web spam. For example, the Action-Reaction propagation (see Section 7.2.3) can be a suitable method in order to penalize those web pages that have some links to spam web pages, or the complementary case: spam web pages that have links to non-spam ones. In this last situation, the spam web page reduces the relevance of the non-spam web page, in order to achieve a better position in the ranking. Applying the Action-Reaction mechanism, the penalization would affect the relevance of the spam web page instead. These phenomena could be easily detected with the help of the content-based heuristics, analyzing those edges in the web graph that link two nodes with

very different values in their content-based metrics. However, this feature must be applied carefully, bearing in mind that, since it is an inferred relation, this assumption is weaker than an exising negative relation between two users in a social network.

Finally, another interesting line of research consists in applying these techniques in the detection of spam in social networks, a problem that is growing in parallel to the use of these on-line communities. The combination of the ideas behind PolaritySpam for the detection of spam messages, and PolarityTrust for the detection of untrustworthy user accounts, can result a really useful method to avoid all kinds of spam in on-line social networks, such as the spam messages in the comments section of a blog (*splogs*), the automatic detection of user accounts intended to perform these kind of actions according to their contributions to the system and their relations into the network, or even the fraud detection in on-line auctions and marketplaces (Chau and Faloutsos, 2005; Dong et al., 2009).

9.3 Trust and Reputation

Concerning the trust and reputation of users in social networks, we plan to further our research by studying other types of attacks against TRS's, including the use of *playbook sequences* (Kerr and Cohen, 2009), consisting in a sequence of actions intended to gain high trustworthiness in a system. There is an infinite set of possible playbook sequences, and they can be influenced by other users playbooks, making these attacks really hard to detect and to avoid. The intuition behind playbook attacks is that it can not be assessed that a TRS is effective just because the potential attackers do not know how it works. In other words: it is necessary to asses whether a TRS is effective or not against an omniscient enemy who knows exactly how it works.

Apart from the Denial of Service (DoS) and similar attacks that harm the on-line systems from the point of view of their operability or their connectivity, and the inherent vulnerabilities of the social networks discussed in Section 6.2.1, the only way of disturbing the performance of a TRS is by taking advantage of the actions allowed by the system itself to its users. In this sense, it could be interesting to carry out a deep theoretical study on the different interactions that can be performed in a social network, building a taxonomy for them according to the functionalities that they provide to their users. Given the high flexibility of PolarityTrust, it would be possible to extend it in order to focus on each category in the taxonomy of social networks. Since the actions that a social network allows to its users can be

also considered as weak points where the system can be attacked from, a deep knowledge of the potential weaknesses of each system is a good first step towards the development of a reliable method to tackle the omniscience of the potentially malicious users.

It is also interesting to test some methods for the selection of sources of distrust in the system, such as link-based heuristics (number of out-links, number of in-links, inverse PageRank). This mechanism could be useful in a semi-supervised schema, serving as an assisting tool for experts or moderators who would only have to assess the (un)trustworthiness of a small amount of users and delegating the rest of the process to our system.

We also plan to study the influence of our TRS in the performance of a social network in terms of time complexity. In a similar way as PolaritySpam, before PolarityTrust could be applied in a real-world social network, it would be necessary to study a possible parallelization of the algorithm, and the costs and benefits that this adaptation would entail in a specific environment.

Finally, we are interested in the application of PolarityTrust in other contexts where the concept of trust and/or reputation can be modeled in terms of positive and negative relations between entities. For example, the detection of trending topics in the blogosphere or in micro-blogging sites like Twitter, taking into account not only the amount but also the polarity of the comments in the blogs (or the "replies" and "retweets" in Twitter).

On the other hand, the reputation management of companies or important figures is also an attractive topic where our research work can be applied, in combination with some NLP methods such as Opinion Mining techniques. Also the recommender systems, where positive and negative opinions are important for the computation of the users preferences, constitute a very interesting research area. In this task, the ideas behind PolaritySpam can be easily applied, inferring previously the polarity of the opinions of the users about the items, and using these polarities to characterize the edge weights of a graph of users and items. The above-mentioned parallelization of our techniques can be very useful for these tasks, due to the real-time requirements on these kind of systems.

Part V

Appendices

Appendix A

Algorithms for the Random Generation of Graphs

A.1 Barabási-Albert model

Barabási-Albert model (Barabási and Albert, 1999) creates graphs whit the *Preferential Attachment* property, consisting in the fact that the newcomers of a network attach preferentially to the most connected nodes of the system. This fact leads to the generation of scale-free networks whose nodes have a probability, $P(k)$, of interacting with k other nodes that follows a power-law distribution of the form $P(k) \sim ck^{-\gamma}$.

