Computational Intelligent Methods for Trusting in Social Networks

By

José David Núñez González

Submitted to the department of Computer Science and Artificial Intelligence in partial fulfillment of the requirements for the degree of Doctor of Philosophy

PhD Advisor:

Prof. Manuel Graña Romay
at The University of the Basque Country (UPV/EHU)

Euskal Herriko Unibertsitatea
Universidad del País Vasco
Donostia - San Sebastián
2016

Acknowledgments

Two lectures were enough for me trust Prof. Manuel Graña and to learn some general ideas about his research and projects. Then I decided to send him an email to ask for a day. During that meeting I told him that I wanted to do the PhD with him. I want to give him special thanks for give me the chance to make this trip by his side, full of new experiences. The truth is that without him this Thesis would not be possible.

I want to thank all my workmates and collegues at GIC who helped me in this task and all my friends that have always been there for me. Thanks to Borja and Oier to show me that there is no machine learning method to remove the noise you generate.

Finally, I want to thank my parents for their unending support in my life. Thanks.

José David Núñez González

Computational Intelligent Methods for Trusting in Social Networks

by
José David Núñez González

Submitted to the Department of Computer Science and Artificial Intelligence on January 11th, 2016, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

This Thesis covers three research lines of Social Networks. The first proposed reseach line is related with Trust. Different ways of feature extraction are proposed for Trust Prediction comparing results with classic methods. The problem of bad balanced datasets is covered in this work. The second proposed reseach line is related with Recommendation Systems. Two experiments are proposed in this work. The first experiment is about recipe generation with a bread machine. The second experiment is about product generation based on rating given by users. The third research line is related with Influence Maximization. In this work a new heuristic method is proposed to give the minimal set of nodes that maximizes the influence of the network.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

This introductory chapter aims to set the stage for the entire Thesis, providing some of the motivations for the research work performed, and some background ideas on each of the topics covered by the Thesis. We also try to clarify the contributions of the Thesis. The Chapter is organized as follows: Section 1.1 gives a brief definition of social systems and associate graphs. Section 1.2 gives some reasoned motivation as well as a more personal motivation for the works related in the Thesis. Section 1.3 talks about Trust prediction systems. Section 1.4 refers a general view on reconmmendation systems. Section 1.5 gives the background on influence maximization systems. Section 1.6 summarizes the Thesis goals, which have been uncovered along the years. Section 1.7 details achieved contributions. Section 1.7.1 enumerates the publications achieved. Section 1.8 describes the Thesis organization and content.

1.1 Brief definition

A social network is a social structure made up of a set of actors that are related according to some criterion. Actors are represented as nodes and relationships among them are represented as lines connecting them. Thus, the social network analysis examines the social structure using Graph Theory and identifying the entities as "nodes" or "vertices" and relationships as "links" or "edges". We can define a social network like a graph $G = (V,E)$ being V the set of nodes and E the set of edges.

1.2 Motivation

Trust, reputation and recommendation systems have become a topic of growing interest for social network community, and are essential ingredients of web-based multi-agent systems. Today it is common to find social network webservices that propose recommendations to users suggesting new contacts, web pages, or products. Several social networks give users the chance to rate other users or items. Thus, the knowledge of the system is built from the ratings given by users. This knowledge might be used by the system to make recommendations. Before giving recommendations it is important to know how to choose the information to give the recommendation, in other words, who should the system Trust to collect information given by other users of the network. In this way, this Thesis contributes with methods towards Trust prediction computation as well as recommendation systems. Another way to enrich the knowledge in a recommendation systems is the obtained feedback by the user. When a recommendation is given to the user and this one leaves a positive rating, this fact allows the system to know that this recommendation corresponds to a good one. Another interesting point related with recommendation systems and social networks is to detect those nodes that are able to influent other nodes of the network, in other words, influence maximization.

Following these motivations, this Thesis has been developed along three research lines as is shown in Figure 1.1. The first research line is related with Trust prediction, the second research line is related with Recommendation Systems and the third research line is related with Influence Maximization. Next Sections 1.3, 1.4 and 1.5 introduce those background ideas for these research lines.

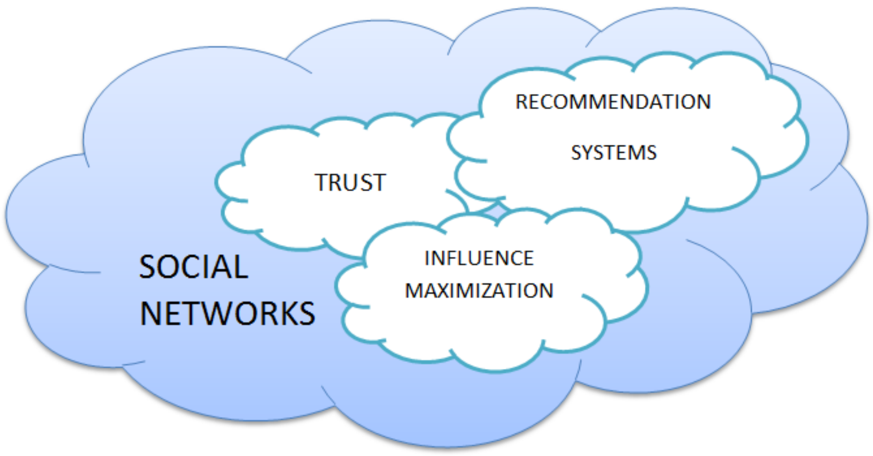


Figure 1.1: Thesis research areas

The project SandS `http://www.sands-project.eu/` has been the mainframe where the works of this Thesis have been developed, so that many of the research lines are due to the participation in this project. The SandS project aimed to develop some kind of social intelligent system focused on the household appliance control. Therefore, one of the first issues that were considered was that of recommendation preparation, so that control recipes could be considered as the product of social recommendation. The issue of Trust appears when you consider the different value of recommendations from diverse system users, due to a diversity of factors, such as geographic diversity. How to predict which degree of Trust we can deposit in one user before any interaction takes place? From this question, we developed the works on Trust prediction. Finally, for a company trying to maximize its influence in the ecosystem developed from SandS ideas we had the push to explore methods for Influence Maximization.

1.3 Trust

A social network is a way of representing a social structure by a graph. If two social actors are related according to some criteria, then an edge connects the nodes representing these elements is constructed. Examples of related work with social networks could be [68], who proposes a survey on online social networks (OSN) and [74] propose a novel Trust framework to address the issue of "Can Alice Trust Bob on a service?" in large online social networks (OSN). They propose the SWTrust framework to discover short Trusted paths based on Trusted acquaintance chains, and generate Trusted graphs for online social networks. In Social Networks, Trust is built from experience along a feedback process.Trust has been a traditional subject of study in four different areas of knowledge, namely social psychology [1], philosophy, economics and market research [17], however it is increaslingly becoming a subject of research in technological domains, such as ad hoc networks [2, 26, 31], Medical Sensor Networks [66], Industrial Digital Ecosystems [41], and e-commerce [98]. There have been some efforts to produce mathematical definitions of Trust [135, 44], however intuitive informal definitions, such as "the degree of subjective belief about the behaviors of (information from) a particular entity" [40], or "the expectation that a service will be provided or a commitment will be fulfilled" [69], are convenient for the purposes of this paper. Trust research can be organized [8] in four major areas: policy-based Trust, reputation based Trust, general models of Trust, and Trust information resources, related with the following applications: networking, semantic web, computational models, game theory and agents, software engineering and information resources. This Thesis focus on the prediction of Trust values in the case of no previous interaction between users.

1.4 Recommendation Systems

A direct application of computational intelligence techniques is found in Recommendation Systems. Recommendations are based on ratings that users give about other users or items. In this way, a Recommendation System will suggest new items based on the Web of Trust (WoT) built from users Trust and product ratings provided by the users in the system. If a user x Trust user y and user y gives a positive rating to an item i, the Recommendation System will suggest the item i to the user u. The more better ratings an item collects from users, the more probability to be chosen from the Recommendation System to suggest the item to users through a collaborative filtering. Collaborative because a lot of users take part rating items and filtering because only the best rated items are chosen to be offered to users. In contrast, another way for the system to suggest an item is via collaborative sanctioning. In this case, users may give negative ratings to items. When an item collects negative ratings, the system remove it as candidate for suggestion to other users. There are many ways to build a collaborative filtering system. They can be memory-based filtering, model-based filtering, and hybrids. Memory-based filtering uses user rating data to compute the similarity between users or items. Two users are similar when they have similar profiles, tastes, interest... According to this, a way to compute similarity is getting the K-nearest neighborhood in order to do prediction or recommendation. Content-based filtering focus on ratings of items. The rating matrix is used by the system to learn how users give their product ratings. Based on given ratings, system will look for items that are similar to items that the user rated positively before. When the WoT is available, the system may take Trust into account in order to select items that are positively rated from user who the target user Trust whom and remove those items that are rated by users distrusted by the target user.

Recommender systems [18] are taking a prominent role in the interaction with the virtual world incorporated by the myriad of web services used on a daily basis by the common people. Early realizations included forms of collaborative filtering, however the advent of the Internet of Things will allow to use implicit, local and personal information gathered by the surrounding environment of smart objects. Recommender Systems are currently being applied in many different domains. Some example applications are: intelligent tourism [20], movie suggestions [21], electronic marketplaces [32], and university library research [126]. The State of the Art techniques involved in recommender systems deal with the problem of accurate representation and management of the user profile, requiring computational tools from many fields of Artificial Intelligence, such as Multi-agent systems, advanced optimization techniques, clustering of the users data to detect communities, and advanced knowledge representation and reasoning for the management of uncertainty. Collaborative filtering social recommender systems [138] use social network information as additional input for improved recommendation accuracy. They define two categories of CF-based social recommender systems: matrix factorization based approaches and neighborhood based approaches, providing a comparison among algorithms. Recommender Systems are common in e-commerce for making personalized marketing. On the other hand, Online Review Systems (ORS) allow users to provide reviews of products and thus become a user-oriented Recommender System. To help the user to navigate the reviews, the ORS provides the possibility to state Trust scores on the reviews, so that reviewers with more positive Trust scores will merit more attention. The issue, then, is how the observed Trust scores given by other users may influence the user, and may serve to predict his own Trust value. Several other works are found in the Recommender Systems literature. For instance, a Colaborative Filtering with Markoviam random walk for recommendation systems proposed by [118] on Epinions and Movie-Lens datasets. Other example is the work proposed by [104] also with Colaborative Filtering for recommendation. Another one is proposed by [105], a recommender system to see propagation of Trust in anonymous social networks.

1.5 Influence Maximization

Influence Maximization [38, 77] is stated as the problem of finding the minimal subset of influential nodes (K-seed nodes) with maximal influence, i.e. that affect the largest number of nodes in the network, where influence is computed by propagation in the network according to a spread model. The K-seed nodes are the initially active nodes spreading their influence according to any of the propagation models discussed above. The most straightforward application is in marketing, when a new product hits the market, a company may want to select the smallest group of (most influential) seed customers to provide them the product for free in order to boost its popularity by propagating it in their social network by word-of-mouth [54]. Additionally, influence maximization has been applied to design negotiation strategies addressing persuasion to the most influent agents [103], and to worm propagation containment in ad networks of smartphones [111].

The analysis of influence propagation through social media started from the consideration of phenomena such as mobs, riots or strikes [64] as pure physical phenomena, stripped out of psychological considerations. That is, the quantitative model considers that individual decisions are taken as the fruit of social pressure defined by social interactions. The same model applies to propagation of innovations, rumours, and advertising [54], so the topic become naturally part of the marketing research area. The research question was to determine the appropriate balance between marketing efforts and word-of-mouth propagation through personal social networks defined by strong and weak links. Cellular automata formalism allowed to build computational models to explore such questions. The two basic spread models of influence propagation are the Independent Cascade model

(ICM) [54], and the Linear Threshold model (LTM) [64].

1.6 Thesis goals

The main goals of this Thesis can be stated as follows:

1. Study of basic social mechanisms such as the propagation of Trust and influence, and the generation of innovations from social interactions. Each of these phenomena have different properties and computational solutions. Hence, they give way to diverse tracks of the Thesis works.
   (a) Regarding Trust, we are interested in the prediction of Trust when there is no preliminar interaction information. Then, the problem is stated as a classification problem.
   (b) The Influence Maximization is a combinatorial optimization problem, that has been tackled by heuristic approaches.
   (c) Social generation of innovation is an emergent phenomena, we are concerned with a specific instance in the framework of the SandS european project, for the generation of new solutions (recipes) for new problems (tasks).
   (d) Recommendation generation based on social knowledge. These recommendations encompase product rating.
2. Study of the applications of Machine Learning algorithms in social networks, for Trust prediction, recommendation systems, and social computing.
3. Study of Computational Intelligence algorithms for social networks, i.e. for Influence Maximization.
4. An operational goal, wich is required for the Thesis works is the creation of a computational substrate for experimentation, and validation given by real and synthetic databases.

1.7 Thesis contributions

Pursuing the above goals, we have achieved several contributions. The main scientific results and contributions from this Thesis are the following:

1. We have formalized a concept of reputation for Trust prediction that allows the application of machine learning algorithms. The reputation features are low dimension and capture the essential information for Trust prediction, as demonstrated empirically.

2. We have provided experimental results of Trust prediction over benchmark databases which prove our approach comparable to state-of-the-art approaches.

3. We have proposed a heuristic for Influence Maximization that improves over the classical greedy approach, and other meta-heuristics, such as Genetic Algorithms, Simulated Annealing, and Harmony Search.

4. A specific formulation of Harmony Search for Influence Maximization has been proposed.

5. In the field of social innovation we have introduced the concept of recipe generation, in the context of household appliances, and we have provided a mechanism for the intelligent recommendation of new optimal recipes for new tasks, based on evolutionary computation and machine learning.

6. For product recommendation we propose two feature extraction algorithms, the first one is based on the existing Web of Trust and the other one is based on the similarity between users build from the product rating matrix. The product ratings from Trusted/similar users are the basis for the target user rating prediction by machine learning.

1.7.1 Publications achieved

- Graña M., Nuñez-Gonzalez J.D., Ozaeta L., Kamińska-Chuchmała A., (2015), “Experiments of Trust Prediction in Social Networks by Artificial Neural Networks”. Cybernetics and Systems. Vol 46, N 1-2, pp 19-34. Taylor and Francis.

- Nuñez-Gonzalez J.D., Graña M., Apolloni B., (2015), “Reputation features for Trust prediction in social networks”. Neurocomputing. Vol 166, pp 1-7. Elsevier.

- Nuñez-Gonzalez J.D., Ayerdi B., Graña M., Wozniak M., (2015) “A new heuristic for Influence Maximization in Social Networks”. Logic Journal of the IGPL. Oxford. (submitted)

- Graña M., Nuñez-Gonzalez J.D., (2015), “An instance of social intelligence in the internet of things: bread making recipe recommendation by ELM Regression”. Hybrid Artificial Intelligent Systems. Vol. 9121. pp 16-25. Springer.

- Nuñez-Gonzalez J.D., Graña M., (2015), “Graph-Based Learning on Sparse Data for Recommendation Systems in Social Networks”. Bioinspired Computation in Artificial Systems. Vol. 9108. pp 61-68. Springer.

- Nuñez-Gonzalez J.D., Graña M., (2014), “On the effect of high order reputation information on Trust Prediction in Wikipedia’s Vote Network”. European Network Intelligence Conference (ENIC), 2014. pp 59-62. IEEE.
- Nuñez-Gonzalez J.D., Graña M., (2014), “Experiments on Trust Prediction Based on Reputation Features”. International Joint Conference SOCO’14-CISIS’14-ICEUTE’14. Vol. 299. pp 367-374. Springer.
- Nuñez-Gonzalez J.D., Graña M., (2014), “ELM predicting Trust from reputation in a social network of reviewers”. Extreme Learning Machines 2013: Algorithms and Applications. Vol. 16. pp 179-187. Springer.
- Graña M., Nuñez-Gonzalez J.D., Apolloni B., (2013), “A discussion on Trust requirements for a social network of eahoukers”. Hybrid Artificial Intelligent Systems. Vol. 8073. pp 540-547. Springer.

1.8 Thesis organization

The remaining the Thesis is organized as follow:

- Chapter 2 shows the State of the Art of the three research lines. The aim of this chapter is to set the stage for the various applications. Therefore, we cover in detail the literature for each aspect, including some background definitions that are used throught the Thesis.
- Experimentation on Trust prediction is reported in Chapter 3. We define and test here our reputation based features, some of them consisting in some summary statistics in order to show the preponderacy of the selected topic
- Chapter 4 deals with recipe generation. Therefore, it includes some specific background and the specification of the computational pipeline involving a direct model for satisfaction evaluation and an inverse model for innovation generation. Experimental results over a synthetic dataset demonstrates the approach.
- Chapter 5 deals with product recommendation systems. Specifically, the main contribution in this Chapter is the definition of two feature extraction processes for rating prediction by machine learning. One based on the WoT of the recommendation system, the other based on user similarity extracted from the rating matrix.
- Chapter 6 deals with computational approaches to Influence Maximization. A new approach is contributed and empirical experimentation shows its improvements.

- Chapter 7 gathers the conclusions from each of the research lines, thus it concludes this Thesis proposing new future work.
- Appendix A describes real and synthetic databases used for experimentation.
- Appendix B gathers diverse computational algorithms used in the Thesis which are not direct contribution of our work, but have been instrumental in the realization of experiments.

Chapter 2

State of the art

In this Chapter we gather descriptions of the state of the art of the three research lines mentioned in the introductory Chapter. The Chapter is structured as folows: Section 2.1 gives some basic definitions that will be used all along the Thesis. Section 2.2 presents a view of the state of the art on trust prediction systems, the subject of Chapter 3. Section 2.3 introduces the problem of social computing for recipe generation, the subject of Chapter 4. Section 2.4 gives some background on recommender systems, the subject of Chapter 5. Section 2.5 gives background information on Influence Maximization, the subject of Chapter 6.

2.1 Some basic definitions

Let us recall some definitions. A graph is described by a collection V of n vertices (or nodes) and m edges E that is denoted as $G(V,E)$. An adjacency matrix A with $n \times n$ size represents the nodes of the graph and existing links. Thus, an element $a_{ij} \in A$ from the adjacency matrix will be $a_{ij} = 1$ if exists an edge that links both vertices i and j. Sometimes edges have an associated weight, which may be a value or set of values. The weight matrix W of size $n \times n$ contains those weights $W = [w_{ij}]_{i,j=1}^{n}; w_{ij} \in \mathbb{R}$. The degree of a vertex is the number of edges linking it with other vertices. The matrix representation of the degrees is $D = [d_{ij}]_{i,j=1}^{n}; d_{ij} \in \mathbb{R}$. The matrix L that represents the graph Laplacian is defined as $L = D - A$. There are some well known measures of graph complexity and structure [106] such as degree centrality, eigenvector centrality, katz centrality, pagerank, closeness centrality and betweenness inter alia. Eigenvector centrality [19] is defined as $x_i = \kappa_1^{-1} \sum_j A_{ij} x_j$ where κ_i are the eigenvalues of A and κ_1 is the largest of them, A_{ij} is an element of the adjacency matrix and x is the centrality of vertex j. So, the centrality x_i of vertex i is proportional to the sum of the centralities of i's neighbors.

2.2 Related works on Trust

Trust is a central issue for users seeking online reliable interaction, whether it consists of information to make some decision, or simply social interaction [76]. However, user specified trust relations are very sparse, only a very small fraction of users do provide explicit trust information. Usually, social webservices allow users to specify publicly or to keep track in private of the trust or distrust on another users, thus creating to a Web of Trust (WoT) embedded in the social service. Examples of trust-aware services are online recommendation systems, or crowd-sourcing systems, such as wikipedia. For instance, trust has been proposed to filter out controversial reviews [132], and in general to improve collaborative filtering in social networks [30, 36].

2.2.1 Trust Properties

Several Trust Properties are been defined which are worth reviewing before talking on metrics and models, because they set the stage of the kind of mathematical model. Transitivity, asymmetry [52] , and personalization-subjectivity [52, 31] as the three main Trust properties that are relevant to algorithm development. Dynamicity, and context-dependency [31], reflexivity [3, 44], non antisymmetry, time-based aging and distance-based aging [3] may be considered in some context.