The pseudo-code of the method is as follows:

Data: N = number of nodes;
Result: $G = randomNetwork$
1 Initialize G with m_0 nodes, where $m_0 \geq 2$ and $\forall n_i \in G, Degree(n_i) \geq 1$;
2 **while** $\mid G \mid < N$ **do**
3 Add n_j to G;
4 **forall the** $n_i \in G$ *with* $n_i \neq n_j$ **do**
5 Create an edge E_{ji} from j to i with a probability of: $P_{ji} = \frac{Degree(n_i)}{\sum_k Degree(n_k)}$;
6 **end**
7 **end**

Algorithm 2: BA algorithm for the generation of random scale-free networks

A.2 Forest Fire model

This generation method is proposed in (Leskovec et al., 2005). It is based in two observations inferred from the growing patterns of real world networks:

- *Densification law*: most of the networks densifies over time, with the number of edges growing exponentially in the number of nodes.
- *Shrinking diameter*: it states that the average distance between the nodes in a graph often shrinks over time.

The pseudo-code of the algorithm can be specified as follows:

Data: N = number of nodes;
p = forward burning probability;
r = backward burning ratio;
Result: $G = randomNetwork$

```
while | G |< N do
    Add a new node, v, to G;
    Pick an ambassador w, uniformly at random;
    Create an edge linking v with w;
    Choose a random number, x, from a binomial distribution with
    mean (1 − p)^−1;
    Node v_i selects x edges of w, picking in-links with probability r
    times less than out-links;
    Let w_1, w_2, ..., w_x be the other ends of these edges;
    for 1 ≤ i ≤ x do
        Apply step [2] recursively to w_i;
    end
end
```

Algorithm 3: ForestFire model for the generation of random scale-free networks

A.3 Evolving Copying models

These models are proposed in (Kumar et al., 2000). They are focused on creating random graphs with two main properties: the power-law distribution of the degrees of the nodes and the *community guided attachment*. The last one is produced by the fact that similar elements (for example, web pages on the same topic) are prone to be highly interconnected, forming communities.

They propose a set of methods based on these ideas. We include in this section the linear-growth model in order to illustrate the intuition behind this technique.

Data: $\alpha \in (0, 1)$;
Max. Out-degree $d \geq 1$;
Number of Nodes $= N$;
Result: $G = RandomNetwork$

```
1  while | G |< N do
2  |   Add a new node, u, to G;
3  |   Pick an ambassador p, uniformly at random;
4  |   while Out − degree(u) < d do
5  |   |   Choose r ∈ (0, 1) uniformly at random ;
6  |   |   if r < α then
7  |   |   |   Choose a node v ∈ G at random;
8  |   |   |   Create an edge from u to v;
9  |   |   else
   |   |   |   // Copying process
10 |   |   |   Choose e, an out-link of p at random;
11 |   |   |   Create an edge from u to the tail of e;
12 |   |   end
13 |   end
14 end
```

Algorithm 4: Evolving copying model: linear growth.

Appendix B

PolarityRank: Algebraic Proof and Convergence

B.1 Introduction

We show in this appendix the algebraic and the convergence proofs of PolarityRank algorithm (see Section 3.3). First we review the definition of the algorithm, the algebraic proof and the convergence proof are subsequently shown.

The mathematical proofs included in this appendix were carried out by Mr. Carlos G. Vallejo in Cruz et al. (2011b).

B.2 Definition

Let $G = (V, E)$ be a directed graph where V is a set of nodes and E a set of directed edges between two nodes. An edge from the node v_i to the node v_j, $e_{ij} \in E$, has an associated weight, $p_{ij} \neq 0$. Let $Out(v_i)$ be the set of j such as $\exists e_{i,j} \in E$ with $v_j \in V$. Let $In^+(v_i)$ as the set of j such as $\exists e_{j,i} \in E$ and $v_j \in V$ with $p_{j,i} \geq 0$, and analogously for $In^-(v_i)$. Let e^+ and e^- be two vectors with values greater than zero for those nodes that are taken as positive or negative seeds in our algorithm, respectively, or zero in other cases.

We compute positive and negative PolarityRank, PR^+ and PR^-, respectively, as follows:.

$$
\begin{aligned}
PR^+(v_i) = (1-d)e_i^+ + \\
+ d\Big(\sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PR^+(v_j) + \\
+ \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PR^-(v_j) \Big) \\
PR^-(v_i) = (1-d)e_i^- + \\
+ d\Big(\sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PR^-(v_j) + \\
+ \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PR^+(v_j) \Big)
\end{aligned}
\tag{B.1}
$$

where $d \in (0, 1]$ is the damping factor, and the sum of all the values of e^+ and e^-, separately, must be equal to the number of nodes in the graph.

B.3 Algebraic Proof

In this section the algebraic foundation of PolarityRank is studied. Let $n = |V|$ be the number of nodes in the graph and P the matrix formed by the weights of the edges, (p_{ij}). Let us define $p_j = \sum_{k \in Out(v_j)} |p_{jk}|$, i.e. the sum of the weights of all the edges starting at v_j. Given P, p_j is the sum of the absolute values of the row j. It can be also written as $p_j = \sum_{k=1}^{n} |p_{jk}|$. Now, we can formulate PolarityRank as follows:

$$
\begin{aligned}
PR^+(v_i) = (1-d)e_i^+ + \\
+ d\Big(\sum_{j \in In^+(v_i)} \frac{p_{ji}}{p_j} PR^+(v_j) + \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{p_j} PR^-(v_j) \Big) \\
PR^-(v_i) = (1-d)e_i^- + \\
+ d\Big(\sum_{j \in In^+(v_i)} \frac{p_{ji}}{p_j} PR^-(v_j) + \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{p_j} PR^+(v_j) \Big)
\end{aligned}
$$

Let us define now $Q = P^t$ as the transposed of P (note that if P describes a non-directed graph then $Q = P$). Thus, $q_j = \sum_{k=1}^{n} |q_{kj}|$, that is the sum of the elements in the column j in Q. Obviously, $p_j = q_j$.

Then, we define two matrices $Q^+ = (q^+_{ij})$ and $Q^- = (q^-_{ij})$ as follows:

$$q^+_{ij} = \begin{cases} q_{ij} & \text{if } q_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$
$$q^-_{ij} = \begin{cases} -q_{ij} & \text{if } q_{ij} < 0 \\ 0 & \text{otherwise} \end{cases}$$

q_j can be seen as the sum of the elements of the column j from matrix Q^+ plus the sum of the elements of the same column in the matrix Q^-.

PolarityRank can be now formulated as follows:

$$PR^+(v_i) = (1-d)e^+_i + d\Big(\sum_{j=1}^{n} \frac{q^+_{ij}}{q_j} PR^+(v_j) + \sum_{j=1}^{n} \frac{q^-_{ij}}{q_j} PR^-(v_j)\Big)$$
$$PR^-(v_i) = (1-d)e^-_i + d\Big(\sum_{j=1}^{n} \frac{q^-_{ij}}{q_j} PR^+(v_j) + \sum_{j=1}^{n} \frac{q^+_{ij}}{q_j} PR^-(v_j)\Big)$$

We define now $A^+ = (a^+_{ij})$ as $a^+_{ij} = q^+_{ij}/q_j$ and the matrix $A^- = (a^-_{ij})$ as $a^-_{ij} = q^-_{ij}/q_j$, then

$$PR^+(v_i) = (1-d)e^+_i + d\Big(\sum_{j=1}^{n} a^+_{ij} PR^+(v_j) + \sum_{j=1}^{n} a^-_{ij} PR^-(v_j)\Big)$$
$$PR^-(v_i) = (1-d)e^-_i + d\Big(\sum_{j=1}^{n} a^-_{ij} PR^+(v_j) + \sum_{j=1}^{n} a^+_{ij} PR^-(v_j)\Big)$$

In order to simplify the notation, we make some definitions. Let $m = 2n$. We define the vector x with $m \times 1$ elements as

$$x = \begin{bmatrix} PR^+(v_1) \\ \vdots \\ PR^+(v_n) \\ PR^-(v_1) \\ \vdots \\ PR^-(v_n) \end{bmatrix}$$

and the vector e with $m \times 1$ elements is

$$e = \begin{bmatrix} e_1^+ \\ \vdots \\ e_n^+ \\ e_1^- \\ \vdots \\ e_n^- \end{bmatrix}$$

Finally, we define the matrix A with $m \times m$ elements as follows:

$$A = \left[\begin{array}{c|c} A^+ & A^- \\ \hline A^- & A^+ \end{array}\right]$$

Given the previously defined auxiliary vectors and matrices, we can write the PolarityRank equations (B.1) much more easily as:

$$x = (1-d)e + dAx \tag{B.2}$$

A is an stochastic matrix: All the elements in A are between 0 and 1 (both inclusive):

$$a_{ij}^+ = \frac{q_{ij}^+}{\sum_{k=1}^n q_{kj}^+ + \sum_{k=1}^n q_{kj}^-} \quad \text{y} \quad a_{ij}^- = \frac{q_{ij}^-}{\sum_{k=1}^n q_{kj}^+ + \sum_{k=1}^n q_{kj}^-}$$

and, obviously, the sum of the elements in each column is 1.

e_i^+ y e_i^- have been chosen in such way that $\sum_{i=1}^n e_i^+ = \sum_{i=1}^n e_i^- = n$ and positive. Thus, $\|e\|_1 = m$.

Now, we define a vector with $m \times 1$ elements, $f = e/m$ (i.e. the elements of e_i^+ and e_i^- divided by m), and the vector u as a vector with $m \times 1$ 1's. Let us analyze the matrix fu^t:

$$fu^t = \begin{bmatrix} e_1^+/m \\ \vdots \\ e_n^+/m \\ e_1^-/m \\ \vdots \\ e_n^-/m \end{bmatrix} \begin{bmatrix} 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} e_1^+/m & \dots & e_1^+/m \\ \vdots & & \vdots \\ e_n^+/m & \dots & e_n^+/m \\ e_1^-/m & \dots & e_1^-/m \\ \vdots & & \vdots \\ e_n^-/m & \dots & e_n^-/m \end{bmatrix}$$

Its elements are between 0 and 1, and the sum of the elements in each column is 1, so fu^t is a stochastic matrix.