Transitivity: in simplified form, mathematical transitivity means that if $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$. The Trust relation does not support transitivity, quoting [122]: "Alice may Trust Bob about movies, but not Trust him at all to recommend other people whose opinion about movies is worth considering or not Trust other people that Bob recommended as much as she Trusts Bob". In fact, Trust diminishes [121, 140] as the chain of Trust recommendations increases in some exponential law of the length of the Trust path.

Asymmetry: Trust does not have to be a symmetrical concept, in other words, two entities need not have the same degree of Trust in each other. A typical example is that in a hierarchical environment the degree of Trust between the supervisor and the employee is different [2, 52].

Personalization-subjectivity: for [52] Trust is inherently a personal opinion. Two entities A and B could have a different opinion about the Trustworthiness of another entity C. For [31] an entity A could Trust another entity B with a certain degree of Trust.

Dinamicity: Trust should be expressed as a continuous variable, rather than as a binary or even discrete-valued entity. A continuous valued variable can represent uncertainty better than a binary variable [2, 31].

Context-dependency: An entity can Trust other entity for some tasks but not for other tasks [15]. For example, a node A from a network can Trust another node B to ask for authentication tasks but not for key management tasks.

Reflexitivity: considering internal actions, if agents Trust themselves we have reflexivity of the Trust relation [44], which may be stated as the fact that the Trust value of A on itself for any context is 1 [3].

Non-Antisymmetry: "If A Trust B and B Trust A, that does not indicate that A=B" [3].

Time-based-aging: "The Trust value of A on B for a specific context C decreases with the passage of time". The Trust on a piece of information obtained at t i time will decrease with the passage of time because in t i+1 some event may change the value of the associated objects. New pieces of information must be more Trustable than the older ones [3].

Distance-based-aging: "If node A collects Trust values about B from other nodes in the network (recommendation), the Trust values collected from closer nodes should be counted with more weight compared to the values collected from distant nodes" [3].

2.2.2 Trust Metrics

Trust propagation and computing models are essentially directed graphs [127] where nodes represent entities and edges trust relations labeled by some trust metric values. Trust management may be centralized (when a central trusted arbiter gives trust evaluations of the partners), or decentralized (where users are responsible for the calculation of their own trust values for any target). It can also be distinguished between reactive computation, that calculate trust values when explicitly required, and proactive, which compute continuously the trust values of the peers, aiming to avoid delay in trust decisions. Trust computing must be resilient to attacks, which may consist in node attacks giving arbitrary opinions on a compromised node, or edge attacks inserting false edges in the network. Adding positive and negative evidence to the trust computation allowing for an accurate and flexible model. In communication networks, trust computing must be built at the routing and protocol levels, as the basis for all the upper layers, quoting: "important issues that should be considered by designers of trust metrics" [140]. Then, there is a part of an example in Ad Hoc Networks using the given taxonomy. The second part shows a selection of trust metrics proposed by [140], giving five types of metrics based on the quoted references:

- Binary State Metric. This metric uses binary states 0 and 1 to express trust and distrust, which in some systems, is only considered as vote or observation. Therefore, binary state opinions are the building blocks of more abstract trust computing.

- Discrete Scale Metric. Allows to choose an option in a given range. Once the choice done, we can convert it to a quantity value, usually discrete.

- Probabilistic Metric usually represents the probability of a evaluated target participant performing actions as the evaluating participant expects.

- Hybrid or multi-metric trust, i.e. to "use multiple metrics as a trust tuple to express more comprehensive trust". For example, [127] uses trust and confidence to form an opinion space.

- Negative Values: Negative trust value can be interpreted as distrust. "Some researches state the necessary of negative trust value to express bad impression. However, introducing negative trust value also brings in vulnerability. Because generally negative of negative will produce positive result, malicious participants can employ this feature to manipulate trust values in order to promote their companies trust value" [140].

2.2.3 Trust Models

There are many trust models in the literature. For example, [127, 26, 134, 25, 24] propose different trust models. We select two models. Marsh's Model Proposed by Marsh in 1994 in his Phd, it is considered the first prominent, comprehensive, formal, computational model of trust [8, 14]. It consists in a set of variables and ways to combine them arriving to a normalized value in the range $[-1, 1]$. Marsh identified three types of trust: basic, over all contexts; general, between two people and all their contexts occurring together; and situational, between two people in a specific context. Bharadwaj's Model A fuzzy computational model for trust and reputation concepts is proposed in [14]. Quoting him, "The most appropriate property to define the symmetric part is the reciprocity while the partner's experience defines the asymmetric part. The reciprocity is the mutual favor or revenge and therefore to model it, we need to find the agreement (both individuals are satisfied or unsatisfied) and disagreement (only one of them is unsatisfied) between two partners. To do so, we can define two fuzzy subsets on each partner's ratings (universe of discourse), namely satisfied and unsatisfied. The membership values of satisfied and unsatisfied fuzzy subsets for a given encounter always sum up to one, for example, 70% satisfaction indicates 30% dissatisfaction. From these two fuzzy sets we can find the agreement and disagreement between the two partners".

2.2.4 Applications

Trust is a pervasive concern in human and computational interactions [15, 117]. We must trust the services we receive and we rely on for our work and daily life, ranging from the use of cars and transports to the use of internet services, or the interaction between computational agents performing searches or other delegated tasks. Trust in automation [70] has been identified as a major concern in the development of human centered computing, proposing active evaluation of trust strategies to correct unjustified trust or mistrust, and to assess their consequences. In general terms the trusters face some degree of risk when they decide to accept the outcome of the trustee. Trust management [95, 94] is related with the prediction of the expected risk or the affordable level of trust on the basis of all available information. When the source of information is the opinion of other agents, the witnesses, the system is based on Reputation. In the field of distributed systems, such as Ad Hoc communication networks, Social Networks, Online Review Systems and Recommender Systems, the issue of managing Trust is critical for the function of the system. Recommender Systems are common in e-commerce for making personalized marketing. On the other hand, Online Review Systems (ORS) allow users to provide reviews of products and thus become a user-oriented Recommender System. To help the user to navigate the reviews, the ORS provides the possibility to state trust scores on the reviews, so that reviewers with more positive trust scores will merit more attention. The issue, then, is how the observed trust scores given by other users may influence the user, and may serve to predict his own trust value. The truster makes decisions based on trust built on the trustee part. Positive or negative results will maintain, increase or decrease the trust value. The cold start problem rises when there is no previous experience about the trustee behavior. Then, there is a need to predict the trust value from indirect information, that is, from reputation information obtained from third party witness. Trust prediction can be formulated as a classification problem, where feature vectors are computed from the reputation information extracted from the network. Although the trust relation between users is not transitive [31, 52, 122], reputation can be accepted as the best guess information that can be used to predict the trustworthiness of a trustee. Trust prediction is becoming a central issue in many computational problems involving the interaction of agents through online services. These agents can be humans or autonomous computational entities. Much of the Semantic Web and the Internet of Things will be supported on trusted interactions [8]. Trust can be built from a history of interactions between a pair of agents, but still the question of the cold start remains. What is the basic attitude of a truster regarding a trustee when there is no previous history of interactions? It can be inferred indirectly from user attributes, i.e. following some homophily reasoning (alike users like similar things), or it can be predicted from the trustee reputation.

2.2.5 Supervised approaches

Supervised systems perform feature extraction to train binary classifiers (e.g. Random Forest, SVM [107, 87, 81]) for trust prediction. Feature extraction can include ancillary information [87, 81] trying to cover all possible influences between users leading to trust. Specifically, in recommendations systems the information about ratings and review evaluations is used to complement the WoT graph features. Some works [107] elaborate on basic philosophical arguments (i.e. trust antecedent framework for ability, benevolence and integrity) to derive feature extraction procedures in recommendation systems. The classification data is strongly imbalanced, so that research into the effect of strategies to cope with this issue is an open research problem. It is also unclear whether the ancillary information is useful or a source of noise in the classification.

2.2.6 Unsupervised approaches

The unsupervised approaches are either graph based methods of trust propagation or try to derive user similarity measures from ancillary information. An instance of graph based trust prediction [4] applies a capacity-first maximal flow algorithm to identify strong paths leading to trusted user groups. Also, [12] performs graph mining to detect patterns that allow to derive rules for the completion of the ego-graph of one user with trusted users. Homophily is proposed [123] to regularize previous unsupervised approaches, such as Pearson Correlation Coefficient, Jaccard coefficient or Matrix Factorization. Low rank matrix factorization searches for small dimension decompositions of the trust graph adjacency matrix, in fact looking for compact (trust) communities with early applications in collaborative filtering [80]. Global and local information effect in matrix factorization approaches is examined in [124]. Generative models [33] try to discover the underlying communities by latent variable analysis. Finally, the Ant Colony Optimization approach is used in [13] to perform trust propagation, including the popularity modeling by pheromone accumulation.

2.2.7 Imbalanced classification problems

Imbalanced classification problems appear when some classes are more frequent than others, resulting in datasets with significantly more samples for some classes. Trust prediction often fails in this category of imbalanced problems, if we consider the negative trust class which is much less frequent than the positive or the neutral, because people tendency is not to explicit the negative trust. Conventional classifier building approaches have the problem of bias towards the most frequent classes. In a Bayesian formulation, classifiers guided by the maximization of the overall accuracy are biased towards classes with the higher *a priori* probability. This implies that minority classes are underestimated. In many real life situations the minority classes are the interesting ones, such as in target detection, or anomaly detection problems.

Two ways of dealing with the issue of class imbalance have been developed in the machine learning community. One assigns different costs to training examples. Assuming that the minority class has a greater cost associated to error committed to it allows to drive the learning process towards its more accurate modeling. The other pre-processes the original dataset, either by over-sampling the minority class and/or under-sampling the majority class, such as the SMOTE process that will presented below [28]. Various representatives of these approaches have been evaluated in [91, 89] against a collection of benchmarking datasets, reaching the conclusion that no method prevails against the others. In other words, no statistically significant differences were found between methods when taking into account the complete collection of benchmarking problems. However new lines of work arise from the study of the intrinsic properties of the datasets [89], or the characteristics of the validation process [90]. Empirical studies on specific domains, such as software quality, are curried out nowadays [116].

A recent cost sensitive approach is [82] proposing a new cost-sensitive ensemble of decision trees. The approach performs an evolutive search for the optimal classifier selection and fusion. Base classifiers are cost-sensitive trees, performing local sequential search at each node. Another evolutionary algorithm approach to develop cost sensitive Fuzzy rule classifiers is presented in [42] following a Pittsburg approach, where each rule corresponds to an individual. Boosted SVM [141] are ensembles of SVM which are trained in a procedure alternating two steps: solving the optimal SVM problem for fixed weights of the data samples, and updating these weights in an external loop. Along these lines, [99] propose the adjustment of the F-measure for the evaluation of the classification results while performing fine tuning of the kernel scaling in SVM based approaches. On the other hand, works on data resampling are less abundant. Recent evolutionary sampling techniques have been proposed [47] in order to select the best representatives for generalized sample representation by hyper-rectangles. A heuristic method for selection of balanced datasets minimizing the majority class while maximizing the minority class is provided in [133].

2.3 Social computing for Recipe Generation

There is an emerging view of social networks as information and knowledge repository at the service of the social agents to solve specific problems or to learn procedures relative to a shared domain of problems. Besides popular web service implementations, social networks have shown to be useful to spread educational innovations. *Social computing* [130] may be defined as the result of social interaction when it is oriented towards information processing or decision making. Preliminary elaborations towards a taxonomy of social computing systems [61] include the term *subconscious intelligent social computing* [59, 60, 61, 62] characterized by some hidden layer of intelligent processes that helps to produce innovative solutions to the problems posed by the social players. The social player asks for the solution of a problem, i.e. how to wash my laundry composed of items with some specific dirtiness and according to my preferences? The social framework provides solutions either from previous reported experiences of other social players or as innovation generated by the hidden intelligent layer.

Intuitive description of a system In the framework of the Social and Smart (SandS) project[1] users are called eahoukers [6]. There are two repositories of knowledge in the SandS Social Network containing tasks to be carried on the appliances and the recipes solving them. When a user requires a task to be performed (blue dashed arrows) there are two possible situations, either the recipe solving the task is known or not. In the second case, the so called Networked Intelligence incorporating the hidden intelligent layer is in charge to produce a new recipe to solve the unknown task. In other words, it is in charge of achieving innovation (green arrow). The recipe found either way is returned to the appliance (black arrows). In the specific case of the breadmaker appliance, we do not have a proper task specification, because it is always the same. In some way it can be said that the specification of the desired satisfaction parameters, i.e. baking, crustines, softness, fragance, are the task specification. So, real life experiments give us pairs of (recipe, satisfaction) vector values, which are always the result of setting the breadmaker parameters and measuring the resulting bread. There is no way in real life to produce the data in the inverse way, setting the user satisfaction to see what is the resulting recipe. Therefore, this inverse map must be estimated from the data gathered in the direct experiments.

The SandS system simplified description introduces the fundamental questions that we are tackling in this paper by designing a prototype recommender system for a specific appliance, the breadmaker, and its validation.

[1] `http://www.sands-project.eu/`

- The first question is: how to build a recipe recommendation from the specification of the user satisfaction? That problem is addressed by building an Extreme Learning Machine[2] [73, 72] from the experimental data that implements the inverse mapping.
- The second question is: how to decide that we need innovation? In other words, the inverse model may produce a recipe which in fact is far from solving the problem, so we need to create some new recipe outside the knowledge embedded in the mappings. How we detect that situation? The answer lies in the application of the direct mapping from recipes to satisfaction, and measuring the distance between the predicted satisfaction vector and the one specified by the user.
- The third question is: how to perform innovation? We need to build some generative process that achieves to create new recipes optimizing expected satisfaction. The solution proposed in [100] is a stochastic search process guided by the learned user satisfaction model, specifically an Evolutionary Strategy approach [55].

A critical issue is the lack of real life data supporting the design and validation of this architecture. The SandS project failed to build the framework that would allow users to experience this social interaction, and no actual data was generated before project completion. So we have to resort to synthetic data in Chapter 4 in order to show the intended workings of a recipe recommendation system.

2.4 Product Recommendation Systems

Recommender systems [18] are taking a prominent role in the interaction with the virtual world incorporated by the miriad of webservices used on a daily basis by the common people. Early realizations included forms of collaborative filtering, however the advent of the Internet of Things will allow to use implicit, local and personal information gathered by the sorrounding environment of smart objects. Recommender Systems are currently being applied in many different domains. Some example applications are: intelligent tourism [20], movie suggestions [21], electronic marketplaces [32], and university library research [126]. The State of the Art techniques involved in recommender systems deal with the problem of accurate representation and management of the user profile, requiring computational tools from many fields of Artificial Intelligence, such as Multi-agent systems, advanced optimization techniques, clustering of the users data to detect communities, and advanced knowledge representation and reasoning for the management of

[2]Source-code: http://www.ntu.edu.sg/home/egbhuang/elm_codes.html

uncertainty. Collaborative filtering social recommender systems [138] use social network information as additional input for improved recommendation accuracy, falling into two categories of systems: matrix factorization based approaches, and neighborhood based approaches.

2.4.1 Applications of learning on graphs

Graph-based learning is already being used in other contexts. For instance, [35] proposes graph-based semisupervised learning graph classifier based on kernel smoothing. A Sequential Predictions Algorithm (SPA) are used as a graph-based algorithm which propagates labels across vertices. The work in [134] gives a graph-based multiprototype competitive learning and its applications to solve partitioning nonlinearly separable datasets. On the other hand, [136] proposes a clustering-based graph Laplacian framework for value function approximation in reinforcement learning by subsampling in Markov Decision Processes (MDPs) with continuous state spaces. In patter recognition [23] proposes learning graph matching. In graph matching, patterns are modeled as graphs and pattern recognition amounts to finding a correspondence between the nodes of different graphs. The used method for learning graph matching is bistochastic normalization, a state-of-the-art quadratic assignment relaxation algorithm. Graph-based semisupervised learning (GSSL) as a paradigm for modeling the manifold structures that may exist in massive data sources in highdimensional spaces is proposed in [88]. Sparse canonical correlation analysis [9, 10] has been used in neurosciences to find the data projection which is best correlated with some measure of interest, in order to find image biomarkers. It performs Singular Value Descomposition (SVD) for dimensionality reduction strategy. The whole approach is called eigenanatomy in the context of neuroimaging data, identifying generalizable, structural MRI-derived cortical networks that relate to distinct categories of cognition.

2.4.2 Sparse data and classification

A sparse matrix has most of its elements equal to zero. When dealing with large matrices, traditional methods to store the array in memory of a computer or for solving systems of linear equations need a lot of memory and processing time. Specific algorithms for sparse matrices have been designed and are available, i.e. the Matlab implementations[3]. In sparse representation (SR) a signal can be approximated by a sparse linear combination of dictionary atoms. It is formulated as follows:

[3]https://sites.google.com/site/sparsereptool/

$$\mathbf{b} = x_1\mathbf{a}_1 + ... + x_k\mathbf{a}_k + \varepsilon = A\mathbf{x} + \varepsilon,$$

where $A = [a_1, ..., a_k]$ is called dictionary, $\mathbf{a}_i$ is a dictionary atom, $\mathbf{x}$ is a sparse coefficient vector, and ε is an random error term. The model parameters are $A, \mathbf{x}, k$. Finding the sparse representation implies solving the problem of finding the linear combination of atoms in a dictionary with minimal size and reconstruction error. Each signal can be reconstructed by only one linear combination of atoms, and the number of non zero coefficients must be as small as possible. Thus, finding the representation corresponds to the following optimization problem:

$$\min_{\mathbf{x}} \frac{1}{2} \parallel \mathbf{b} - A\mathbf{x} \parallel_2^2 + \lambda^T \|\mathbf{x}\|_1 \, \text{s.t.} \mathbf{x} \geq 0.$$

2.4.3 Applications of sparse representations

Several works focus their interest in solving sparsification problems. We summarize some works about it. For instance, [101] relates sparse coding and Multilayer Perceptron (MLP) by converting sparse code into convenient vectors for MLP input for classification of any sparse signals. In [11] a Bayesian framework 3-D human pose estimation from monocular images to get a posterior distribution for the sparse codes and the dictionaries from labeled training data is proposed. On the other hand, [16] gives a method that constructs a series of sparse linear SVMs to generate linear models in order to reduce data dimensionality. A survey of algorithms and results to induce grammars from sparse data sets can be found in[34]. Models of biological neurons to create sparse representations with true zeros for naturally sparse data are given in [51]. The large-scale matrix factorization problem is solved via optimization algorithm [97], based on stochastic approximations. Transfer learning for image classification with sparse prototype representations is described in [113]. Finally, [128] combines a general Bayesian framework with Relevance Vector Machines in order to obtain sparse solutions to regression and classication tasks utilising linear models of the parameters.

2.5 Influence Maximization

Influence Maximization was proven to have NP-complete computational complexity in [77], for both the Independent Cascade model (ICM) [54], and the Linear Threshold model (LTM) [64] propagation models. In fact, the computational cost of spreading influence is #P-hard, while the combinatorial search for the mini-max set of nodes in equivalent to NP-hard problems. They also show that the greedy search solution is guaranteed to be at worst within $(1 - {}^1\!/_e)\%$ of the optima for these propagation models, on the basis of previous results for submodular functions. In general the estimation of the influence $\sigma(S)$ of IM-Seed node set $S \subset V$ must be carried out by simulation, i.e. repeating the random process of influence propagation a number of times. Influence $\sigma(S)$ was proven to be a submodular function for both LTM and ICM propagation models, as well as for their generalizations. The critical computational load is, therefore, in the estimation of influence $\sigma(S)$, [67] propose a fast computation of $\sigma(S)$ that stores some of the previously computed influence sets, so that they do not need to be recomputed each time. Its disadvantage is the large memory requirements to store the precomputed influences. Another approach to reduce the time complexity of influence computation is a a Divide-and-Conquer method [120] applied to Influence Maximization on large-scale mobile social networks in two steps. First, the large-scale social network is divided into communities selected according to information diffusion, assuming ICM propagation model. Second communities are selected to look for influential nodes by dynamic programming. Further, a model of parallel computation of the influence spread in each community is proposed. Similar community decomposition is proposed in [114] where IM-Seed nodes are then selected from the communities. Another kind of acceleration is preprocessing the graph to obtain the spread trees which allow efficient computation of influence probabilities. This approach makes [83] in the context of targeted influence maximization, where some nodes are the target of the viral marketing, while others are susceptible or immune. Dealing with immune nodes requires some care, but it is not a source of complexity. Besides, graph communities are useful to reduce influence computation problem complexity also in [83]. The greedy algorithm has quadratic complexity on the number of nodes (which can be large in real life social networks), in order to attack this problem the Cost-Effective Lazy Forward selection) (CELF) method has been proposed [86], which consists in maintaining an ordered table of nodes and their marginal gain, so that influence candidates are taken from the top of this list. An enhancement to CELF, the CELF++ [57] fully exploits influence spread function submodality. Further, the Simpath algorithm [58] is based on the idea that under LTM propagation it is possible to compute an estimation of the spread by enumeration of the simple paths emanating of a node. Another optimization comes by the hand of a new propagation model called *credit distribution* [56], which avoids Monte Carlo simulation to achieve estimation of the spread on the basis of propagation traces. Another optimization considers the number of simple paths departing from a node as the indicator of spread potential [45] (ASIM), achieving a scalable algorithm for Influence Maximization under the ICM.

A different approach is the use of heuristics, such as Ant Colony Optimization (ACO) [39], to search for an almost optimal solution. One approach maps the influence maximization problem into the problem of finding a cycle of prescribed length with maximum influence spread. ACO are well suited to find cycles in graphs, however the approach only contemplates the selection of IM-Seed nodes, it does not reduce the complexity of computing influence spread. Another heuristic search tested is Simulated Annealing [75], where minimal set solution was trivially encoded as a binary vector and the influence spread was computed by Markov random simulation.

To avoid the computational load of influence spread, some authors use node features such as betweenness, diversity of community belonging, or k-shell decomposition value as indirect measures of influence [43]. In this same line [115] proposes "supermediators" using as indirect measure of influence the information spread, which decreases if the supermediators are removed from the network.

Chapter 3

TRUST PREDICTION

Trust prediction appears in much diverse areas of computational sciences, from mobile communications to social networks. To tackle this problem we propose two different classification system approaches in this Chapter. Section 3.2 relates the results of the experiments using Artificial Neural Networs (ANN) for Trust prediction. Section 3.3 describe the second aproach proposing a new feature extraction process ensuring that all reputation features are of the same dimension. The description of the datasets and the computational algorithms are in Appendices A and B, respectively. Conclusions are gathered in Chapter 7.

3.1 Problem definition

Given a Web of Trust (WoT) associated to some social system, specificed by weighted graph $G = (U,E,T)$, where there is a trust value $t_e \in \{-1,1\}$ associated with each edge $e \in E$. We want to predict how a User A would trusts another User B, positive or negatively, i.e. we want to predict t_{AB} , from our knowledge of the WoT. Therefore we have a two class classification problem, which can be dealt with by a variety of available algorithms, given an appropriate feature definition.

3.2 Experiments of trust prediction in social networks by Artificial Neural Networks

This section describes our first attempt to solve the classification problem using directly the trust values of the “friends” of the truster user A on the query trustee B. Formally, we build a feature set $F = \{t_{CB} | (A,C) \in E \& (C,B) \in E\}$. As these feature sets have quite differen sizes, to have feature vectors of the same dimension we discard elements. Computational experiments test the effect of discarding such data.

3.2.1 Experimental design

The computational experiments have been designed trying to answer the following questions:

- How well the ANN classifiers would generalize trust prediction? This question is addressed by the application of cross-validation methodology, ensuring that the test set is fully independent of the training set.
- How sensitive are ANN to the future growth of the social database? To answer this question, we have applied several partitions of the data into folds. The smaller partitions, such as 2-fold cross-validation, correspond to the situation where the size of the database is expected to double. On the other hand, the larger number of folds, i.e. 20-fold cross-validation, correspond to the situation where database size increase is marginal. Specifically, we make a 2, 5, 10, 15 and 20-fold cross validation experiments.
- What is the influence of pre-processing procedures, such as SMOTE, on the generalization results, specially in the minority class of distrusted relations? For this question, we have repeated all the experiments with and without SMOTE preprocessing.
- What is the effect of discarding data to obtain constant feature size?.

Regarding the performance measures reported in the experimental results section, we have focused on the Precision and Recall measures, defined as PRECISION $=\frac{TP}{TP+FP}$ and RECALL = $\frac{TP}{TP+FN}$, respectively. We report these values for each of the classes, trust and distrust, in order to assess the quality of response in each case. Recall is the classifier true positive prediction ratio relative to the entire positive class data, while Precision is the classifier true positive prediction ratio relative to the total positive predictions. Minority classes suffer from small recall and precision values in imbalanced classification problems.

3.2.2 Experimental results

The figures in this section contain plots of average Precision and Recall for varying number of folders in the cross-validation process. As said before, small number of folds can be interpreted as expecting a greater growth of the database between training and operational phases. A crucial question, because we want to assess whether predictors built at one moment in time will remain valid in the future. Plots refer to different classifiers (MP=Multi-Layer Perceptron, SVM=Support Vector Machine, RBFC=Radial Basis Function Conjugate training, RBFN=k-means plus LSE) and feature vector dimension ($d = 3$ or $d = 10$). Notice that the range of values changes

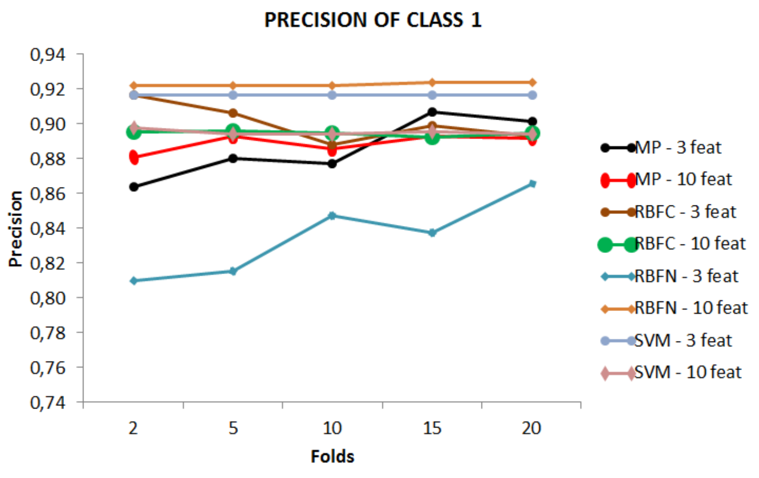


Figure 3.1: Average Precision of Trust prediction for diverse numbers of folders.

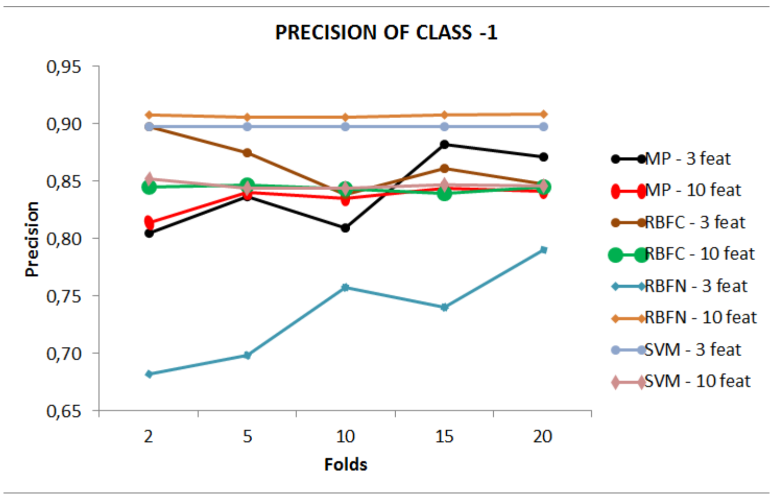


Figure 3.2: Average Precision of Distrust prediction for diverse numbers of folders.

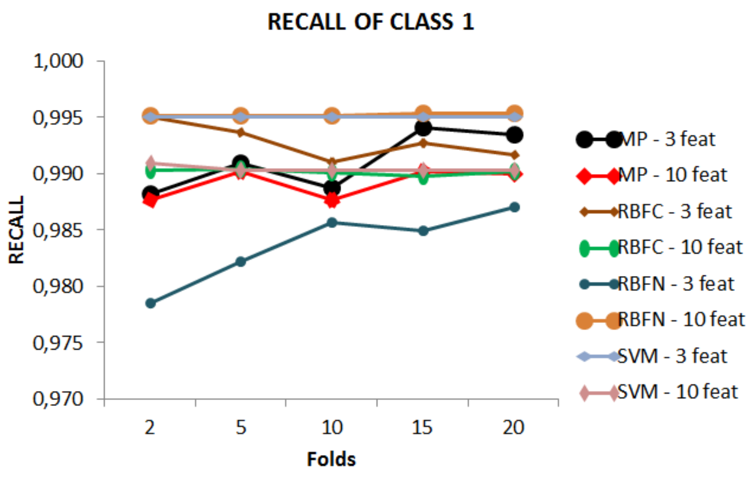


Figure 3.3: Average Recall of Trust prediction for diverse numbers of folders.

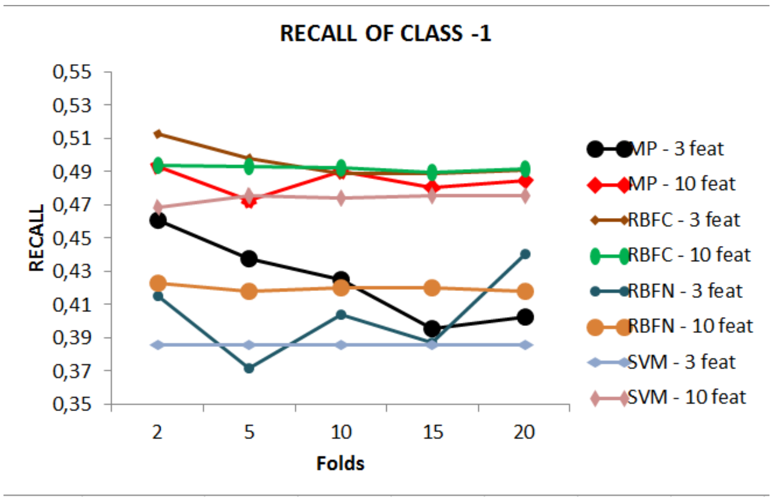


Figure 3.4: Average Recall of Distrust prediction for diverse numbers of folders.

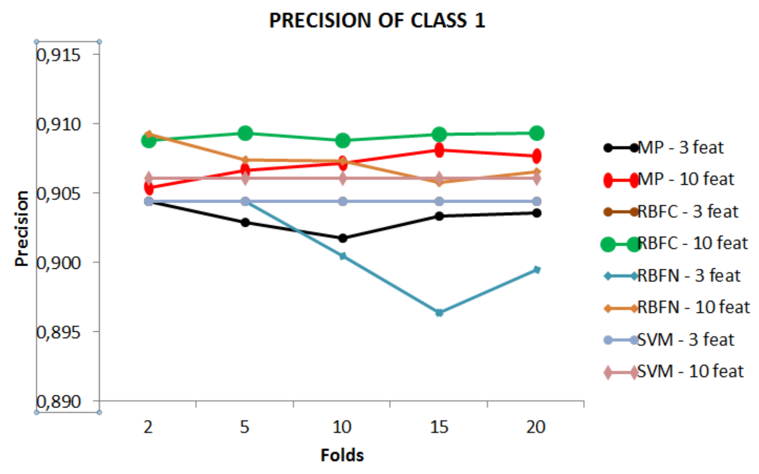


Figure 3.5: Average Precision of Trust prediction for diverse numbers of folders after SMOTE preprocessing of the datasets.

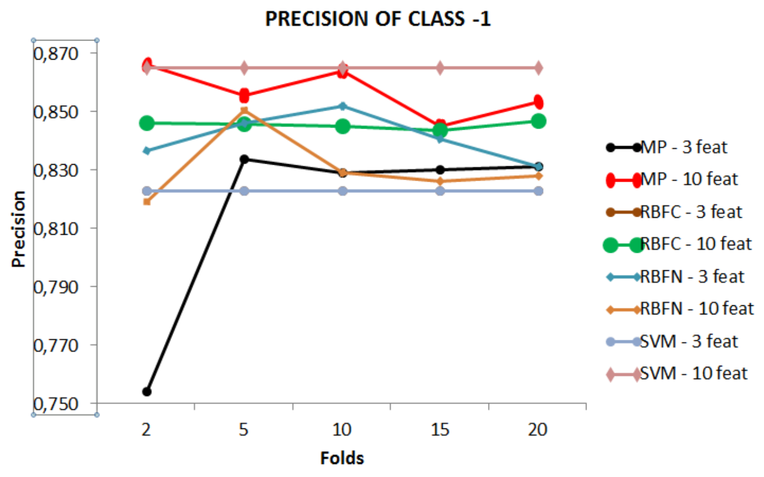


Figure 3.6: Average Precision of Distrust prediction for diverse numbers of folders after SMOTE preprocessing of the datasets.

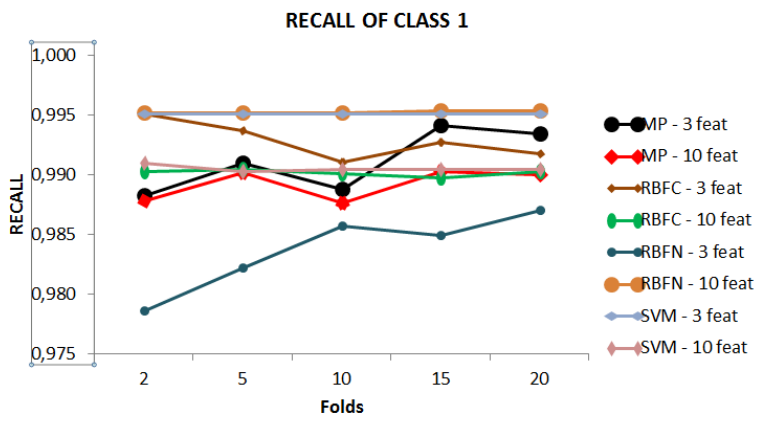


Figure 3.7: Average Recall of Trust prediction for diverse numbers of folders after SMOTE preprocessing of the datasets.

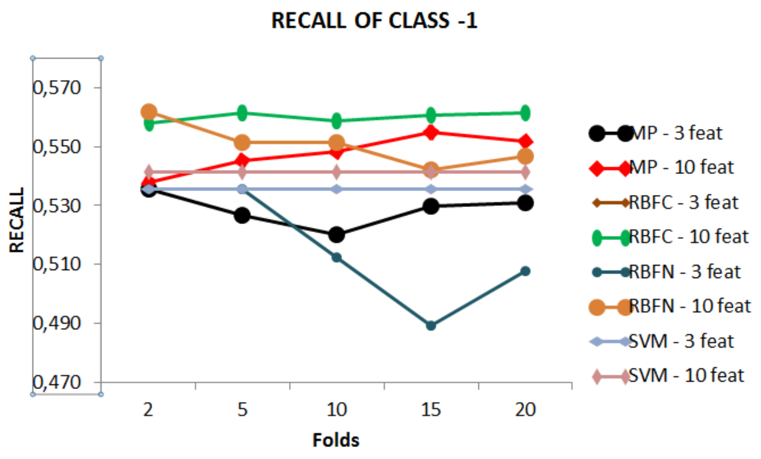


Figure 3.8: Average Recall of Distrust prediction for diverse numbers of folders after SMOTE preprocessing of the datasets.

from one plot to another. We have restricted them in order to highlight the differences between classifiers.

The first collection of experiments are carried out without any balancing preprocessing of the dataset. Figures 3.1 and 3.2 provide plots of the Precision of the Trust and Distrust classes, respectively, while figures 3.3 and 3.4 provide plots of the Recall of the Trust and Distrust classes, respectively. The second collection of experiments are carried out performing a SMOTE preprocessing of the dataset to improve imbalanced classes. Figures 3.5 and 3.6 provide plots of the Precision of the Trust and Distrust classes, respectively, while figures 3.7 and 3.8 provide plots of the Recall of the Trust and Distrust classes, respectively. In the following we discuss the most relevant effects that can be appreciated from the figures.

Effect of class imbalance It is quite notorious comparing the Recall plots of class Trust with those of the class Distrust. Best Recall results for Distrust are below 60% while for Trust are above 97% in all cases, sometimes quite close to 100%. The difference in Precision between Trust and Distrust classes is not so dramatic, both are very high. Trust Precision is above 90% for many classifiers, while the Distrust Precision is 2% down in the best cases, and 10% in the worst cases.

Effect of SMOTE preprocessing Comparison of results with and without SMOTE preprocessing show that there is a small decrease in Precision for both Trust and Distrust classes, though at the same time there is an effect of compression of the classifier Precision results. Without SMOTE the interval between best and worst Precision results (removing some outliers) is about 6% for the Trust class and 10% for the Distrust class, while after SMOTE this interval is reduced to 2% for the Trust class and 4% for the Distrust class. SMOTE has almost no effect on the Recall of Trust class, but there is relatively strong effect on the Recall of the Distrust class: there is an increase of 7% of the best classifiers, and a big reduction of the interval between the best and worst classifiers, from more than 10% down to 3%. This effect is remarkable because SMOTE adds robustness to the choice of classifier and feature vector size. However, the effect of SMOTE is not uniform on all the classifiers. For instance, focusing in the MLP classifier with feature vectors of size 10 (red line), its ranking among the classifiers changes if SMOTE is applied in almost all plots. Another example of this variable effect is the change of ranking of the RBFN classifier in the plots of Distrust Recall.

Effect of feature vector size The comparison of plots for feature vector sizes 3 an 10 shows that there is a big effect of the number of features, in general general towards worse results with the smaller set of features. However, there are some paradoxical results, such as the plots for Recall of the Trust class (with and without SMOTE) that show the reverse effect. This may be due a stronger bias towards the majority class with smaller feature vector. It seems that the SVM is the classifier less affected by this change of feature size. In general, it seems that 10 features is enough to obtain good results on the Trust class. However, the observation of the big performance gaps in the performance measures of the Distrust class suggest that increasing the feature size would improve results on this class.

Effect of classifier Regarding Precision of both classes and Trust Recall, the SVM provides the best results (with some exception in the Trust Precision after SMOTE). The SVM is quite robust to the number of features and cross-validation folders. Interestingly, in the most difficult issue of Distrust Recall the RBF and MLP provide better results and are more robust. This may point out to some kind of overfitting to the most frequent class by the SVM. However, this is not a general assessment of the performance of SVM as we are using linear kernel SVM.

Effect of the number of folders The number of folders in the cross-validation process is the way we have to pose the question: the classifier remains valid after the growth of the system? The smaller number of folders is 2, meaning that we expect the system to double in size for testing. It can be appreciated a general trend to improved results as the number of folders increases, meaning that most classifiers can cope with small additions to the system. In fact, for some plots this parameter seems to have the greatest effect. A very interesting result is the resiliency of SVM to this parameter: they show almost a flat response in all plots. In general, this is a very encouraging result, because it allows to expect that current studies will remain useful in the future. Interestingly, the MLP and RBF architectures also show this resiliency in the difficult Distrust Recall case, with and without SMOTE preprocessing.

Comparison with results in the literature Though a rigorous meta-analysis on the results found in the literature is difficult due to the diversity of approaches, performance measures and experimental details, we can refer some comparative results found on the Epinions database which show that the proposed approach is state-of-the-art. In [124] the Accuracy reported for various sizes of training sets is in the order of 80% to 90%, we can roughly compare it with the mean of Trust and Distrust Precisions, finding that our approach improves it. The Precision reported in [12] for Epinions dataset is below 90%, hence our results improve almost always. In the works of [4] the goal is to recover groups of trusted people, hence the definition of Precision and Recall is somewhat different because they refer to the percentage of the collection of n peers tested that are trustworthy, however their

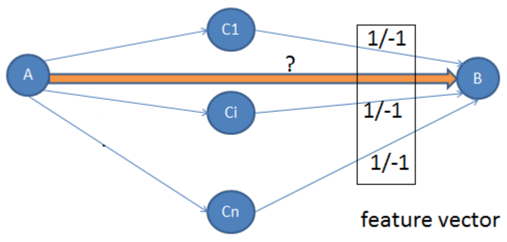


Figure 3.9: Unconditional reputation features of witnesses $\{C_i\}$ on the trustee B.

results are quite low compared with ours (maximum 20% Precision, 30% Recall). Best Precision reported in [13] is below 90%. In [81] best Precision and Recall results are below 80% and 70%, respectively.