If x is normalized so that $\|x\|_1 = m$, then $u^t x = m$, provided that the elements in x are positive; but if we start giving non-negative values, they will always remain non-negative since the elements in A are also non-negative.

Therefore, the equation (B.2) can be written as follows:

$$\begin{aligned} x &= (1-d)e + dAx = \\ &= (1-d)e\frac{1}{m}u^t x + dAx = \\ &= (1-d)fu^t x + dAx = \\ &= ((1-d)fu^t + dA)x \end{aligned}$$

Let $B = ((1-d)fu^t + dA)$. The definition of PolarityRank (B.1) can be rewritten as:

$$x = Bx \tag{B.3}$$

B is the convex combination (linear combination where the two coefficients are positive and sum to 1) of two stochastic matrices, then B is also stochastic. Hence, as a result of Perron-Frobenius theorem, the equation (B.3) has the eigenvalue 1, and the rest of eigenvalues have a modulus less than 1 (may be complex). The previous system has a solution that is just the eigenvector corresponding to eigenvalue 1.

B.4 Convergence Proof

The computation of PolarityRank (vector x) can be performed from the expression (B.1) by an iterative process. In this section we prove the convergence of this process.

As we have discussed before, the expression (B.1) equals the (B.2): $x = (1-d)e + dAx$. Let us express x_k as the k-th term of the PolarityRank iterative computation:

$$x_{k+1} = (1-d)e + dAx_k$$

and

$$x_k = (1-d)e + dAx_{k-1}$$

Therefore,

$$\|x_{k+1} - x_k\| = d\|A(x_k - x_{k-1})\| \leq d\|A\|\|(x_k - x_{k-1})\|$$

for any norm compatible with the vector and matrix. For example, using norm 1,

$$\|x\|_1 = \sum_{i=1}^{m} |x_i|$$

and

$$\|A\|_1 = \max_{j=1...m} \sum_{i=1}^{m} |a_{ij}|$$

In this case, $\|A\|_1 = 1$ (recall that it is stochastic), and $d < 1$, then $\lim_{k\to\infty} \|x_{k+1} - x_k\| \to 0$.

Thus, the convergence is guaranteed.

Bibliography

J. Abernethy, O. Chapelle, and C. Castillo. Web spam identification through content and hyperlinks. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pages 41–44, New York, NY, USA, April 2008a. ACM. ISBN 978-1-60558-159-0.

J. Abernethy, O. Chapelle, and C. Castillo. Web spam identification through content and hyperlinks. In *AIRWeb '08: Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pages 41–44, New York, NY, USA, 2008b. ACM. ISBN 978-1-60558-159-0.

G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005. ISSN 1041-4347.

N. Agarwal, H. Liu, L. Tang, and P. S. Yu. Identifying the influential bloggers in a community. *Proceedings of ACM WSDM Conference*, pages 207–218, 2008.

C. Aggarwal, J. Wolf, K. Wu, and P. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 201–212. ACM, 1999. ISBN 1581131437.

E. Agirre and A. Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'09, pages 33–41, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1609067.1609070`.

I. Agudo, C. Fernández-Gago, and J. López. An evolutionary trust and distrust model. *Electronic Notes in Theoretical Computer Science*, 244: 3–12, 2009. ISSN 1571-0661.

E. Aguirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on - NAACL '09*, page 19, Morristown, NJ, USA, 2009. Association for Computational Linguistics. ISBN 9781932432411. doi: 10.3115/1620754.1620758.

L. Akritidis, D. Katsaros, and P. Bozanis. Identifying Influential Bloggers : Time Does Matter. *Acm, IEEE W I C*, 2009. doi: 10.1109/WI-IAT.2009.18.

E. Amigó, J. Artiles, J. Gonzalo, D. Spina, B. Liu, and A. Corujo. Weps3 evaluation campaign: Overview of the on-line reputation management task. In *CLEF 2010 LABs and Workshops, Notebook Papers*, 2010.

S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010. European Language Resources Association (ELRA).

L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. *CoRR*, abs/1111.4570, 2011.

J. Baldridge, T. Morton, and G. Bierner. Maxent, mature java package for training and using maximum entropy models. Technical report, 2005.

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, Oct. 1999. ISSN 00368075.

L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of web spam. In *Proceedings of the Second International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 1–8, 2006.

A. A. Benczur, K. Csalogany, T. Sarlos, M. Uher, and M. Uher. Spamrank - fully automatic link spam detection. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 25–38, 2005.

C. Biemann, I. Matveeva, R. Mihalcea, and D. Radev. Graph-based algorithms for natural language processing. In *Workshop at HLT/NAACL*, 2007.

I. Bíró, D. Siklósi, J. Szabó, and A. A. Benczúr. Linked latent Dirichlet allocation in web spam filtering. In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web - AIRWeb '09*, page 37, New York, New York, USA, Apr. 2009. ACM Press. ISBN 9781605584386. URL `http://portal.acm.org/citation.cfm?id=1531914.1531922`.