3.3 Reputation Features for Trust Prediction in Social Networks

This section describes how we extract reputation feature vectors of constant size, appropriate for machine learning algorithms, invariant to the size of the reputation sets used in the previous section. We report increasingly good prediction results.

3.3.1 Feature extraction

From the original databases, we perform feature extraction in two different ways to obtain the reputation feature datasets, which will be published at the group's website[1] for independent third party assessment of results and experimentation. From the original database of triplets, we build several databases of reputation features, consisting on the observation of the Trust values of related users. Each database entry is composed of a feature vector of specific dimension and the trust value to be predicted. Let us introduce some common notation: For each triplet (A, B, t_{AB}) we construct a list of witness users $L_{AB} = \{C \,|\, (C, A, t_{AC}) \in \mathscr{D} \wedge (C, B, t_{CB}) \in \mathscr{D}\}$, where $\mathscr{D}$ denotes the original database of triplets. The node A queries its trusted peers C_i about their trust on target trustee B. The computation of the feature extraction took several days due to the large database sizes and the need to perform exhaustive examinations. We did not record computation times precisely.

[1]http://www.ehu.es/ccwintco/index.php/GIC-experimental-databases

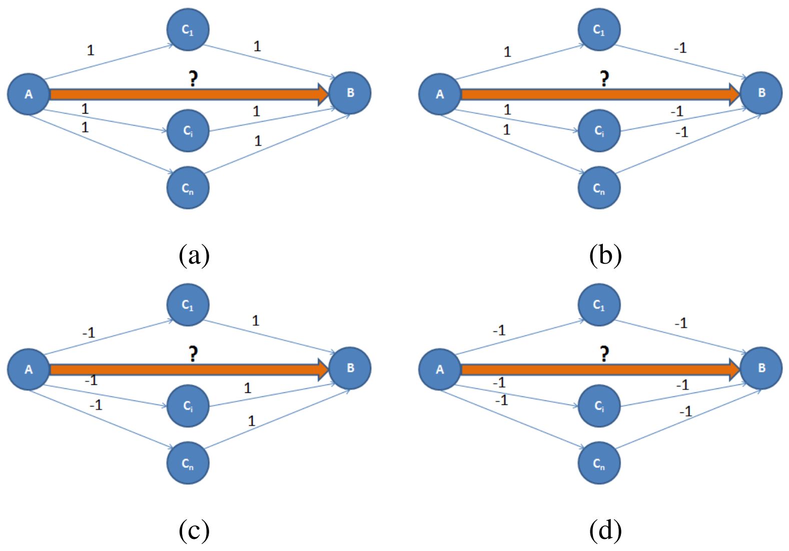


Figure 3.10: The four possible paths from truster A to trustee B through a witness C according to the trust labels used for the probabilistic reputation features.

3.3.1.1 Raw reputation vectors of fixed dimension

Figure 3.9 illustrates the reputation features provided by the witness and that are used to achieve the construction of the feature vector for a given (A,B) pair. Machine learning classifiers are often working in a data space of specific dimension, so that feature vectors are of fixed dimension. The set of witness that provide the reputation values may have any size. To solve this problem, a naive approach to the construction of the reputation database is as follows: Given feature vector dimension d, we discard the triplet (A,B,t_{AB}) if $|L_{AB}| < d$. If $|L_{AB}| > d$, we perform a random selection of d witness nodes C obtaining L^*_{AB} such that $|L^*_{AB}| = d$. The input/output pair (X,Y) in the reputation feature database corresponding to triplet (A,B,t_{AB}) is constructed such as $X = \{t_{C,B} | C \in L^*_{AB}\}$ and $Y = t_{AB}$. We have considered $d = 3$ and $d = 10$ in the present paper, in other words, the input data X is a matrix of 3 or 10 columns.

3.3.1.2 Probabilistic feature vectors

Another approach to obtain fixed size feature vectors is to consider some functions on the variable size reputation sets obtained from the witness sets, i.e. some statistics. In this paper we consider probabilistic features which are the conditional probabilities of the witness trust on the trust of the truster on the witness. They are computed as follows: For each set of witness L_{AB} we differentiate the following sets:

$$L_{CB}^{++} = \{C \in L_{AB} \,|t_{AC} = +1 \wedge t_{CB} = +1\},$$

$$L_{CB}^{+-} = \{C \in L_{AB} \,|t_{AC} = +1 \wedge t_{CB} = -1\},$$

$$L_{CB}^{-+} = \{C \in L_{AB} \,|t_{AC} = -1 \wedge t_{CB} = +1\},$$

$$L_{CB}^{--} = \{C \in L_{AB} \,|t_{AC} = -1 \wedge t_{CB} = -1\}.$$

These sets correspond to the possible kinds of paths linking the truster to the trustee through some witness. Figure 3.10 illustrates these four possible paths. Then we can compute the following conditional probabilistic features of the reputation feature set:

$$P(t_{CB} = +1 \,|t_{AC} = +1) = \frac{|L_{CB}^{++}|}{|L_{AB}|},$$

$$P(t_{CB} = -1 \,|t_{AC} = +1) = \frac{|L_{CB}^{+-}|}{|L_{AB}|},$$

$$P(t_{CB} = +1 \,|t_{AC} = -1) = \frac{|L_{CB}^{-+}|}{|L_{AB}|},$$

$$P(t_{CB} = -1 \,|t_{AC} = -1) = \frac{|L_{CB}^{--}|}{|L_{AB}|}.$$

Therefore, we obtain a feature vector of low dimension (i.e. 4) that summarizes the trust information on the witness set. After calculation of the probabilistic feature vectors in the case of Wikipedia, we remove instances with NaN values, so that the final feature dataset has 75,760 instances (78.45% of class "1", and 21.55% of class "-1"). From the Epinions database, we obtain a dataset with 547,694 instances (89.01% of class "1", and 10.99% of class "-1"), so that coverage of the Epinions database is 100% in the experiments.

3.3.2 Experimental work and Results

Machine learning classification algorithm implementations are obtained from Weka[2]. Specifically, we have tested Naive Bayes (NB), Multilayer Perceptron (MLP), Radial Basis Function classifier (RBFC) and network (RBFN), Support Vector Ma-

[2]http://www.cs.waikato.ac.nz/ml/weka/

chine (SVM), AdaBoost, and decision tree algorithms JRip and J48. Computational experiments consist on 10-fold cross-validation over the whole databases. We report overall average accuracy (OA), and *per* class recall (R) and precision (P) measures. The two classes considered are Trust (+1) and Notrust (-1).

3.3.2.1 Results on raw reputation features

In the first experiment, we build classifiers over raw reputation feature vectors of static size. Feature vectors are composed of the trust (1) and distrust (-1) values of the witness users towards the trustee. The class distributions are very imbalanced, which account for the poor recall and precision results of the Notrust class, and the corresponding low OA. We also report results after the application of database balancing techniques, i.e. SMOTE, which improve the performance over the Notrust class.

Results

Table 3.1: Results of cross-validation experiments on the raw reputation features of the Epinions Database without SMOTE preprocessing - 10 and 3 features.

	10 features					3 features				
		Trust		Notrust			Trust		Notrust	
Classif.	OA	R	P	R	P.	OA	R	P	R	P
NB	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0
MLP	93.80	98.8	94.6	49.0	81.5	93.41	99.3	93.7	39.9	87.1
RBFC	93.90	98.9	94.6	49.24	82.7	93.41	99.3	93.8	45.2	86.1
RBFN	93.75	99.5	93.9	41.6	90.8	92.77	98.6	93.7	40.40	75.8
SVM	93.90	99.0	94.5	47.6	84.6	93.44	99.5	93.6	38.6	89.8
AdaBoost	93.43	98.4	94.5	48.5	77.2	93.24	98.2	94.5	48.9	74.6
JRip	93.87	98.8	94.6	49.1	82.3	93.44	99.5	93.6	38.6	89.8
J48	93.86	99.0	94.5	47.6	83.9	93.44	99.5	93.6	38.6	89.8

Table 3.2: Results of cross-validation experiments on the raw reputation features of the Epinions Database after one SMOTE iteration of database balancing - 10 and 3 features.

	10 features					3 features				
		Trust		Notrust			Trust		Notrust	
Classif.	OA	R	P	R	P	OA	R	P	R	P
NB	85.77	88.2	94.0	74.7	58.5	90.02	100.0	90.0	0.0	0.0
MLP	90.03	97.8	90.7	54.9	84.9	89.45	97.5	90.4	53.1	82.6
RBFC	90.08	97.6	90.9	56.1	84.0	89.49	97.4	90.4	53.6	82.3
RBFN	89.84	97.3	90.9	56.2	82.1	89.49	97.4	90.4	53.6	82.3
SVM	90.15	98.1	90.6	54.1	86.5	89.49	97.4	90.4	53.6	82.3
AdaBoost	89.74	97.5	90.7	54.8	82.8	89.49	97.4	90.4	53.6	82.3
JRip	90.09	98.0	90.7	54.5	85.7	89.49	97.4	90.4	53.6	82.3
J48	90.12	98.0	90.7	54.5	85.7	89.49	97.4	90.4	53.6	82.3

Table 3.3: Results of cross-validation experiments on the raw reputation features of the Wikipedia Database without SMOTE preprocessing - 10 and 3 features.

	10 features					3 features				
		Trust		Notrust			Trust		Notrust	
Classif.	OA	R	P	R	P	OA	R	P	R	P
NB	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0
MLP	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0
RBFC	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0
RBFN	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0
SVM	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0
AdaBoost	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0
JRip	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0
J48	90.02	100.0	90.0	0.0	0.0	90.02	100.0	90.0	0.0	0.0

Table 3.4: Results of cross-validation experiments on the raw reputation features of the Wikipedia Database after one SMOTE iteration - 10 and 3 features.

	10 features					3 features				
		Trust		Notrust			Trust		Notrust	
Classif.	OA	R	P	R	P	OA	R	P	R	P
NB	75.19	81.6	84.7	56.1	50.5	90.02	100.0	90.0	0.0	0.0
MLP	76.86	93.8	79.2	26.4	58.6	75.56	95.1	77.4	17.2	54.0
RBFC	77.04	93.1	79.7	28.9	58.5	75.98	94.8	77.9	20.0	56.0
RBFN	76.87	93.3	79.5	27.9	58.1	75.87	94.3	78.0	20.7	55.03
SVM	76.87	94.0	79.1	25.7	58.9	75.99	96.2	77.3	15.6	57.9
AdaBoost	76.42	95.1	78.2	20.6	58.5	75.93	94.3	78.1	21.0	55.3
JRip	76.40	91.5	79.9	31.2	55.2	75.99	96.2	77.3	15.6	57.9
J48	76.78	94.1	78.9	25.0	58.7	75.99	96.2	77.3	15.6	57.9

Table 3.5: Results of cross-validation experiments on the raw reputation features of the Wikipedia Database after two SMOTE iteration - 10 and 3 features.

	10 features					3 features				
		Trust		Notrust			Trust		Notrust	
Classif.	OA	R	P	R	P	OA	R	P	R	P
NB	71.31	81.2	73.6	56.5	56.5	90.02	100.0	60.0	0.0	0.0
MLP	71.03	75.8	75.8	63.9	63.9	69.84	79.1	72.9	55.9	64.2
RBFC	71.27	76.1	76.0	64.1	64.2	69.84	79.1	72.9	55.9	64.2
RBFN	70.97	69.1	79.1	73.7	61.5	69.84	79.1	72.9	55.9	64.2
SVM	70.97	70.1	79.0	73.4	61.8	69.84	79.1	72.9	55.9	64.2
AdaBoost	67.46	88.9	67.3	35.4	68.1	67.42	87.1	67.7	38.1	66.3
JRip	71.27	73.2	77.6	68.4	63.1	69.84	79.1	72.9	55.9	64.2
J48	71.25	72.3	78.1	69.6	62.7	69.84	79.1	72.9	55.9	64.2

Tables 3.1 to 3.5 provide results on the Epinions and Wikipedia datasets of raw reputation features of vector dimension 10 and 3. The effect of database imbalance is stronger for the Wikipedia database. Tables 3.1 and 3.3 present the results without the application of the SMOTE preprocessing. The OA values are heavily influenced by the Trust class success, in Epinions the OA average for 10 and 3 features (there is not significant difference due to feature size) is 93% in Table 3.1, and 90% in the Wikipedia database. In the latter, results for Notrust class are zero for all classifiers and feature sizes. The application of SMOTE does not improve the OA, in fact it goes down to 89% in Epinions (Table 3.2) and to 76% in Wikipedia (Table 3.4), however there is a clear improvement on the Notrust classification in recall and precision of Notrust which can compensate the worsening for Trust if the cost of false Trust is much higher than that of false Notrust. Additional iterations of

SMOTE do not improve the OA, and continue the decrease/increase of recall and precision for the Trust/Notrust class, as shown in Table 3.5 where Notrust reaches 73% recall for 10 features and 55% recall for 3 features. The effect of feature size is also greater for Wikipedia than for Epinions database.

3.3.2.2 Results on probabilistic reputation features

Table 3.6: Average performance results of cross-validation experiments with different classifiers over the probabilistic reputation features. (OA) Overall Accuracy, (F1) F1 score, (AUC) area under the ROC.

	Wikipedia			Epinions		
	OA	F1	AUC	OA	F1	AUC
NB	100	98.3	0.973	100	98.7	0.983
MLP	99.99	99.1	0.981	100	99.2	0.991
RBFC	100	98.7	0.965	100	99.3	0.971
RBFN	99.99	98.6	0.966	100	99.4	0.976
AdaBoost	100	99.4	0.986	100	99.7	0.989
JRip	99.99	98.4	0.977	100	98.8	0.975
J48	99.99	98.1	0.962	100	98.2	0.972

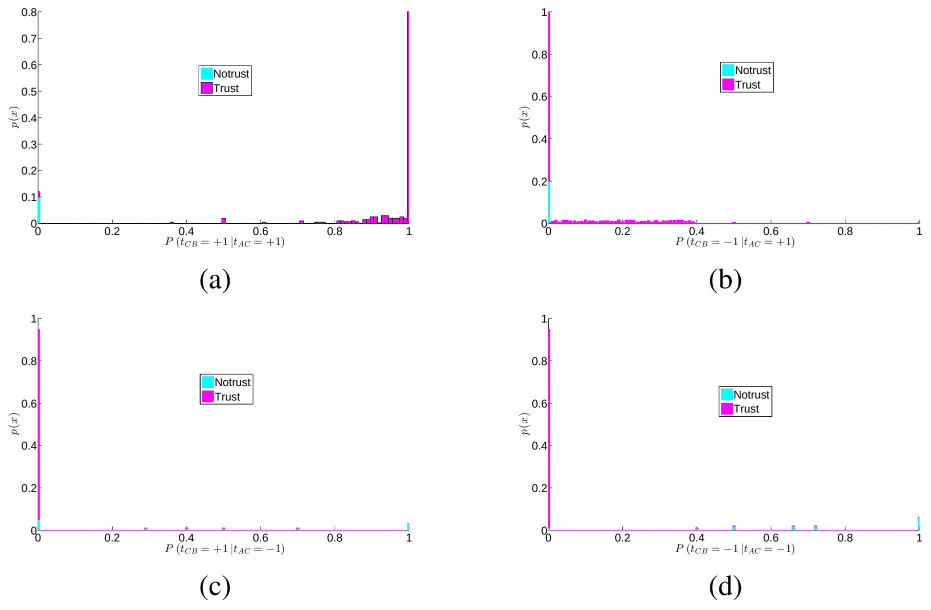


Figure 3.11: *A prori* histograms of probabilistic features from Epinions database. Magenta corresponds to Trust, Blue to Notrust. (a) $P(t_{CB}=+1\,|t_{AC}=+1)$, (b) $P(t_{CB}=-1\,|t_{AC}=+1)$, (c) $P(t_{CB}=+1\,|t_{AC}=-1)$, (d) $P(t_{CB}=-1\,|t_{AC}=-1)$.

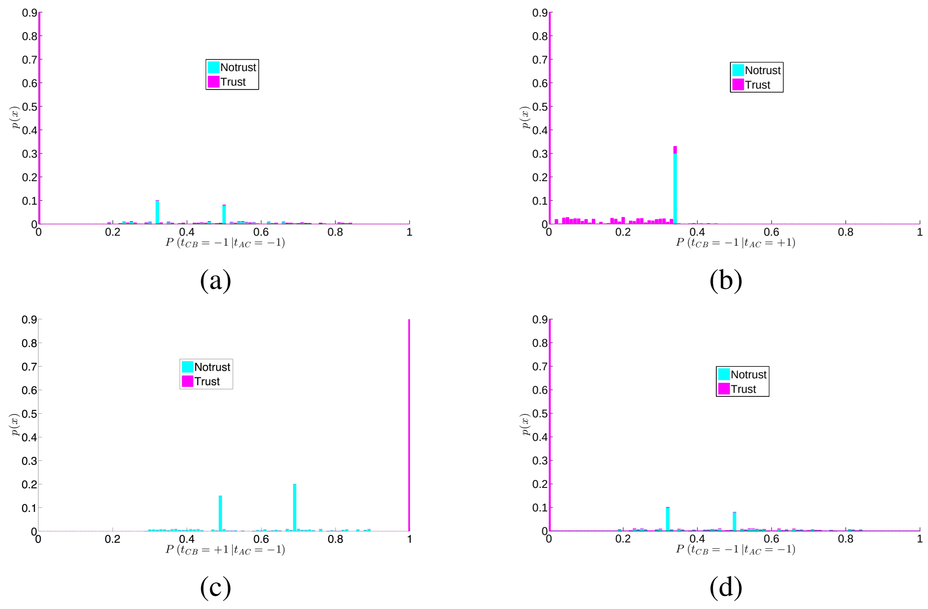
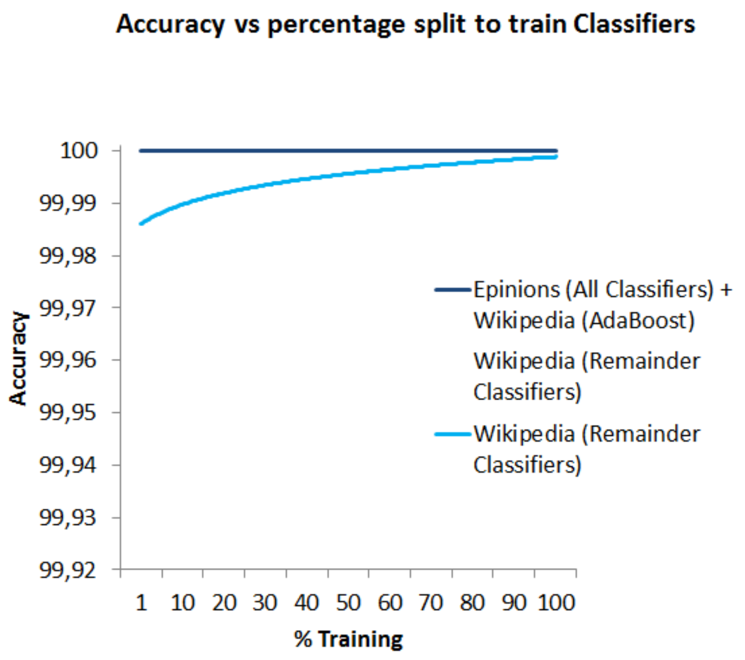
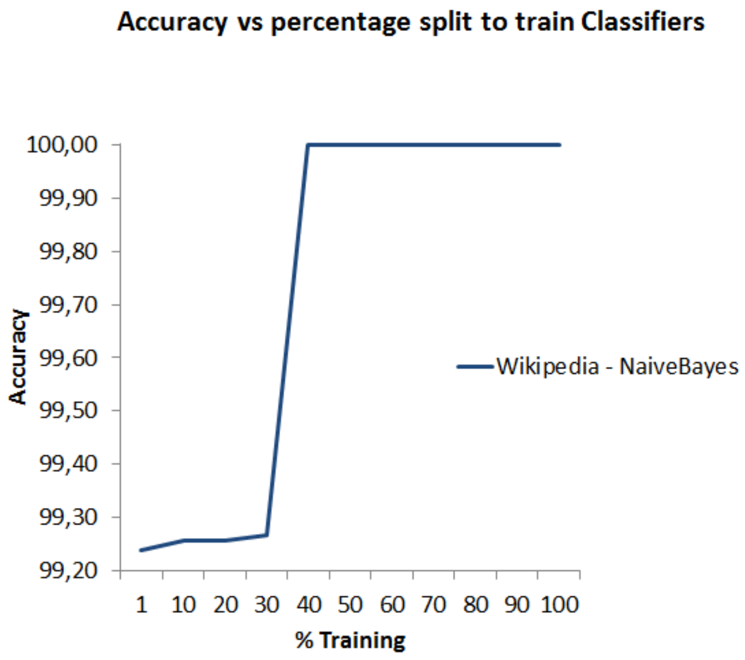


Figure 3.12: *A priori* histograms of probabilistic features from Wikipedia database. Magenta corresponds to Trust, Blue to Notrust. (a) $P\left(t_{CB}=+1\,|t_{AC}=+1\right)$, (b) $P\left(t_{CB}=-1\,|t_{AC}=+1\right)$, (c) $P\left(t_{CB}=+1\,|t_{AC}=-1\right)$, (d) $P\left(t_{CB}=-1\,|t_{AC}=-1\right)$.