R. Blanco and C. Lioma. Random walk term weighting for information retrieval. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07*, page 829, 2007. doi: 10.1145/1277741.1277930.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. 3: 993–1022, 2003. URL `http://dblp.uni-trier.de/db/journals/jmlr/jmlr3.html#BleiNJ03`.

T. Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, Seattle, USA, 2000.

T. Bray. Measuring the web. *Comput. Netw. ISDN Syst.*, 28:993–1005, May 1996. ISSN 0169-7552. URL `http://dx.doi.org/10.1016/0169-7552(96)00061-X`.

L. Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. ISSN 0885-6125.

E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

R. Burke. Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The adaptive web*, pages 377–408. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-72078-2.

R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5.

C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, and S. Vigna. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, December 2006. ISSN 0163-5840.

C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: web spam detection using the web topology. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7.

D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators and algorithms. *ACM Computing Surveys*, 1(2):1–78, 2006.

S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *SIGMOD Rec.*, 27:307–318, June 1998. ISSN 0163-5808. URL `http://doi.acm.org/10.1145/276305.276332`.

D. H. Chau and C. Faloutsos. Fraud detection in electronic auction. In *European Web Mining Forum at ECML/PKDD*. In European Web Mining Forum at ECML/PKDD, 2005. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.88.2438`.

Z. Cheng and N. Hurley. Analysis of robustness in trust-based recommender systems. In *Adaptivity, Personalization and Fusion of Heterogeneous Information*, RIAO '10, pages 114–121, Paris, France, France, 2010. Le Centre de Hautes Etudes Internationales D'Informatique Documentaire.

J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 161–172, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

M. Clements, A. de Vries, and M. Reinders. Exploiting positive and negative graded relevance assessments for content recommendation. *Algorithms and Models for the Web-Graph*, pages 155–166, 2009.

G. V. Cormack, M. Smucker, and C. L. A. Clarke. Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets. *Computing Research Repository*, abs/1004.5, 2010.

N. Cristianini. Support Vector and Kernel Machines. In *18th International Conference on Machine Learning*, 2001.

F. L. Cruz, J. A. Troyano, and F. Enríquez. Supervised textrank. *Advances in Natural Language Processing*, 31:632–639, 2006a. ISSN 0302-9743.

F. L. Cruz, J. A. Troyano, F. Enríquez, and F. J. Ortega. Textrank como motor de aprendizaje en tareas de etiquetado. *Procesamiento del Lenguaje Natural*, 37:7, 2006b.

F. L. Cruz, J. A. Troyano, F. J. Ortega, and F. Enríquez. Automatic Expansion of Feature-Level Opinion Lexicons. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011)*, pages 125–131, Portland, Oregon, June 2011a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W11-1716`.

F. L. Cruz, C. G. Vallejo, F. Enríquez, and J. A. Troyano. Polarityrank: Finding an equilibrium between followers and contraries in a network. *Information Processing and Management*, 2011b. ISSN 0306-4573.

L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, January 2005.

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. mbt: A memory-based part of speech tagger generator. In *Proceedings of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT, 1996.

N. Dai, B. D. Davison, and X. Qi. *Looking into the past to better classify web spam.* ACM Press, New York, New York, USA, Apr. 2009. ISBN 9781605584386. URL `http://portal.acm.org/citation.cfm?id=1531914.1531916`.

P. K. Desikan, N. Pathak, J. Srivastava, and V. Kumar. *Divide and conquer approach for efficient pagerank computation.* ACM Press, New York, New York, USA, July 2006.

S. Deterding, K. O'Hara, M. Sicart, D. Dixon, and L. Nacke. Gamification: Using game design elements in non-gaming contexts. pages 2425–2428, 2011.

Y. Ding. Topic-based pagerank on author co-citation networks. *Journal of the American Society for Information Science and Technology*, 62(3): 449–466, 2011. ISSN 1532-2890. doi: 10.1002/asi.21467.

D. Donato and M. P. Stefano Leonardi. Combining transitive trust and negative opinions for better reputation management in social networks. In *Workshop on Social Network Mining and Analysis (SNA-KDD)*, 2008.

F. Dong, S. M. Shatz, and H. Xu. Combating online in-auction fraud: Clues, techniques and challenges. *Computer Science Review*, 3(4):245–258, 2009.

M. Erdélyi, A. Garzó, and A. A. Benczúr. *Web spam classification*. ACM Press, New York, New York, USA, Mar. 2011. ISBN 9781450307062. URL `http://portal.acm.org/citation.cfm?id=1964114.1964121`.

G. Erkan and D. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1): 457–479, 2004.

C. Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998. ISBN 026206197X. URL `http://www.worldcat.org/isbn/026206197X`.

D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases*, pages 1–6, New York, NY, USA, 2004. ACM.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada, 2003.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

M. Gamon. Graph-Based Text Representation for Novelty Detection. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, pages 17–24, New York City, June 2006. Association for Computational Linguistics.