(a)



(b)

Figure 3.13: Average accuracy obtained with training sets of increasing size expressed as percentage of the total database to train classifiers. (a) Joint plot of most classifiers on Epinions and Wikipedia databases. (b) Specific plot of Naïve Bayes results over the Wikipedia database.

As shown in Table 3.6, carrying a 10-fold cross-validation experiment over the probabilistic reputation features we achieve performance results close to perfect measured in average overall accuracy (OA), F1score (F1), and area under the ROC (AUC). The differences between the different classifiers are minimal, not significant in the statistical sense (*t*-test). The reason for such spectacular success is the fact that the probabilistic features are greatly discriminant of the classes in the problem at hand, as can be appreciated by inspection of figures 3.11 and 3.12 showing the class conditional a priori distributions of the values of each probabilistic feature, where Magenta corresponds to the Trust class and blue to the Notrust class. In all plots, the separation of classes is very clear making the problem very easy for conventional classifiers.

The final experiment with the probabilistic reputation features is aimed to test the ability of the classifier to stand in face of the expected continuous increase in size of the social network. To that end, we train the classifiers with a small training set and test them over the remaining database. The smaller the training set, the earlier in the life of the social network. We repeat the training and testing 10 times to obtain average values. The figure 3.13 shows the plot of the average accuracy obtained over the Epinions and Wikipedia reputation features. The percentages of training data go from 1% up to 99%. With the Epinions database, all classifiers achieve an accuracy of 100% from the smaller training set 1%. However, with Wikipedia we find the following:

- AdaBoost gets an accuracy of 100% for all training set sizes.
- NaiveBayes gets an accuracy of 99.2387% with a training size of 1%. Increasing training data size until 38% of the database, the accuracy is improved up to 99.2665%. With training sizes greater than 39% the accuracy reaches 100%. (Figure 3.13(b))
- The remaining classifiers reach an accuracy of 99.99% from a training data of 1% until 58% with little improvement. For training sizes above 59% the accuracy is 100%. (Figure 3.13(a)).

Chapter 4

RECIPE GENERATION

This Chapter deals with the problem of generating innovative solutions from the information gathered by a social network on a specific kind of tasks. By innovative we mean that the system must be able to generate solutions for previously unseen instances of the task, and that it may be able to improve over known solutions to known task instances. The kind of task that we dealt with in the context of the Thesis are tasks performed by household appliances. In order to do that, the system must be able to model user satisfaction, wich is the measure of the goodness of the solution given, and the inverse model, which provides the innovation desired for a new task. The Chapter is structured as follows: Section 4.1 comments on the problem definition. Section 4.2 gives a specification for the breadmaker appliance. Section 4.3 gives some experimental results. Dataset is described in Appendix A. Finally, conclusions are gathered at Chapter 7.

4.1 Problem definition

In this section we give the specification of the recommendation problem that we try to solve. The experiment context is the "SandS" European project (http://www.sands-project.eu/). In this project, Eahoukers (word that refers to easy house workers, in other words, users) provide a description of a problem dealing with household appliance usage to the social network. The system gives back a "recipe" that solves the proposed problem. These recipes are either proposed by the knowledge provided by other users or by the underlying intelligent layer [63]. Once that recipe is proposed, user can give the order to the system to execute it in the choosen appliance. Finally, users give a satisfaction of the recipe, this feedback is used to tune the intelligent layer and/or to personalize the system. In the following we formalize these definitions in the precise instance of an appliance, the breadmaker. As there is not real life data to validate our proposal, we have had to build some models to generate a synthetic dataset with some degree of arbitrary complexity, so that if our approach succeeds on this dataset, it can be successful in real life experiments. Appendix A contains some instances of the dataset and a description of the data synthesis process.

4.2 Specification of the recipe recommendation problem

This experiment demonstration of the approach is focused on the case of the breadmaker. It has some specific features that differentiate the way recommendations are generated. First, there is no task description per se. The user only gives the order to make the bread stating some expected satisfaction values with the result which are not stated beforehand. In other words, we only have the recipe and satisfaction pairs. The recommendation system then has two problems to solve, first it must learn the map from recipes to satisfaction, in order to predict the user satisfaction. Second, it must learn the inverse model from satisfaction to recipes in order to propose the best recipe for the user. It is also possible, once we have this inverse model, to tune the recipe to specific values of the predicted satisfaction. It may even possible to work with missing values, that is, to provide a recipe that matches some of the satisfaction parameters, when the other are left undefined. We have not touched this aspect in this experiment.

Recipes The baking operation consists of 5 steps carried sequentially: first leavening, second leavening, precooking, cooking and browning. Each step is specified by a pair $[Time, Temperature]$. Thus, in this case, the recipe consists in 10 variables $[r_1, ..., r_{10}]$.

Satisfaction The user give a satisfaction feedback. For the breadmaker, the satisfaction consists in 4 parameters: $[fragance, softness, baking, crust]$. These parameters are represented in 4 variables $[s_1, ..., s_4]$.

Problem specification The problems that we want to solve with this experiment are two:

- Direct prediction: What will be the satisfaction feedback obtained from the user for a given recipe?
- Inverse recommendation: Which is the recipe that I need to get a specific satisfaction?

Let us define:

- Let be R a recipe described by bread making variables, so $[r_1,...,r_{10}] = R$. Thus, $R \in \mathbb{R}^{10}$ and each r_i is normalized in the range $[0..1]$

- Let be S a satisfaction described by $[s_1,..,s_4] = S$. Thus, $S \in \mathbb{R}^4$ and each s_j is a number in the set $\{0,1,2,3,4,5\}$

To answer these questions, we define:

- A direct mapping $\phi(R) = S$ to predict the satisfaction of the user with the quality of the bread resulting from a proposed recipe (first question) $\phi : \mathbb{R}^{10} \longrightarrow \mathbb{R}^4$

- The inverse mapping $\phi^{-1}(S) = R$ that looks for the recipe that would provide the desired satisfaction parameter values (second question)$\phi^{-1} : \mathbb{R}^4 \longrightarrow \mathbb{R}^{10}$

We model the experiment with the numbers and parameters defined before but numbers and set of variables could be adapted to any other context of similar experimentation. These mappings are built by ELM because of the quick learning time which allows frequent updates when the experience of the users increase the database for learning. Notice that we only have information about experiments going in the direct prediction sense, i.e. we can try a recipe and ask the user its satisfaction. It is not possible to obtain experimental data in the other direction.

The first learning experiment is to calculate the regression of satisfaction values from given recipes. We denote this experiment as $\phi(R) \rightarrow S_j$. The second experiment is to calculate the regression o f recipe valuesfrom a given satisfaction, i.e. to create the recommendation. We denote this experiment as $\phi^{-1}(S) \rightarrow R_i$. We divided the dataset in several datasets according to the application requirements of the 10-fold cross-validation technique. Figure 4.1 is the pipeline which sumarizes the process of the experiment.

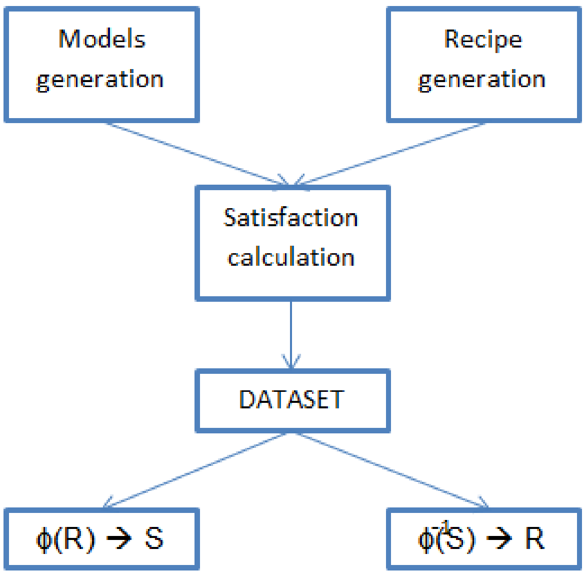


Figure 4.1: Pipeline of experiment

4.3 Experimental results

Experiments are carried out using ELM standar code in Matlab[1]. We select Sine ('sin') activation function for executions. We test results with 1 hidden unit until 525 hidden units that are the maximum hidden units allowed without raising a memory exception. Increasing neurons, square error decreases significantly. Table 4.1 shows the average regression error for the direct mapping regression for each satisfaction parameter obtained in a 10-fold cross-validation experiment for the two extreme ELM sizes. The best result is equivalent to a relative error, computed dividing the regression error by the variable range which is 5 for all satisfaction values, is below 0.01. Table 4.2 shows the average regression error for the inverse mapping regression for each recipe parameter obtained in a 10-fold cross-validation experiment for the two extreme ELM sizes. The best relative error, computed dividing the regression error by the variable range which is 1 for all satisfaction values, is below 0.2, still too high for the purposes of this experiment.

[1]Source-code: http://www.ntu.edu.sg/home/egbhuang/elm_codes.html

	1 hidden unit	525 hidden units
s1	1.4490	0.4972
s2	1.7756	0.4790
s3	1.6259	0.5639
s4	1.1084	0.4832

Table 4.1: Average cross-validation error results of satifaction prediction for given recipes: $\phi(R) \rightarrow S_j$

	1 hidden unit	525 hidden units
r1	0.4382	0.2816
r2	0.3910	0.2887
r3	0.4298	0.2919
r4	0.4080	0.2659
r5	0.4433	0.2923
r6	0.4063	0.2837
r7	0.3936	0.2903
r8	0.4743	0.2885
r9	0.4308	0.2911
r10	0.4456	0.2688

Table 4.2: Average cross-validation error results of recipe recommendation for desired satisfactions: $\phi^{-1}(S) \rightarrow R_i$

Chapter 5

PRODUCT RECOMMENDATION

In this Chapter we focus on a central problem in recommendation systems, that of deciding the kind of features that might be used for the prediction of the rating that a user will have of a given product, in order to decide if it can be recommended or not. We consider the specific case of product review webservices, of whom the Epinions site is a paradigmatic instance. The Epinions site has an associated Web of Trust (WoT) where users can provide feedback on the confidence of other users reviews. We consider two kind approaches for feature construction based on the ratings provided by the users: one is supervised in the sense of using the WoT reputation structure to build the features, the other is unsupervised in the sense that features are extracted from an unsupervisedly constructed similarity graph between users. The structure of the Chapter is as follows: Section 5.1 gives a definition of the problem that we inted to solve. Section 5.2 describes both feature selection approaches. Section 5.3 provides the experimental results over the Epinions dataset. Conclusions are gathered at chapter 7.

5.1 Problem statement

Given a social system with $u \in U$ users, and a catalog of items (products) $i \in I$ belonging to several categories C, the ratings of the products by (some of) the users are stored in a matrix R of size $|U| \times |I|$. The recommendation problem consists in the prediction of the rating $R_{u,i}$ that a user u would give to a product i using the information provided by the social system and/or the ratings given by other users. The problem may be addressed as a regression/classification problems, using features extracted from the rating matrix R.

5.2 Feature construction approaches

We propose two approaches to build the product features for recommendation computation and rating prediction. The first one takes into account the WoT associated to the users social network, using the opinion, specified by their product ratings, of the users trusted by the target user in order to build a product feature matrix. The second approach builds an unsupervised similarity graph between users based on their distances, computed from the knowledge about the product ratings given by each user stored in matrix R. A Singular Value Decomposition (SVD) of the R gives a matrix of user eigenvector descriptors which can be used for similarity computation. The ratings of the most similar users are used to build the product feature matrix.

5.2.1 Feature matrix based on the Web of Trust

Algorithm 5.1 Algorithm extracting the Web of Trust for each target user.

```
Given $G_u(U,T)$, ratings $R$
For each target user $u_i$ in $G_u(U,T)$
  if $(\exists t_{ij} \in T)$ then
    $[trustedUsers]_{u_i} \leftarrow u_j$
    $R_i \leftarrow R(u_j)$
  endif
```

Focusing on the WoT we can get the explicit collection of users that are trusted by a target user u_i : $U_i = \{u'_1, ..., u'_n | t_{u_i u'} = 1\}$. Algorithm 5.1 ilustrates the feature extraction. Graph $G_u(U,T)$ is built from the trust assertions $t_{ij} \in T$ given by the users. These communities of trusted users are then the basis for the estimation of the score prediction. Therefore the input for the training of the regressors doing the predictions will be the sparse representation of the vectors containing the recommendations. The final step ensures that the ratings of the trusted user are taken into consideration as the features for recommendation prediction of the user, stored in matrix R_i.

5.2.2 Feature matrix based on user similarities

Algorithm 5.2 Algorithm for extraction of target user similar users based on ratings.

```
For each target user $u_i$ in $G_r(\{\{U \cup I\}, R\})$
  For each $R_c$ matrix rating for category $c$
    $\psi \Lambda \phi^T = SVD(R_c)$
    $\Phi = \phi * \Lambda$  /*Each row of $\Phi$ are eigenvectors from a user $i$
    for each user  /*Get distances
      $d(u_i, u_j) = \|\Phi_i - \Phi_j\|$
    end
  end
end
Select $\alpha$ most similiar users in $D_u$
$[SimilarUsers]_{u_i} \leftarrow \alpha$ similar users
$R_i \leftarrow R(u_j)$ for $u_j$ in $[SimilarUsers]_{u_i}$
```

In this approach we construct features based on implicit knowledge collected in the system. We will operate the rating matrix using Singular Value Decomposition (SVD) that is a method for identifying and ordering the dimensions along which data points exhibit the most variation. This tool is often used in recommender systems to predict people's item ratings. We know that using SVD, rating matrix I can be stated as follows:

$$I = \psi \Lambda \phi^T$$

where ψ is the matrix of eigenvectors of $I(I)^T$, ϕ is the matrix of eigenvectors of $(I)^T I$, and we transpose it to get ϕ^T, and finally Λ is the matrix of square roots of non-zero eigenvalues ordered in decreasing order. Having an attribute matrix of $n \times p$ we would add as many zero-columns as necessary to Λ to keep the proper dimensions to allow multiplication of $\psi \Lambda \phi^T$. Thus, computing eigenvectors $\Phi = \phi \Lambda$ we get distance matrix D_u between nodes (users) calculating the distance between vectors. This way, we define distance between users as follows:

$$d(u_i, u_j) = ||\Phi_i - \Phi_j|| = \sqrt{\sum (\Phi_i - \Phi_j)^2}.$$

Once we have the distance matrix, we can choose a set of users U_d that are the closest users to the target user u_i. Those users are used to build the feature matrix as well as we will described in the next Section, but instead of having the array of trusted users we have the array of similar users. Algorithm 5.2 is used to obtain the array of similar users. For a user u_i the α most similar users will be choosen to perform the feature matrix. The final step ensures that the ratings of these similar users are stored in the matrix R_i of features for rating prediction.

5.3 Experimental results

Once we have the users whose opinion (ratings) we are interested in, we build the feature matrix being rows items and columns users ratings. We use the rating values of trusted/similar users to build a user specific rating matrix R_i. Thus, we have to predict rating values of the target user from data in R_i. Because of the high dimensionality of the matrix, we select 5000 users and 5000 items for experimentation. Items with no rating and users that have not rated any item are removed from the feature matrix.

The rating prediction is a regression problem instead of a classification problem. The performance measures taken into account accordingly are: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Relative absolute Error (RAE) and Root Relative Squared Error (RRSE).

Tables 5.1 and 5.2 show the results of built regressors based on Web of Trust users information, and on distances in hte user attribute space, respectively. We have tested six different regressors provided in the open source and free Weka software[1], some of the approaches are also described in Appendix B:

- Linear Regression which uses the Akaike criterion for model selection, and is able to deal with weighted instances. Best results with this classifier are obtained without attribute selection.

- Multilayer Perceptron, which is the most famous artificial neural network that uses backpropagation to classify instances.

- SMOreg classifier which is an implementation of Support Vector Regression.

- K-nearest neighbours classifier with K=1 gives the best results in this data.

- Random Tree uses constructs a tree that considers K randomly chosen attributes at each node.

- Finally, Additive Regression which is a meta classifier that enhances the performance of a regression base classifier. Each iteration fits a model to the residuals left by the classifier on the previous iteration. Prediction is accomplished by adding the predictions of each classifier. The Additive Regression version in this paper is Support Vector Regression with a Polynomial Kernel, and Support Vector Machines as optimization function.

[1]http://www.cs.waikato.ac.nz/ml/weka

	MAE	RMSE	RAE	RRSE
Linear Regression	0.37	0.79	32.13%	56.87%
Multilayer Perceptron	0.59	0.94	51.67%	67.42%
Support Vector Regression	0.81	0.34	30.09%	57.96%
KNN	0.36	0.79	31.60%	56.84%
Additive Regression	*0.30*	*0.96*	*25.91%*	*68.69%*
Random Tree	0.36	0.79	31.60%	56.84%

Table 5.1: Results of features extracted from Web of Trust

	MAE	RMSE	RAE	RRSE
Linear Regression	0.74	1.49	44.47%	79.11%
Multilayer Perceptron	0.97	1.64	58.15%	87.23%
Support Vector Regression	*0.56*	*1.14*	*33.43%*	*60.41%*
KNN	1.25	1.39	85.10%	84.32%
Additive Regression	*0.56*	*1.14*	*33.43%*	*60.41%*
Random Tree	0.74	1.49	44.47%	79.11%

Table 5.2: Results of features extracted from user distances

All experiments are tested with fold cross validation. Features extracted from WoT give better results than features based on user distances. In the first case, in general, the Mean Absolute Error is over 0.30-0.36 marks. In the second one, in general, the Mean Absolute Error is over 0.56-0.74 marks. The best classifiers in both experiments is Additive Regression.

Chapter 6

INFLUENCE MAXIMIZATION

Influence Maximization (IM) is a problem of economical importance for advertising and other industries. Its relevance is highlighted in the advent of the social network web services, which have exploded the amount of information available about the costumers and their relations. In this Chapter we deal with the heuristic solution of this problem by two specific novel approaches: the formulation of Harmony Search for Influence Maximization, and a novel heuristic that is closely related to the classical Greedy Search (GS) approach. We have moved the detailed description of conventional approaches, i.e. GS, Genetic Algorithm (GA), Simulated Annealing (SA), and Harmony Search (HS) to Appendix B in order to have a more clear exposition of results.

The structure of the Chapter is as follows: Section 6.1 provides the problem statement. Section 6.2 details the application of Harmony Search to the solution of IM. Section 6.3 describes our novel heuristic solution. Section 6.4 provides detailed description of the experimental design and the experimental results obtained. Conclusions are gathered in Chapter 7.

6.1 Problem statement

Social networks are represented by a weighted directed graph $G(V,E,W)$ where nodes $v \in V$ represent individuals of the community, edges $(v,v') \in E$ represent social relationships between them, and W are the weights of either nodes or edges. The two basic spread models of influence propagation are the Independent Cascade model (ICM) [54], and the Linear Threshold model (LTM) [64].

In the LTM diffusion model, nodes are weighted by a decision threshold $w_v \in \mathbb{R}$, while in the ICM the weights are placed on the edges, they are propagation probabilities. Nodes can be active or inactive, i.e. they have been influenced or not. When a node becomes active it is possible to spread influence to an inactive node from its neighborhood, and the influence propagation is modeled by an iterative process.

In the LTM, the node becomes active when the percentage of active neighbors is above the threshold, i.e. $\frac{1}{|V|}\sum_{v'\neq v} a_{v,v'}\sigma_{v'}(S) \geq w_v$, where $a_{v,v'} \in [0,1]$ is the entry in the adjacency matrix A such that $a_{v,v'} = 1$ iff $(v,v') \in E$, and $\sigma_v(S)$ is an indicator function that values 1 iff node v belongs to the influence spread of a IM-seed set S. When a node v becomes active it is added to the actual influence spread, i.e. $\sigma(S) \leftarrow \sigma(S) \cup \{v\}$. In the ICM, the weights $w_{v,v'} \in \mathbb{R}$ are measures of the strength of the relation, i.e a probabilistic measure of the influence capability of one node over another. Obviously, we have $w_{v,v'} = 0$ iff $a_{v,v'} = 0$. Each node activates its neighbors by carrying out a stochastic decision by Monte Carlo sampling the Bernoulli distribution defined by edge probability $w_{v,v'}$. A variation of the LTM allows an inactive to become active when the summation of influence degrees on the incoming links is greater than the node threshold. There are works such as [93] analyzing the computational complexity of influence maximization problem in the deterministic LTM.

In both ICM and LTM there is no reversibility of states so that the influence propagation process will end when no more nodes may become active. In the literature ICM and LTM propagation models are usually applied separately but some works subsume both models [110], in some cases new evidences such as trust are added to the propagation model [102]. In other words, we do not consider viral propagation models [112] which contemplate *infection* and *recovery* of individuals. Study of such models by mean field analysis [92] show that it is possible to determine the diffusion rate that ensures that the system reaches a steady state where the infection persists.

The goal of Influence Maximization (IM) [38, 77] is to find the minimal subset of maximally influential nodes (IM-seed nodes). The influence spread in a graph $G = (V,E)$ of a subse of nodes $S \subset V$ is a set function $\sigma : 2^V \to \mathbb{R}$, which is non-negative, i.e. $\sigma(S \geq 0)$ for all $S \subseteq V$, monotone, i.e. $\sigma(S) \leq \sigma(T)$ for all $S \subseteq T$, and submodular, i.e. $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(T \cup \{v\}) - \sigma(T)$ for all $S \subseteq V$ and $v \in V$. The IM problem is therefore posed as

$$S^* = \min_{|S|}\max_{S\subset V} \sigma(S).$$

It has been shown that IM is NP-complete, hence with computational time growing exponentially on the problem size, given by the number of nodes, and the density of the social graph. Exhaustive search and exact solution of the problem is, therefore, out of the question so that most published approaches follow some kind of approximation, aka heuristics or metaheuristics. Ensuing sections present our proposals.

6.2 Harmony Search

Algorithm 6.1 Harmony Search algorithm adapted to Influence Maximization

```
1. Given probabilistic social graph $G = (V, E, W)$
2. Initialize HS parameters and $HM_0$
3. while $t \leq NI$
    (a) for $i = 1 \ldots N$        //Improvise new harmony $\mathbf{x}'$ ,
        i. if $r \leq HMCR$
            A. $x'_i \overset{\in}{\leftarrow} \{x^1_i, \ldots, x^{HMS}_i\}$,
            B. if $r < PAR$ then $x'_i \leftarrow x'_i + \alpha_i$; $\alpha_i \sim U[-BW, BW]$;
        ii. otherwise $x'_i \overset{\in}{\leftarrow} X_i$
    (b) Evaluate harmony $f(\mathbf{x}')$
    (c) If $f(\mathbf{x}') > f(\mathbf{x}^{HMS})$
        i. replace $\mathbf{x}^{HMS}$ in HM, and sort HM.
4. Return best Harmony
```

Algorithm 6.1 shows a pseudo-code of the HS optimization procedure specialized for the IM problem resolution. In the first step problem data is read and algorithm parameters are initialized. The graph where the IM problem is posed can be a randomly generated graph or a real social network graph. Next, HS algorithm parameters controlling the optimization process are specified: the harmony memory size (HMS) specifying the number of solution vectors stored in the harmony memory, the harmony memory considering rate (HMCR) specifying if a variable improvisation is extracted from the memory, the pitch adjusting rate (PAR), and termination criterion (maximum number of searches). In the next step the harmony memory (HM) is initialized. Next we carry the improvisation of a new harmony. A new harmony vector, $\mathbf{x}' = (x'_1, ..., x'_N)$ is generated based on memory considerations, pitch adjustments, and randomization. With probability HMCR the value of the design variable x'_i is selected from the collection of values in the HM, i.e. $x'_i \overset{\in}{\leftarrow} \{x^1_i, \ldots, x^{HMS}_i\}$ where $\overset{\in}{\leftarrow}$ denotes random selection from a set of values. In Algorithm 6.1, $r \sim U(0,1)$ denotes a random number with uniform distribution in the interval $(0,1)$. If $r > HMCR$, in other words with probability $1 - HMCR$, the value of the variable is extracted from its range set X_i. The new value can be fine tuned with probability PAR after a positive test with HMCR. i.e. $x'_i = v_{i,k\pm m}$where $v_{i,k\pm m}$ is either the next value of the range set of a discrete variable or a random mutation in continuous variables. In some implementation, the first variable is always assigned a value from the history. If there is pitch adjustment for x_i' , the pitch-adjusted value of $x_i(k))$ is $x_i' \leftarrow x_i' + \alpha_i$ where α_i is a sample of a random variable following a uniform distribution $U(-BW, NW)$, where BW is an arbitrary distance bandwidth for the continuous design variable. In the next step HM is updated. If the new harmony vector is better than the worst harmony in the HM in terms of the objective function value, the new harmony is included in the HM and the existing worst harmony is excluded from the HM. The HM is then sorted by the objective function value.

For IM, harmonies are binary vectors encoding the IM-Seed set, that is, a vector component value is 1 if the corresponding graph node belongs to the seed set, otherwise it is zero. Influence Maximization is a multi-objective problem, because we want to achieve two goals: (a) maximize spread, and (b) minimize IM-Seed size. Given an harmony $\mathbf{x}$ and a probabilistic graph G_i', the evaluation returns:

$$f(\mathbf{x}, G_i') = \sigma(S_{\mathbf{x}}) + 10^{-log_{10}V} * (V - S_{\mathbf{x}}), \tag{6.1}$$

where V is the set of nodes in the network, $\sigma(S_{\mathbf{x}})$ is the number of nodes that have been visited through the spread model (Independent Cascade Model) and $S_{\mathbf{x}}$ the number of actives nodes in the harmony $\mathbf{x}$. In this way, the harmony which visits the largest amount of nodes in the network with the minimum active nodes in the harmony will be reported as local optimum at the end of the computational process.

6.3 New Heuristic for IM solution

Our proposed new heuristic method starting step is to identify nodes with zero in-degree, i.e. no incoming edge ending to them, (S_0 in step 2). The justification is that any K-seed set whose influence covers all the graph must include them because they can not be influenced by any other node. The set of remaining nodes R_0 consists of all nodes not in the initial solution S_0 nor in its influence spread $\sigma(S_0, A_b)$ computed using the base adjacency matrix. The set R_0 contains all candidate nodes to enlarge the solution, because the removed nodes add nothing to the influence spread of the actual solution S_0. Next, the adjacency matrix is simplified removing edges ending into or departing from nodes removed from R_0, because these edges will not play any role in ensuing influence computations. The algorithm proceeds by iterating the following steps until the set of remaining nodes R_t is empty. The first step is to find the node v^* with maximal influence in one step $\sigma_1(\{v\}, A_t)$, i.e. paths of lenght 1, using the simplified adjancency matrix A_t. The K-seed solution is increased adding v^*, while the set of remaining candidate nodes is decreased removing v^* and its one step influences. The adjacency matrix is updated

Algorithm 6.2 Proposed IM heuristic (IMH) solution algorithm

```
1. Given social graph $G = (V, E, W)$, with adjacency matrix $A_b = \left[a^b_{v,v'}\right]$
2. $S_0 = \left\{ v \middle| \sum_{v' \neq v} a^b_{v',v} = 0 \right\}$
3. $R_0 = V - \{S_0 \cup \sigma(S_0, A_b)\}$
4. $A_0 = \left[a^0_{v,v'}\right]$ s.t. $a^0_{v,v'} = 0$ if $v \notin R_0 \vee v' \notin R_0$; otherwise $a^0_{v,v'} = a^b_{v,v'}$
5. $t = 0$
6. iterate until $R_t = \emptyset$
    (a) $v^* = \arg\max_{v \in R_t} \{\sigma_1(\{v\}, A_t)\}$
    (b) $S_{t+1} = S_t \cup \{v^*\}$
    (c) $R_{t+1} = R_t - \{\{v\} \cup \sigma_1(\{v\}, A_t)\}$
    (d) $A_{t+1} = \left[a^{t+1}_{v,v'}\right]$ s.t. $a^{t+1}_{v,v'} = 0$ if $v \notin R_t \vee v' \notin R_t$; otherwise $a^{t+1}_{v,v'} = a^t_{v,v'}$
    (e) $t \leftarrow t + 1$
7. Return $S_t$
```

accordingly. The heuristic of assuming that maximal one step influences would correspond to maximal influence spreads is a extreme form of greedines, but that appears to be effective from the experimental results. At the same time, removing only one step influences may leave candidate nodes which in fact do not improve the influence spread when added to the solution, however from the experimental results, it seems to have little effect. We will denote this heuristic as IMH in the following.

6.4 Experimental results

In this section we report experimental results comparing the HS, SA, GA, GS algorithms and the proposed IMH. First we comment on the construction of the experimental graphs. We evaluate the methods comparatively on a large collection of synthetic graphs of increasing size, and finally we report results on subgraphs of a real life social network of increasing size. The code and the data for these experiments can be found in the following site: [1].

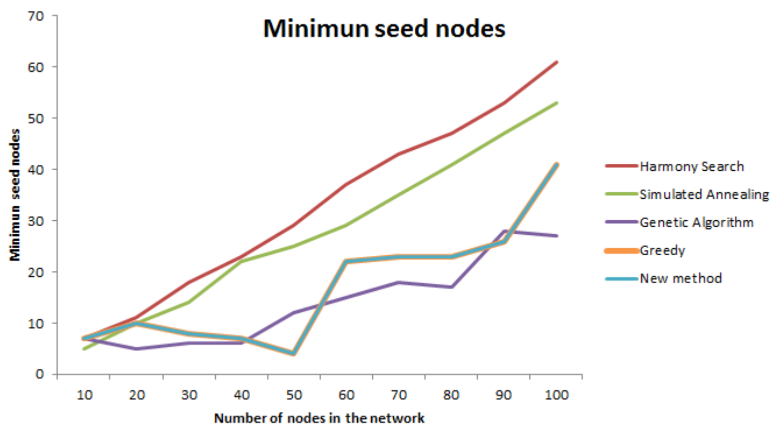
6.4.1 Graph construction

A social network is defined as a directed graph $G(V,E)$ where V is the set of nodes that represents the set of users and E the set of edges that represents the set of relationships among users. Given a directed graph $G(V,E)$ edges are weighted by $w_{ij} = 1/degree_{in}(j)$, thus the graph becomes a weighted graph $G(V,E,W)$ where W is the set of weights that are values in the interval $[0,1]$. Some of the experiments reported here are done on synthetic graphs. To build such graphs we generate the random weights of a complete graph, determining the in-degree of each node. Thus, from the a probabilistic adjacency matrix we generate graph instances $G'_i(V,E')$ where edges will appear according with the probabilistic weight. An weighted edge with a high weight has more probability to appear in the sampled graph G'_i. We define $G' = \{G'_1,...,G'_n\}$ as the set of sample graphs used to test the diverse IM heuristics. For the experiments referred below, the ICM influence propagation estimation consists in the average of the propagations over the set of sample graphs.

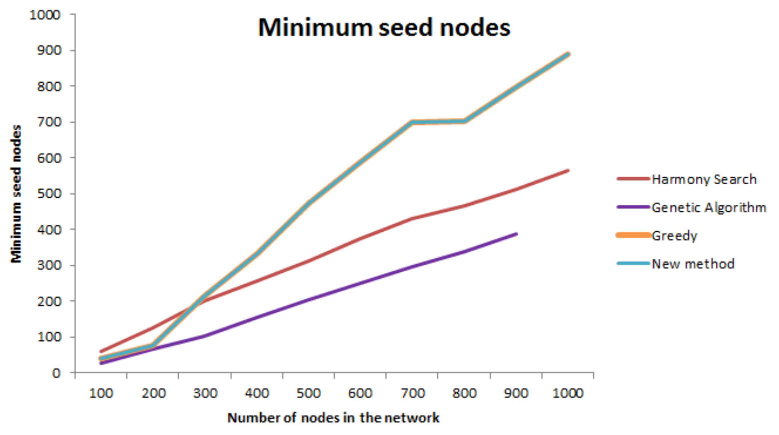
6.4.2 Experimentation with synthetic graphs

Figures6.1, 6.2, and 6.6 show a comparison among HS, SA, GA, GS algorithms, and our proposition IMH. Figure 6.1 plots the K-seed node set size found by the algorithms for simulated social network graphs of increasing sizes, while influence

[1] `http://www.ehu.eus/ccwintco/index.php?title=Influence_Maximization`



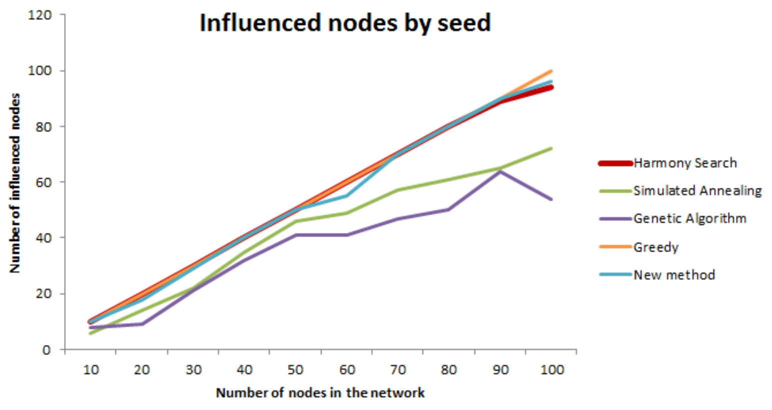
(a)



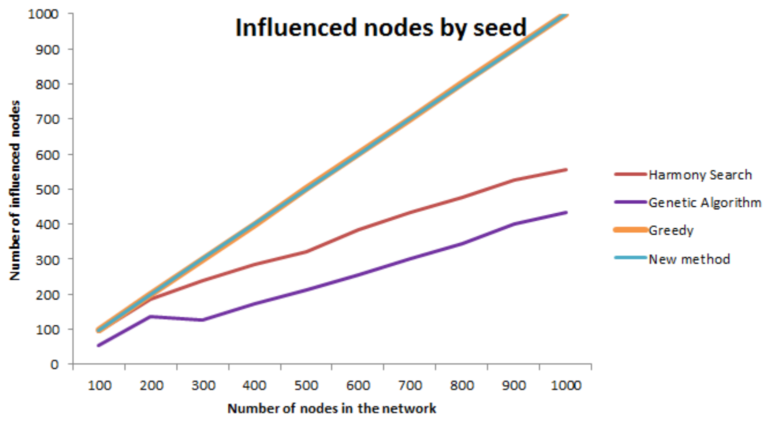
(b)

Figure 6.1: Results for increasing size random graphs of GA, SA, HS, Greedy and New Method measured as the K-seed size found.

spread size is plotted in Figure 6.2. Figure 6.3 plots the ratio of the influence spread size versus the size of the K-seed set in order to visualize the relative success of each seed node. Figures 6.1(a), 6.2(a), and 6.6(a) give results for small size graphs. Figures 6.1(b), 6.2(b), and 6.6(b) give results for big size graphs. We have not computed SA for large graphs due to its very slow convergence and big computational times. In all cases, IMH and GS algorithms are always the methods which achieve the biggest influence spread with the minimal K-seed nodes. Our method is faster than Greedy algorithm. We give experimental results about speed-up improvements in another section.

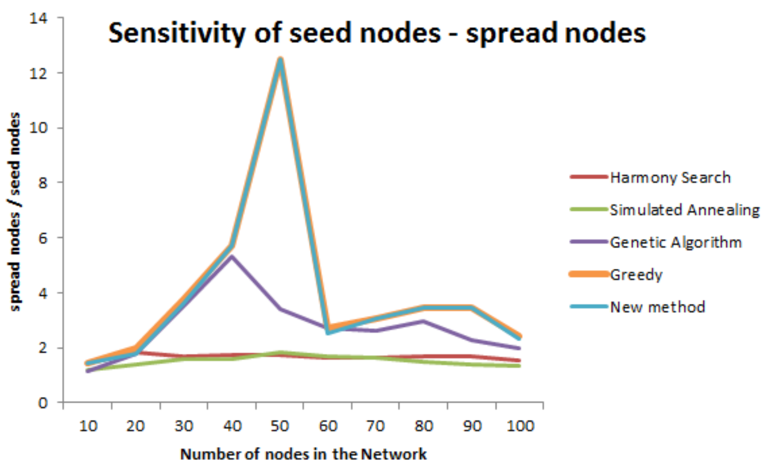


(a)

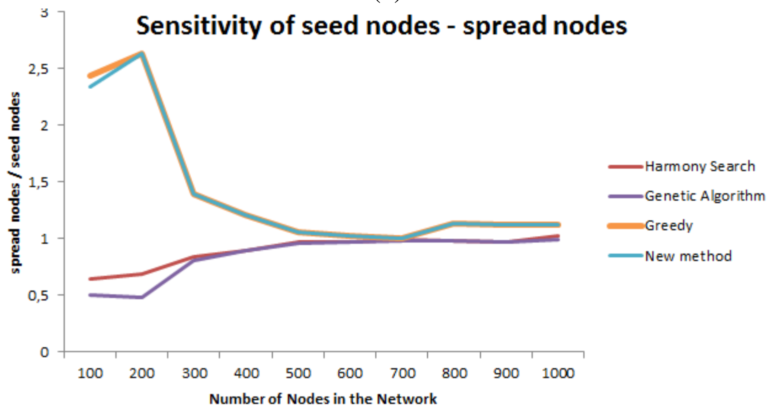


(b)

Figure 6.2: Results for increasing size random graphs of GA, SA, HS, Greedy and New Method measured as the size of influence spread.



(a)



(b)

Figure 6.3: Results for increasing size random graphs of GA, SA, HS, Greedy and New Method measured as the ratio of influence spread to the K-seed size.

6.4.3 Experimentation with Epinions database

In order to evaluate the algorithms on a realistic problem, we apply them to the Web of Trust graph from the Epinions site [2], which is a social webservice where users provide reviews of products of any kind, ranging from music up to perfumes or construction hardware. These product reviews are the base for the establishment of trust relations between users. Trust is a binary variable taking values in $\{-1,1\}$: a truster user can choose to trust (1) or distrust (-1) another, the trustee. Each sample is a triplet (A,B,t_{AB}) composed of two user indexing numbers (no personal data of any form is included) and the binary Trust value of the first user on the second user. Therefore, Trust relations define a directed graph, with weighted edges. Experimentation is done over subnetworks of sizes are in range 10 to 100 and 100 to 700. Figures 6.4, 6.5, and 6.6 show results of experimentation over these Epinions subnetworks. In all cases, the IMH and GS algorithms provide the greatest spread, while GA and SA provide more compact seed of smaller influence spread. Looking at the ratios plot in figure 6.6 we find that there is significative difference between methods. As mentioned in previous experiment, our method is faster than Greedy algorithm. We propose experimentation about speed-up improvements in the following section.

6.4.4 Computation time comparison experiments

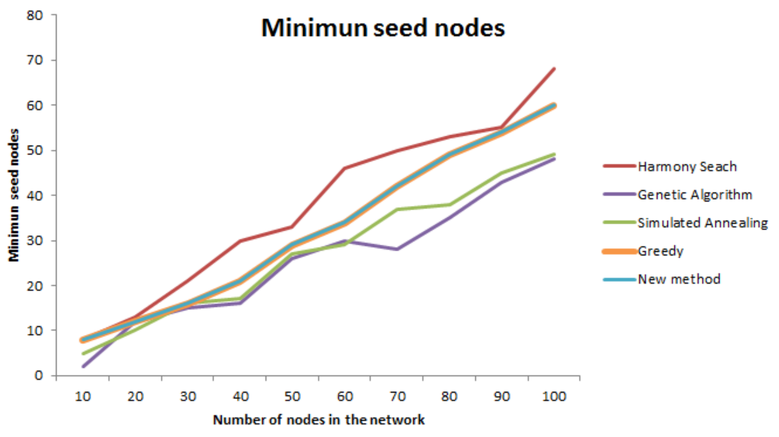
We propose two experiments to compare the computation time cost of the GS algorithm and the proposed IMH method. The first experiment consists on getting the minimum k-seed nodes from four different graphs, each of 1000 nodes but with different edge density. Table 6.1 shows the time spent to get the approaximates to the optimal K-seed set of influential nodes. When the graph has low density, reaching the solution is quite fast for GS but the bigger the density is the greater time the GS requires. In contrast, IMH method achives a solution always under one second. The experiments were run on a 3GHz 4 core Intel computer.