G.-g. Geng, X.-c. Zhang, D.-x. Zhang, and X.-b. Jin. Evaluating Web Content Quality via Multi-scale Features. In *Proceedings of the ECML/PKDD*, number 1, pages 5–8, 2010.

N. S. Glance, M. Hurst, and T. Tomokiyo. BlogPulse : Automated Trend Discovery for Weblogs. In *WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*. ACM, 2004.

J. Golbeck and J. Hendler. Filmtrust: movie recommendations using trust in web-based social networks. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 1, pages 282–286, jan. 2006. doi: 10.1109/CCNC.2006.1593032.

J. A. Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland at College Park, College Park, MD, USA, 2005.

H. Guan, J. Zhou, and M. Guo. A class-feature-centroid classifier for text categorization. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 201–210, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. URL `http://doi.acm.org/10.1145/1526709.1526737`.

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

Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases*, volume 30, pages 576–587, Toronto, Canada, 2004. VLDB Endowment. ISBN 0-12-088469-0.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278.

v. Halteren, J. Zavrel, and W. Daelemans. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27:31, 2001.

S. Hassan and C. Banea. Random-walk term weighting for improved text classification. In *Proc. of the International Conference on Semantic Computing*, pages 242–249, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2997-6.

V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

T. Haveliwala. Efficient Computation of PageRank, 1999.

T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15:2003, 2003.

M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2):11–22, Fall 2002.

K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys*, 42(1):1–31, dec 2009. ISSN 03600300.

J. L. Iribarren and E. Moro. Branching dynamics of viral information spreading. *Physical Review E*, 84:046116, Oct 2011. doi: 10.1103/PhysRevE.84.046116. URL http://link.aps.org/doi/10.1103/PhysRevE.84.046116.

K. Ishida. Extracting latent weblog communities: A partitioning algorithm. In *WWW 2005 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*. ACM, 2005.

M. Jamali and M. Ester. TrustWalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 397–406. ACM, 2009.

G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. URL http://doi.acm.org/10.1145/775047.775126.

Y. Jin, Y. Matsuo, and M. Ishizuka. Ranking companies on the web using social network mining. In I.-H. Ting and H.-J. Wu, editors, *Web Mining Applications in E-commerce and E-services*, volume 172 of *Studies in Computational Intelligence*, pages 137–152. Springer Berlin / Heidelberg, 2009. ISBN 978-3-540-88080-6. URL http://dx.doi.org/10.1007/978-3-540-88081-3_8.

A. Jø sang and J. Golbeck. Challenges for robust trust and reputation systems. In *5th International Workshop on Security and Trust Management*, Saint Malo, France, 2009. Elsevier.

A. Jø sang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

S. Kamvar, T. Haveliwala, and G. Golub. Adaptive Methods for the Computation of PageRank, 2003a.

S. D. Kamvar, M. T. Schlosser, and H. Garcia-molina. The eigentrust algorithm for reputation management in p2p networks. In *In Proceedings of the Twelfth International World Wide Web Conference*, pages 640–651. ACM Press, 2003b.

R. Kerr and R. Cohen. Smart cheaters do prosper : Defeating trust and reputation systems the security of trses. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, Budapest (Hungary), 2009.

J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models, and methods. In *Computing and Combinatorics*, volume 1627 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin / Heidelberg, 1999.

J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999. ISSN 0004-5411.

C. Kohlschütter, P.-A. Chirita, R. Chirita, and W. Nejdl. Efficient Parallel Computation of PageRank, 2006.

P. Kolari, T. Finin, and A. Joshi. Svms for the blogosphere: Blog identification and splog detection. In *AAAI Spring Symposium on Computational Approaches to Analysing Weblogs*, pages 92–99, Baltimore County, Maryland,USA, 2006. University of Maryland.

Y. Koren and R. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer US, 2011. ISBN 978-0-387-85820-3.

V. Krishnan. Web spam detection with anti-trustrank. In *ACM SIGIR workshop on Adversarial Information Retrieval on the Web*, Seattle, Washington, USA, 2006. ACM.

R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 57–65. IEEE Computer Society, 2000. ISBN 0-7695-0850-2. doi: 10.1109/SFCS.2000.892065.

R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 568–576, New York, NY, USA, 2003. ACM. ISBN 1-58113-680-3. URL `http://doi.acm.org/10.1145/775152.775233`.

J. Kunegis, A. Lommatzsch, and C. Bauckhage. The slashdot zoo: Mining a social network with negative edges. In *18th International World Wide Web Conference*, page 741, apr 2009.

R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Transactions on Information Systems*, 19(2):131–160, 2001. ISSN 1046-8188. doi: 10.1145/382979.383041.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *International Conference on Knowledge Discovery and Data Mining*, pages 177—-187, Chicago, USA, 2005.

R. L. Levien. *Attack Resistant Trust Metrics*. PhD thesis, 2002.

R. L. Levien and A. Aiken. Attack-Resistant Trust Metrics for Public Key Certification. In *7th USENIX Security Symposium*, pages 229–242, 1998. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.9335`.

E. Lex, I. Khan, H. Bischof, and M. Granitzer. Assessing the Quality of Web Content. In *Proceedings of the ECML/PKDD*, 2010.

X. Li, S. Szpakowicz, and S. Matwin. A WordNet-based algorithm for word sense disambiguation. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 14, pages 1368–1374. Citeseer, 1995.

Y.-R. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. Tseng. *Splog Detection using Content, Time and Link Structures*. IEEE, July 2007. ISBN 1-4244-1016-9. URL `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4285079`.

P. Lops, M. Gemmis, and G. Semeraro. Content-based Recommender Systems: State of the Art and Trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, chapter 3, pages 73–105. Springer US, Boston, MA, 2011. ISBN 978-0-387-85819-7. doi: 10.1007/978-0-387-85820-3_3.

C. Macdonald, I. Ounis, and I. Soboroff. Overview of trec-2009 blog track. In *In Proceedings of TREC 2009*, 2009.

M. Marchiori. The quest for correct information on the web: Hyper search engines. *COMPUTER NETWORKS AND ISDN SYSTEMS*, 29(8):1225–1235, 1997.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330, 1993.

S. Marti and H. Garcia-molina. Taxonomy of trust : Categorizing p2p reputation systems. *Internation Journal of Computer and Telecommunications Networking*, 50(August 2005):472–484, 2006.

D. Micol, O. Ferrández, R. Muñoz, and M. Palomar. DLSITE-2: Semantic Similarity Based on Syntactic Dependency Trees Applied to Textual Entailment. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 73–80, Rochester, NY, USA, 2007. Association for Computational Linguistics.

R. Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004. Interactive poster and demonstration sessions*, page 20, Morristown, NJ, USA, 2004. ACL.

R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004.

G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In *In Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 1–6, Bethlehem, PA USA, 2005. Lehigh University.

M. Najork. Web spam detection. In *Encyclopedia of Database Systems*, pages 3520–3523. Springer US, 2009. ISBN 978-0-387-35544-3, 978-0-387-39940-9.

M. A. Najork. Comparing the effectiveness of hits and salsa. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 157–164, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9.

M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, 2003.

V. Nikulin. Web-mining with Wilcoxon-based feature selection, ensembling and multiple binary classifiers. In *Proceedings of the ECML/PKDD*, 2010.

A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 83–92, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9.

F. J. Ortega, C. Macdonald, J. A. Troyano, and F. Cruz. Spam detection with a content-based random-walk algorithm. In *2nd International Workshop on Search and Mining User-generated Contents, at CIKM 2010*, Toronto, Ontario (Canada), 2010.

F. J. Ortega, J. A. Troyano, F. L. Cruz, and F. E. de Salamanca. Polaritytrust: Measuring trust and reputation in social networks. In *Fourth International Conference on Internet Technologies and Applications (ITA 11)*, Wrexham, North Wales, United Kingdom, 9 2011a.

F. J. Ortega, J. A. Troyano, F. J. Galán, C. G. Vallejo, and F. Cruz. Str: A graph-based tagging technique. *International Journal on Artificial Intelligence Tools*, 20(5):955–967, 2011b. ISSN 1793-6349. doi: 10.1142/S0218213011000437.

F. J. Ortega, J. A. Troyano, F. L. Cruz, and C. G. Vallejo. Polarityspam: Propagating content-based information through a web-graph to detect web-spam. *International Journal of Innovative Computing, Information and Control*, 8(5), May 2012. ISSN 1349-4198.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *World Wide Web Internet And Web Information Systems*, 54(66):1–17, 1999.

J. Pehcevski, A.-M. Vercoustre, and J. A. Thom. Exploiting locality of wikipedia links in entity ranking. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 258–269, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78645-7, 978-3-540-78645-0. URL `http://portal.acm.org/citation.cfm?id=1793274.1793308`.

E. Pianta, L. Bentivogli, and C. Girardi. Multiwordnet: developing an aligned multilingual database. In *Proceedings of the First International Conference on Global WordNet*, January 2002.

M. Platakis, D. Kotsakos, and D. Gunopulos. Discovering hot topics in the blogosphere. In *Proceedings of the 2nd Panhellenic Scientific Student Conference on Informatics, Related Technologies and Applications (EUREKA)*, pages 122–132, 2008.

S. P. Ponzetto and M. Strube. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the main conference on Human Language Technology Conference of the North American*

Chapter of the Association of Computational Linguistics -, pages 192–199, Morristown, NJ, USA, June 2006. Association for Computational Linguistics. doi: 10.3115/1220835.1220860.

A. Qamra, B. Tseng, and E. Y. Chang. Mining blog stories using community-based and temporal clustering. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 58–67, New York, NY, USA, 2006. ACM. ISBN 1-59593-433-2. URL `http://doi.acm.org/10.1145/1183614.1183627`.