Density	New method	Greedy
0.000011	0.018 sec.	5.587 sec.
0.00011	0.144 sec.	5.817 sec.
0.0011	0.031 sec.	> 5 min.
0.011	0.050 sec.	> 5 min.

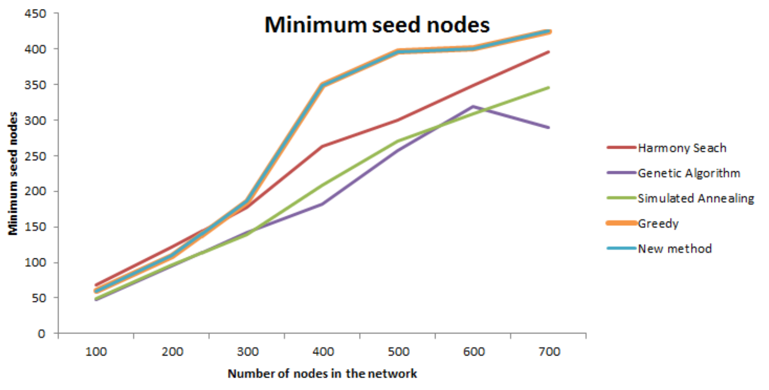
Table 6.1: Comparison of speed using matrix of different density

For the second experiment we keep a small edge density in the graph but we

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

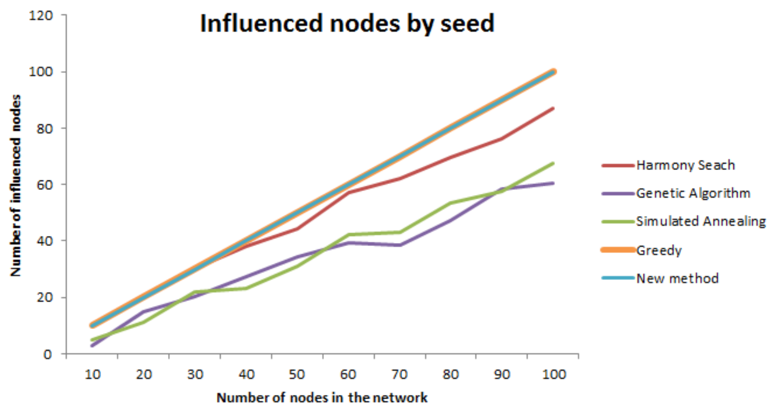


(a)

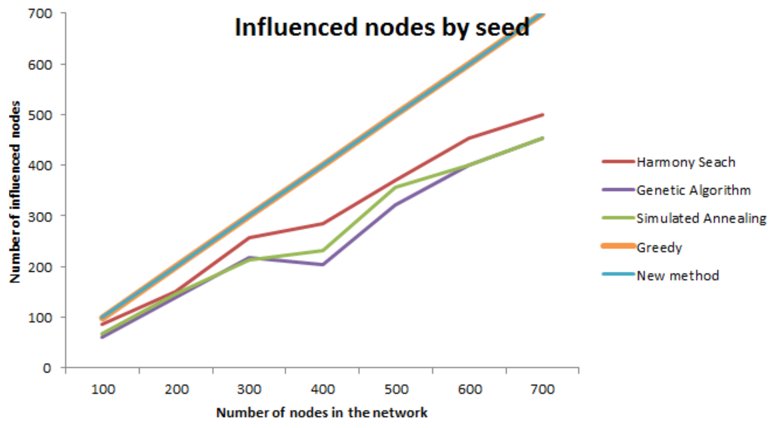


(b)

Figure 6.4: results for increasing size Epinions subgraphs of GA, SA, HS, Greedy and New Method measured as the K-seed size.

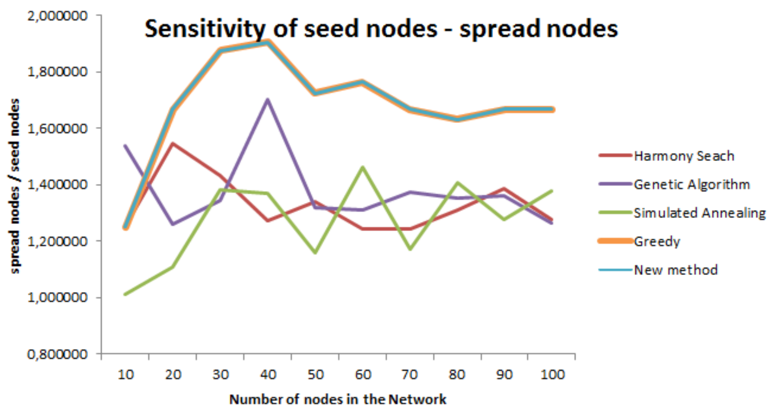


(a)

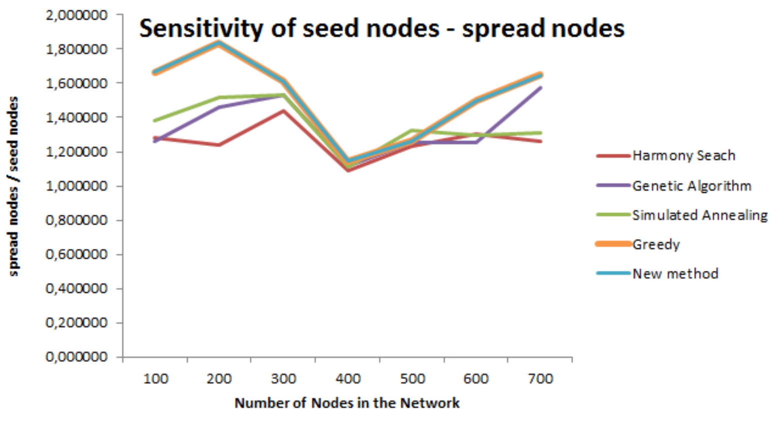


(b)

Figure 6.5: Results for increasing size Epinions subgraphs of GA, SA, HS, Greedy and New Method measured as the size of influence spread.



(a)



(b)

Figure 6.6: Results for increasing size Epinions graphs of GA, SA, HS, Greedy and New Method measured as the ratio of influence spread to the K-seed size.

increase the number of nodes of the graph. We work with 10 graphs from 1000 nodes to 10000 nodes increasing a thousand nodes each time. Table 6.2 shows spent time to get the minimum k-seed nodes. Being the matrix little (1000 nodes) the spent time for execution with Greedy is already high enough in comparison with our method. When the size increase, the Greedy algorithm is getting slower. In contrast, our method executes the experiment fast under 2.5 seconds with the biggest matrix.

Sizes	New method	Greedy
1000	0.065 sec	3.586 sec
2000	0.119 sec	35.938 sec
3000	0.182 sec	> 5 min
4000	0.363 sec	> 5 min
5000	0.446 sec	> 5 min
6000	0.706 sec	> 5 min
7000	1.702 sec	> 5 min
8000	1.914 sec	> 5 min
9000	2.012 sec	> 5 min
10000	2.435 sec	> 5 min

Table 6.2: Comparison of speed using matrix of different sizes

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Thesis general goals where proposed in the Introduction.Three were the proposed goals of this Thesis. All of them are reached as explained in next Sections 7.1.1,7.1.2 and 7.1.3.

7.1.1 Trust

- Proposed work with Artificial Neural Networks is rather straightforward compared with other works in the literature: feature vectors are built as reputation vectors from the trustworthiness assertions of trusted users on the target trustee. Then, trust prediction becomes a two class classification problem that can be directly solved by training on the feature dataset. The approach gives good results, though the problem suffers from the extreme imbalance of the classes. On a benchmark study on the Epinions data, we achieve good Precision results for both classes and encouraging results for the Recall of the less frequent Distrust class. Distrust Recall improves when applying SMOTE preprocessing. Interestingly, we found that classical ANN classifiers (MLP and RBF) achieved better results on the most difficult issue of Distrust Recall, pointing to some bias of the SVM. A rough comparison with state-of-the-art approaches is favorable to our definition of reputation features.

- The second work proposed in this research line is a Trust prediction system based on reputation features obtained from the trust values of witness users. The system has been demonstrated over two benchmark trust databases in the public domain, extracted from the Epinions and Wikipedia sites. We tested two kinds of features, raw reputation vectors and probabilistic reputation features. The former features lead to classification systems that are heavily influenced by the database imbalance. Attempts to improve results applying a SMOTE approach do not improve the overall accuracy, but provide improvements on the minority class, the Distrust class. The probabilistic reputation features provide excellent results reaching 100% in both databases. The major inconvenient is that their coverage of the Wikipedia database is small due to many singularities in the computation of the probabilities. The resiliency of the classifiers based on the probabilistic features to social network growth has been assessed by performing training experiments with very small training datasets, achieving optimal results even then.

7.1.2 Recommendation Systems

- We propose the application of regression ELM to build a breadmaker recommender system which is an instance of the social intelligence in the Internet of Things framework of the SandS European Project. We have proposed a dataset synthesis procedure to carry the experimental validation of the system, due to the lack of real-life data. The experimental results are quite good for the direct mapping from recipes to satisfaction evaluations, but not so good for the inverse mapping, which will require a more careful tuning for the practical application.

- We propose methods for Recommendation Systems in Social Networks based on Colaborative Filtering. Those methods follow the way of using information from the Web of Trust built from users trust values and the way of the intelligent layer of the system. To represent this idea we propose a feature extraction based on similarities among users. Better results are obtained with the Web of Trust provided by user explicit statements.

7.1.3 Influence Maximization

- A new heuristic search method for Influence Maximization (IMH) is proposed in this Thesis. It relaxes the influence spread search by considering only paths of length 1, however it is guaranteed to terminate covering the entire graph. We provide comparison with other well known heuristics, namely Simulated Annealing, Genetic Algorithm, Harmony Search and Greedy Search algorithm over a collection of synthetic and real social network graphs. The proposed heuristic method compares with the Greedy

Search algorithm, providing the largest influence spread results with minimum seed nodes, improving other heuristics. Moreover, the proposed IMH method is always faster than Greedy algorithm.

7.2 Future Work

Next Sections 7.2.1, 7.2.2 and 7.2.3 describe the future work plan for the three main areas shown in this Thesis. Although different work lines are presented, the goal is to join all of them as far as possible.

7.2.1 Trust

Further work will involve testing new approaches to cope with imbalanced datasets, as well as other feature definitions. Also, other knowledge modeling approaches based on experience may be tested on this data [7, 129, 139]. Future work will be carried out within the framework of the SandS European project [1], where the authors are working towards the development of a social network of home appliance users (named Eahoukers in the project). The goal is that the eahoukers benefit from the socially generated knowledge to deal with the home appliances in a domestic environment. Trust prediction is relevant for SandS in three ways: (a) for the identification of rogue users that may try to sabotage competitors' appliances, (b) to assess the quality of the recommendations coming from specific users, and (c) to build the consensus between users. The Trust prediction system presented in this Thesis will be useful to assess the recommendations from other users in the elaboration of appliance use recipes. The final system will be a recommendation system enhanced by Trust prediction.

7.2.2 Recommendation Systems

Further work will be addressing the computational experiments on real life data, once it is available from the breadmaking experiments being carried out by other project partners. It is also possible to open the experimentation to the general public by the implementation of a social network of breadmaking. This implementation would be a real test of the idea of subconscious social intelligence, which in this setting will encompass the application of both direct an inverse mappings.

According to product recommendation, we want to exploit further the sparse computational methods in order to propose improvements on Recommendation Systems.

[1] `http://www.sands-project.eu/`

7.2.3 Influence Maximization

Future works will address the formulation of methods, able to deal with very large social networks, with node sites in the order of hundreds of million nodes as well as the algorithm is updated to find the node v^* with maximal influence in one step $\sigma_1(\{v\}, A_t)$, i.e. paths of lenghts greater or equals than 1.

Appendix A

EXPERIMENTAL DATABASES

Proposed experimentation and contributions along this Thesis is performed handling different databases. Often, the approaches and algorithms from Computational Intelligence have the validity given by the datasets where they have been validated. Therefore, detailing these datasets is part of the process of establishing trust on the algorithms. Both, real-life databases and syntethic databases have been used for validation of the approaches proposed. Section A.1 presents the real-life datasets, and Section A.2 presents the synthetic databases.

A.1 Real databases

There are several online and public repositories which allow to download different databases that can be useful for sytem validation allowing reproducibility of the results. For this Thesis, we are interested in Social Networks databases. Examples of repositories are KONECT, Arizona State University's repository and SNAP.

- KONECT (the Koblenz Network Collection) [1] is s a project to collect large network datasets of all types in order to perform research in network science and related fields. KONECT contains 235 network datasets of various types (directed, undirected, bipartite, weighted, unweighted, signed and rating networks). Those networks belong to many different areas such as social networks, hyperlink networks, authorship networks, physical networks, interaction networks and communication networks. For each dataset an analysis of it is given. Arizona State University also provides some social networks datasets [2] about product reviews and trust among users.

[1] `http://konect.uni-koblenz.de/`

[2]`http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm`

- SNAP (Stanford Network Analysis Platform) [3] is a general purpose network analysis and graph mining library. There are many dataset about different areas. Some statiscs are given for each dataset.

Epinions Trust and Rating Network and Wikipedia Voted Network are described in the following subsections, respectively. Epinions networks datasets are used in Chapters 3 (Trust), 4 (Recommendation Systems) and 5 (Influence Maximization) for experimental validation. Experimentation based on Wikipedia Voted Network in provided in Chapter 3 (Trust).

A.1.1 Epinions

The Epinions site[4] is a social site where user provide reviews of products of any kind, from music to perfumes or construction hardware. These reviews are the base for the establishment of trust relations between users.

Web of Trust Network In Epinions dataset, Trust is specified by a binary variable taking values in {-1,1}: a user can choose to trust (1) another or not (-1). Negative trust values are not published in the web service, but the anonymized dataset provided for experimentation, contains also negative Trust values. This dataset has 841,372 data samples. Each sample is a triplet composed of two user indexing numbers (no personal data of any form is included) and the binary Trust value of the first user on the second user. Therefore, Trust relations define a directed graph, with weighted edges. The database is heavily unbalanced: 85.3% of instances show positive trust (717,667 triplets), versus 14.7% of negative trust instances (123,705 triplets). Giving a formal description, a directed graph $G_u(U,T)$ represents users as nodes and trust links as edges. A link will exist from user u_i to another user u_j if the first user trusts/distrusts the second one. On the dataset, each link among two users is represent in this way: $< u_i, u_j >$.

Ratings Network The graph of the rating dataset could be defined as:

$$G_r\left(U \cup (C \times I), E, R\right),$$

where nodes are users U and items I (products), and edges in E are links between users and items, i.e. $E \subset U \times (C \times I)$. Users and items become linked when a user gives a rating about an item. Edges have a a weight corresponding to the rating. Thus, the information on the dataset is represented by the triplet $\langle (u,(c,i)), r \rangle$

[3] `http://snap.stanford.edu/`
[4] `http://www.epinions.com/`

means that a user u_i gives a rating r to the item i of the category c. There are 27 categories, because of the importance of the domains in Recommendation Systems, we define the set of whole items as $I = \{I_1,...,I_{27}\}$ according with the number of categories. Having the trust network graph and having the 27 rating weighted graphs, we are able to build classifiers. Each I_i is the rating matrix for the items of the category i. Ratings are values in the range $\{1..5\}$. Zero means that no rating is given for an item.

A.1.2 Wikipedia Voted Network

Wikipedia is a free encyclopedia built by crow-sourcing efforts from volunteers around the world. A small part of Wikipedia contributors are administrators, who are users with access to additional technical features supporting Wikipedia maintenance. In order for a user to become an administrator a Request for adminship (RfA) is issued and the Wikipedia community via a public discussion or a vote decides who to promote to adminship. The actual database[5] employed in this paper has the following format *per* voting record:

- E: did the elector result in promotion (1) or not (0)
- T: time election was closed
- U: user id (and screen name) of editor that is being considered for promotion
- N: user id (and screen name) of the nominator
- V: vote(1: support, 0: neutral, -1: oppose) user_id time screen_name

For the work in this paper, the trust triplet (A, B, t_{AB}) is built from three attributes: voting user, voted user and the vote value. Regarding the vote value, we are interested in two out of of three possible vote values: 1 (support) and -1 (oppose). We ignore the vote "0". For this reason, we reorganize the database as follows: each row has the three attributes mentioned before: $[UserA, UserB, vote]$. In summary, we obtain a social network trust database containing 103,591 instances (78.83% for class “1” and 21.17% for class “-1”).

A.2 Synthetic databases

These databases are not found in any repository. They have been created by ad hoc programs for specific experiments. Experimentation based on a data base for

[5]http://snap.stanford.edu/data/wiki-Elec.html

recipe generation is reported in Chapter 4 (Recommendation Systems) while experimentation based on a synthetic social network is reported in Chapter 5 (Influence Maximization).

A.2.1 Non Linear Generation of Recipes

The system reported in Chapter 4 aims to the autonomous intelligent recipe generation for household appliances. The design faces the obstacle of the lack of actual data of household appliances working in a controlled environment, therefore to show the workings of the proposed system we resort to synthetic datasets. We generate a dataset of 100,000 instances of recipe satisfaction pairs taking into consideration the following:

- We consider that there is a non-linear map that models the contribution from each recipe parameter to each satisfaction parameter value. For the experimental work, we have created arbitrary maps which are shown in Figure A.1. Each entry (i, j) in the table is a map randomly generated relating recipe parameter r_i with each satisfaction parameters_j. Thus, we have 40 models.

- We consider that the satisfaction value is a linear combination of the contributions of the recipe parameters. If a_{ij} is the satisfaction value in variable s_j induce from a given value from recipe variable r_i, then:

$$s_j = \frac{\sum_{r=1}^{10} \alpha_i * a_{ij}}{r'}$$

  using α_i as weighting factor of recipe variable and being r' the normalizing value to obtain the weighted average. If result has decimal part, we round the number to the nearest natural one. We have choosen the value of the α_i arbitrarily for the experiments reported here.

Once we have models, we are able to generate a synthetic dataset according with models. To generate a database we generate randomly 100,000 instances of $[r_1, ..., r_{10}]$ then, we use the models to calculate the satisfaction. As example, we show in A.1 the first row of the dataset.