R. L. T. Santos, C. Macdonald, and I. Ounis. How diverse are Web search results? In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Beijing, China, 2011. ACM.

H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proc. of International Conference on New Methods in Language Processing*, September 1994.

D. Sculley, C. Ave, and G. M. Wachman. Relaxed Online SVMs for Spam Filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 415–422, New York, NY, USA, 2007. ACM.

H.-C. Seo, H. Chung, H.-C. Rim, S. H. Myaeng, and S.-H. Kim. Unsupervised word sense disambiguation using wordnet relatives. *Computer Speech & Language*, 18(3):253 – 273, 2004. ISSN 0885-2308. doi: DOI:10.1016/j.csl.2004.05.004.

G. Shen, B. Gao, T.-y. Liu, G. Feng, S. Song, and H. Li. *Detecting Link Spam Using Temporal Information*. IEEE, Dec. 2006. ISBN 0-7695-2701-7. URL `http://portal.acm.org/citation.cfm?id=1193207.1193305`.

R. Sinha and R. Mihalcea. Unsupervised graph-based word sense disambiguation using measures of word semantic similarity. In *Proceedings of the International Conference on Semantic Computing*, pages 363–369, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2997-6.

L. Sokolov, T. Urvoy, and O. Ricard. Madspam consortium at the ecml/pkdd discovery challenge 2010. In *Proceedings of the ECML/PKDD*, 2010.

J. Troyano, F. Enríquez, F. Cruz, J. C. nete Valdeón, and F. J. Ortega. Improving the performance of a tagger generator in an information extraction application. *Journal of Universal Computer Science*, 13(9):1287–1299, 2007.

B. L. Tseng. Tomographic clustering to visualize blog communities as mountain views. In *WWW 2005 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*. ACM, 2005.

T. W. Valente and P. Pumpuang. Identifying opinion leaders to promote behavior change. *Health education & behavior : the official publication of the Society for Public Health Education*, 34(6):881–96, Dec. 2007. ISSN 1090-1981. doi: 10.1177/1090198106297855. URL `http://www.ncbi.nlm.nih.gov/pubmed/17602096`.

C. G. Vallejo, J. A. Troyano, and F. J. Ortega. Instancerank: Bringing order to datasets. *Pattern Recognition Letters*, 31(2):133 – 142, 2010. ISSN 0167-8655. doi: DOI:10.1016/j.patrec.2009.09.022.

P. Vossen. *Introduction to EuroWordNet*, pages 1–17. Kluwer Academic Publishers, Norwell, MA, USA, 1998. ISBN 0-7923-5295-5. URL `http://portal.acm.org/citation.cfm?id=314515.315174`.

J. R. Wicks and A. Greenwald. *More efficient parallel computation of pagerank*. ACM Press, New York, New York, USA, July 2007.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Elsevier, October 2000. ISBN 1558605525.

B. Wu, V. Goel, and B. D. Davison. Propagating trust and distrust to demote web spam. In *Proceedings of Models of Trust for the Web (MTW), a workshop at the 15th International World Wide Web Conference*, Edinburgh, Scotland, 2006a. CEUR-WS.org.

B. Wu, V. Goel, and B. D. Davison. *Topical TrustRank*. ACM Press, New York, New York, USA, May 2006b. ISBN 1595933239. URL `http://portal.acm.org/citation.cfm?id=1135777.1135792`.

L. Xiong and L. Liu. Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. *Knowledge and Data Engineering, IEEE Transactions on*, 16(7):843 – 857, july 2004. ISSN 1041-4347. doi: 10.1109/TKDE.2004.1318566.

B. Yu and M. P. Singh. Detecting deception in reputation management. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS '03, pages 73–80, New York, NY, USA, 2003. ACM. ISBN 1-58113-683-8. URL `http://doi.acm.org/10.1145/860575.860588`.

H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, and G. Attardi. Ranking very many typed entities on wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 1015–1018, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9. URL `http://doi.acm.org/10.1145/1321440.1321599`.

R. Zhou and K. Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460–473, Apr. 2007. ISSN 1045-9219. doi: 10.1109/TPDS.2007.1021.

X. Zhu, A. Goldberg, J. V. Gael, and D. Andrzejewski. Improving diversity in ranking using absorbing random walks. *HLT-NAACL*, pages 97–104, 2007.

Durante los últimos años la **Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN),** en su constante interés en la actualización y divulgación de la investigación en este campo, ha convocado la Edición Anual de los premios SEPLN a la Investigación en Procesamiento del Lenguaje Natural.

Este concurso admite trabajos monográficos de investigación de extensión variable que no hayan sido publicados con anterioridad, y escritos por un miembro de la SEPLN.

Siguiendo con esta dinámica, esta publicación recoge la undécima monografía correspondiente a la XII Edición de los mencionados premios SEPLN a la Investigación en Procesamiento del Lenguaje Natural.

ISBN 978-84-616-5169-6

sociedad española para el Procesamiento del Lenguaje Natural

www.sepln.org