	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	s1	s2	s3	s4
#1	0.6	0.3	0.4	0.2	1	0	0.4	0.8	0.3	0.4	2	2	2	2
#i	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table A.1: Example of the content of Dataset

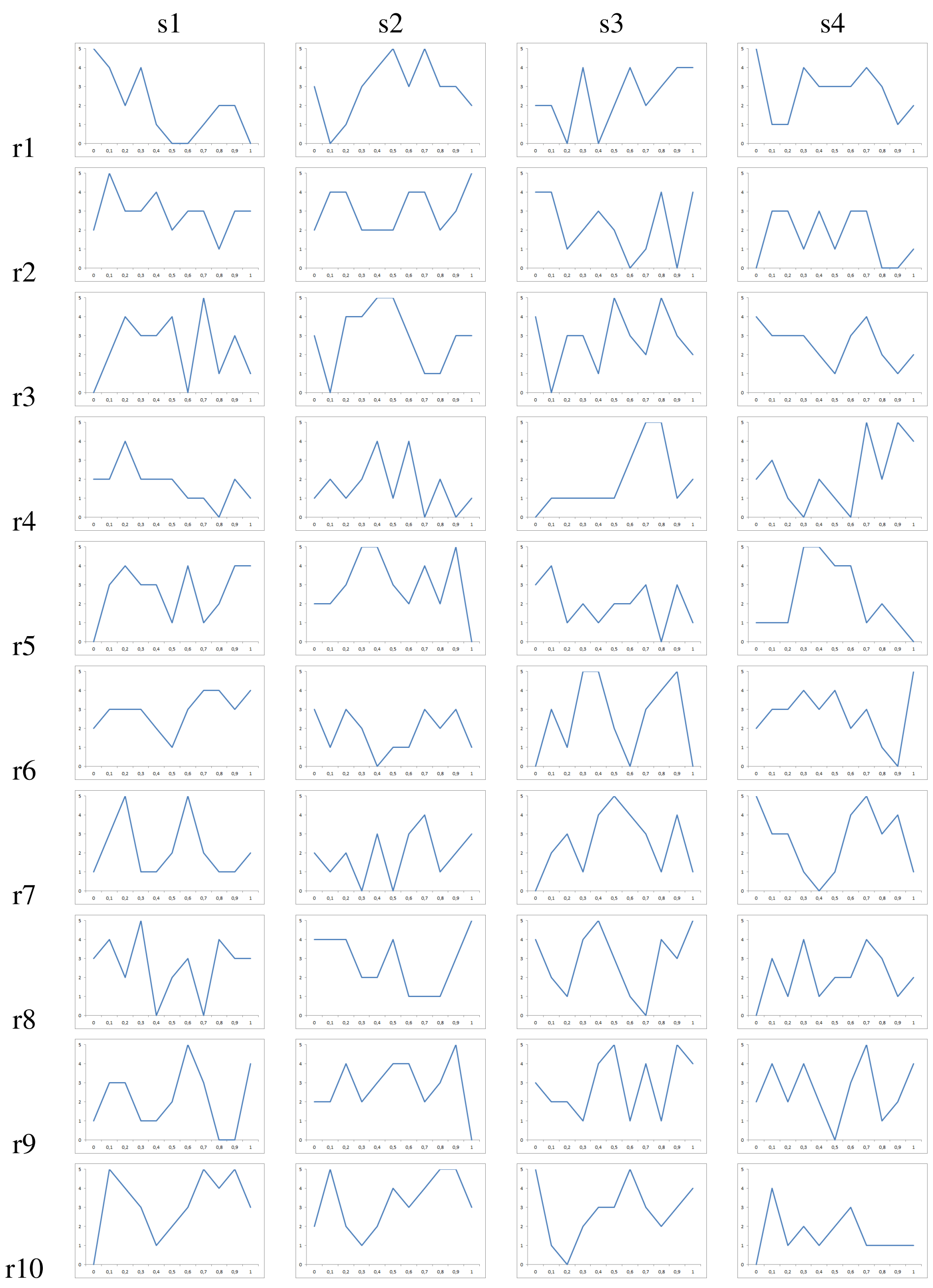


Figure A.1: Maps specifying the influence of the recipe paramters into the satisfaction parameters. Each each enty relates a pair of recipe-satisfaction variables. Each plot has horizontal axis in the range $[0,1]$ and the vertical axis in the set $\{0,1,2,3,4,5\}$.

A.2.2 Influence Maximization

A social network is defined as a directed graph $G(V,E)$ where V is the set of nodes that represents the set of users and E the set of edges that represents the set of relationships among users. Given a directed graph $G(V,E)$ edges are weighted by $w_{ij} = 1/degree_{in}(j)$. Now the definition of the graph is $G(V,E,W)$ where W is the set of weights that are values in the range $[0..1]$. Some of the experiments reported here are done on synthetic graphs. To build such graphs we generate the random weights of a complete graph, determining the in-degree of each node. Thus, from the a probabilistic adjacency matrix we generate graph instances $G'_i(V,E')$ where edges will appear according with the probabilistic weight. An weighted edge with a high weight has more probability to appear in the sampled graph G'_i. We define $G' = \{G'_1,...,G'_n\}$ as the set of sample graphs used to test the diverse Influence Maximization heuristics. For the experiments referred below, the Independet Cascade Model influence propagation estimation consists in the average of the propagations over the set of sample graphs.

Appendix B

COMPUTATIONAL METHODS

This Appendix gathers the description of standard computational methods used as building blocks of the solutions offered to the problems tackled in several domains. The implementations used are standard ones found in Matlab distributions. We have organized the Appendix according to the problems tackled in the referred chapters, so that the Section B.1 contains the description of the methods used for Trust prediction, Section B.2 contains the methods used for recipe recommendation, and Section B.3 contains methods used in the solution of the Influence Maximization problem.

B.1 Computational methods for Trust prediction

In this section we introduce the formal definition of the classifier building methods that have been applied in the experiments of Trust prediction based on reputation features reported in Chapter 3. We have applied two Artificial Neural Network approaches, the Multi-Layer Perceptron and the Radial Basis Function Network, and a statistical classifier, the well-known Support Vector Machines. We also describe the SMOTE data preprocessing aiming to cope with imbalanced classification problems.

B.1.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) is an artificial neural network (ANN) consisting of multiple layers, allowing to solve complex discrimination problems that are not linearly separable, i.e. there is no single hyperplane separating the samples of the classes, which is the main limitation of the Perceptron (also called Single Perceptron). The MLP can be totally or locally connected. In the first case each output of a neuron of layer "i" is input to all neurons of layer "i + 1" while every neuron in the second layer "i" is a number of input neurons (region) layer "i + 1". We restrict our presentation of this classifier to train the weights of the MLP for a two class problem. Let the instantaneous error E_p be defined as:

$$E_p(\mathbf{w}) = \frac{1}{2}\left(y_p - z_K(\mathbf{x}_p)\right)^2, \tag{B.1}$$

where y_p is the p-th desired output y_p, and $z_K(\mathbf{x}_p)$ is the network output when the p-th training exemplar $\mathbf{x}_p$ is input to the MLP composed of K layers, whose weights are aggregated in the vector $\mathbf{w}$. The output of the j-th node in layer k is given by:

$$z_{k,j}(\mathbf{x}_p) = f\left(\sum_{i=0}^{N_{k-1}} w_{k,j,i} z_{k-1,i}(\mathbf{x}_p)\right), \tag{B.2}$$

where $z_{k,j}$ is the output of node j in layer k, N_k is the number of nodes in layer k, $w_{k,j,i}$ is the weight which connects the i-th node in layer $k-1$ to the j-th node in layer k, and $f(\cdot)$ is the non-linear sigmoid function, which has a simple derivative:

$$f'(\alpha) = \frac{df(\alpha)}{d\alpha} = f(\alpha)(1 - f(\alpha)). \tag{B.3}$$

The convention is that $z_{0,j}(\mathbf{x}_p) = \mathbf{x}_{p,j}$. Let the total error E_T be defined as follows:

$$E_T(\mathbf{w}) = \sum_{p=1}^{l} E_p(\mathbf{w}), \tag{B.4}$$

where l is the cardinality of X. Note that E_T is a function of both the training set and the weights in the network. The back-propagation learning rule is defined as follows:

$$\Delta w(t) = -\eta \frac{\partial E_p(\mathbf{w})}{\partial w} + \alpha \Delta w(t-1), \tag{B.5}$$

where $0 < \eta < 1$, which is the learning rate, the momentum factor α is also a small positive number, and w represents any single weight in the network. In the above equation, $\Delta w(t)$ is the change in the weight computed at time t. The momentum term is sometimes used $(\alpha > 0)$ to improve the convergence of the algorithm by smoothing the adaptation changes proposed by the simple gradient descent. The algorithm defined by equation (B.5) is often termed as *instantaneous back-propagation* because it computes the gradient based on a single training vector. Another variation is *batch back-propagation*, which computes the weight update using the gradient based on the total error E_T. To implement this algorithm we must give an expression for the partial derivative of E_p with respect to each weight in the network. For an arbitrary weight in layer k this can be written using the Chain Rule:

$$\frac{\partial E_p(\mathbf{w})}{\partial w_{k,j,j}} = \frac{\partial E_p(\mathbf{w})}{\partial z_{k,j}(\mathbf{x}_p)} \frac{\partial z_{k,j}(\mathbf{x}_p)}{\partial w_{k,j,i}}. \tag{B.6}$$

B.1.2 Radial Basis Function

Radial Basis Function networks (RBF) [29] are a type of artificial neural networks to calculate the output function according to the distance from a point called the center. As well as multilayer perceptrons, RBFs are a universal approximators too. The radial basis function is a function whose output is an exponential function of the Euclidean distance of an input vector x with respect to a center c. Each neuron of the input layer has a radial basis function output and a weight. The output pattern enters an output neuron which makes the summation of all inputs and gives an output as a result. The RBF networks have a rigid construction of three layers: input layer, hidden layer and output layer (unlike other backpropagation networks). Any arbitrary function $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ can be approximated by a map defined by a RBF network with a single hidden layer of K units:

$$\hat{g}_\theta(\mathbf{x}) = \sum_{j=1}^{K} w_j \phi\left(\sigma_j, \left\|\mathbf{x} - \mathbf{c}_j\right\|\right), \tag{B.7}$$

where θ is the vector of RBF parameters including $w_j, \sigma_j \in \mathbb{R}$, and $\mathbf{c}_j \in \mathbb{R}^n$; let us denote $\mathbf{w} = (w_1, w_2, \ldots, w_p)^T$, then the vector of RBF parameters can be expressed as $\theta^T = \left(\mathbf{w}^T, \sigma_1, \mathbf{c}_1^T, \ldots, \sigma_K, \mathbf{c}_K^T\right)$. Each RBF $\phi(\cdot)$ is defined by its centre $\mathbf{c}_j \in \mathbb{R}^n$ and width $\sigma_j \in \mathbb{R}$, and the contribution of each RBF to the network output is weighted by w_j. The RBF $\phi(\cdot)$ is a non-linear function that monotonically decreases as $\mathbf{x}$ moves away from its centre $\mathbf{c}_j$. The most common RBF used is the isotropic Gaussian:

$$\hat{g}_\theta(\mathbf{x}) = \sum_{j=1}^{p} w_j \exp\left(-\frac{\left\|\mathbf{x} - \mathbf{c}_j\right\|^2}{2\sigma_j^2}\right).$$

The RBF network can be thought as the composition of two functions

$$\hat{g}_\theta(\mathbf{x}) = W \circ \Phi(\mathbf{x}),$$

the first one implemented by the RBF units $\Phi : \mathbb{R}^n \to \mathbb{R}^K$ performs a data space transformation which can be a dimensionality reduction or not, depending on whether $K > n$. The second function corresponds to a single layer linear Perceptron W :

$\mathbb{R}^K \rightarrow \mathbb{R}$ giving the map of the RBF transformed data into the class labels. This approach corresponds to the RBFN architecture in the experiments. Other approaches perform a gradient descent of the error function involving all the parameters of the network simultaneously, this approach corresponds to the RBFC architecture in the experiments below.

B.1.3 Support Vector Machines

Support Vector Machines (SVM) [131] look for the set of support vectors that allow to build the optimal discriminating surface in the sense of providing the greatest margin between the classes[1]. SVM separates a given set of binary labelled training data with a hyperplane that is maximally distant from the two classes (known as the maximal margin hyperplane). The objective is to build a discriminating function using training data that will correctly classify new examples $(\mathbf{x}, y)$. When no linear separation of the training data is possible, SVMs can work effectively in combination with kernel techniques using the kernel trick, so that the hyperplane defining the SVMs corresponds to a non-linear decision boundary in the input space that is mapped to a linearised higher-dimensional space [131]. In this way, the decision function can be expressed in terms of the support vectors only:

$$f(\mathbf{x}) = sign\left(\sum \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + w_0\right),$$

where $K(.,.)$ is a kernel function, α_i is a weight constant derived from the SVM process and the $\mathbf{s}_i$ are the support vectors [131].

Given feature training vectors $\mathbf{x}_i \in R_n, i = 1, \ldots, l$ of samples from the two classes, and a vector $y \in R^l$ such that $y_i \in \{-1, 1\}$ is the class label of sample $\mathbf{x}_i$, in our case, for example, untrusted connections were labelled as -1 and trust connections as 1. To maximize the classification margin, the SVM approach solves the following optimization problem:

$$\min_{w,b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \xi_i$$

subject to $y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq (1 - \xi_i)$, $\xi_i \geq 0, i = 1, 2, \ldots, n$. The dual optimization problem is

$$\min_{\alpha} \frac{1}{2}\alpha^T\mathbf{Q}\alpha - \mathbf{e}^T\alpha,$$

subject to $\mathbf{y}^T\alpha = 0$, $0 \leq \alpha_i \leq C$, $i = 1, \ldots, l$, where $\mathbf{e}$ is the vector of all ones,

[1]The implementation used in most studies is included in the libSVM (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`) software package.

$C > 0$ is the upper bound on the error, $\mathbf{Q}$ is an $l \times l$ positive semi-definite matrix, $Q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ is the kernel function that describes the behavior of the support vectors. Here, the training vectors $\mathbf{x}_i$ are mapped into a higher (maybe infinite) dimensional space by the function $\phi(\mathbf{x}_i)$. C is a regularization parameter used to balance the model complexity and the training error.

Depending on the kernel function chosen, different kinds of SVM are defined with different performance levels, and the choice of the appropriate kernel for a specific application is a difficult task. In this study two different kernels were tested: the linear and the radial basis function (RBF) kernel. The linear kernel function is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = 1 + \mathbf{x}_i^T \mathbf{x}_j,$$

this kernel shows good performance for linearly separable data. The RBF kernel is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}).$$

This kernel is best suited to deal with data that have a class-conditional probability distribution function approaching the Gaussian distribution [22]. The RBF kernel is largely used in the literature because it corresponds to the mapping into an infinite dimension feature space, and it can be tuned by its variance parameter σ.

B.1.4 SMOTE

Umbalanced datasets, where the number of samples for each class is different, so that one may be greatly underrepresented while other is overrepresented, poses a number of problems for classification system building and validation. The trivial solution of making the decision only on the favor of the most frequent class can provide high accuracy while completely failing for the minority class. Often the minority class is some specific condition that needs to be detected, so that the cost of failing in the detection may be much greater than for the majority class. Hence, some special strategy is required to improve results on the minority class, and specific performance measures must take into account this phenomena. For instance, the True Positive Ration (TPR) can be a better measure than the conventional classification accuracy.

One of the strategies for classification building aimed to improve performacne is the Synthetic Minority Over-sampling Technique (SMOTE) [28]. which consists in the generation of new samples of the minority (less frequent) class in order to obtain a more balanced representation of the classes. Instead of performing a re-sampling with replacement from the original database, which only introduces repetitions of the already sampled points in the feature space, SMOTE performs interpolation processes in order to generate new sampling points in feature space, because mere replication of sampling points do not alter the decision boundary. As a side effect, we are not sure of the actual discrimination functions, so that SMOTE might be introducing spurious class boundary displacements.

SMOTE works in feature space by the following process for each minority class sample (or a random subset of them) $\mathbf{x}_0$:

1. Select its k minority class nearest neighbors $N_k(\mathbf{x}_0)$
2. Draw the line between each $\mathbf{x}_i \in N_k(\mathbf{x}_0)$ and $\mathbf{x}_0$.
   (a) compute the new sample picking a random position in this line (random linear interpolation):
   $$\mathbf{x}_i^* = \alpha \mathbf{x}_0 + (1-\alpha)\mathbf{x}_i,$$
   where $\alpha \sim U(0,1)$ is a random number between 0 and 1.
3. Add the generated samples to the minority class training data.

This process can be tuned specifying the number of nearest neighbors. SMOTE can be used in combination with majority class under-sampling (removing samples). Notice that SMOTE may "fill the gaps" in data distributions that show disperse connected regions.

B.2 Computational methods on Recommendation Systems for Recipe Generation

In Chapter 4 we use Extreme Learning Machine (ELM) [73, 72] in order to obtain a quick solution to the training problem of the recipe evaluation method. ELM provides a simple learning algorithm for Single-Hidden Layer Feedforward Neural network (SLFN), reducing it to the Moore-Penrose generalized inverse providing the minimum Least-Squares solution of general linear systems. The hidden layer weights are given by random sampling, obviating the need of lengthy and difficult to tune gradient descent learning of the input to hidden layer weights.

B.2.1 Basic ELM

For N arbitrary distinct samples $(\mathbf{x_i}, t_i)$, where input variables are $\mathbf{x_i} = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbb{R}^n$ and target values are $\mathbf{t_i} = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in \mathbb{R}^m$. The training of a standard

SLFN with N hidden neurons and activation function $g(x)$ is mathematically modeled as solving the following equation to estimate the value of the SLFN parameters:

$$\sum_{i=1}^{hn} \beta_{\mathbf{i}} \cdot g(\mathbf{w_i} \cdot \mathbf{x_j} + b_i) = \mathbf{t_j},\ j = 1, \cdots, N. \tag{B.8}$$

where $\mathbf{w_i} = [w_{i1}, w_{i2}, \dots, w_{in}]^T$ is the weight vector connecting the i-th hidden neuron and the input neurons, $\beta_{\mathbf{i}} = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$ is the weight vector connecting the **i**-th hidden neuron and the output neurons, and b_i is the threshold of the ith hidden neuron. $\mathbf{w_i} \cdot \mathbf{x_j}$ denotes the inner product of $\mathbf{w_i}$ and $\mathbf{x_j}$ and hn is the number of hidden neurons. The activation function can be the identity for the so-called linear kernel approaches, sigmoid for the Multilayer Perceptron approaches, or Gaussian for Radial Basis Function approaches.

The equation (B.8) can be written in matrix form as:

$$\mathbf{H}\beta = \mathbf{T}, \tag{B.9}$$

where $\mathbf{H}$, of size $N \times hn$, is the output matrix resulting of the SLFN hidden layer activated by the input samples, β is the output weight matrix of size $hn \times m$, and $\mathbf{T}$ is the target matrix with size $N \times m$. Training of SLFN is accomplished by computing the least-squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$, given by $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{H}$.

B.3 Computational methods for Influence Maximization

This section describes the conventional optimization algorithms that have been applied to compare with our specific proposals for Influence Maximization (IM) in Chapter 6. We retain the notation associated with the IM statement in the description of the algorithms, assuming that the objective function is the influence spread denoted by $\sigma(S)$. The straightforward approach is to perform an exhaustive search, where all possible combinations of IM-Seed node sets are evaluated. In a exhaustive systematic procedure, a search tree would be traversed where each node correspond to a solution. For NP-complete problems, such as IM, this and any other other exact global search methods are unfeasible. Therefore, we must resort to heuristic approaches, which provide suboptimal but good results in a reasonable time, such as the greedy search [77], which has quadraticly growing computational time depending on the number of nodes. In this Appendix we describe the well known heuristic algorithms: Simulated Annealing, canonical Genetic Algorithm, Harmony Search (HS). and Greedy Search (GS) algorithm. The objective function to maximize is the influence spread, i.e. $\max_S \{\sigma(S)\}$. For a given candidate solution, $\sigma(S)$ is computed by a ICM Monte Carlo approximation specified in Chapter 6. Often the heuristic approaches need some codification of the candidate solution. The common codification is a vector of binary valued components such that $S_v = 1$ iff node v has been included in the IM-Seed solution S. For convenience, the components are not explicit in most algorithm presentations. Note that the vector space dimension is the size of the node set V.

B.3.1 Simulated annealing

Simulated Annealing (SA) was proposed [79, 78, 109] as a general purpose probabilistic metaheuristic for the global optimization of non-convex functions (often non-differentiable) in large search spaces. It is a nature inspired technique, mimicking the heating and cooling process followed to obtain some materials, for example high quality steel. SA was shown to provide the global optima under very strict conditions, and provides good approximations in a reasonable computational time. It generates a sequence of solutions whose objective function values converge to the global optimum value. A temperature parameter allows to control the search. The temperature parameter typically starts off high and is slowly "cooled" or lowered in every iteration. At high temperatures, the process accepts state (solution) changes that deteriorate the objective function to a limited extent. This prevents the search from getting trapped in local optima at early stages. At decreasing temperatures, it becomes a hill climbing algorithm that only accepts improvements of the objective function. Algorithm B.1 presents a pseudocode of the Simulated Annealing, where S is the current solution, which is a binary codification of the nodes, one bit per node which is on if the node belongs to the IM-Seed influence set. S_{new} is a candidate new solution generated by the $neighbour()$ function from the current solution, by randomly changing one of the components to its opposite value. The acceptance probability $P_a(E(S), E(S_{new}), T)$ is a function of the temperature T and the difference of the objective function that is the influence spread of the IM-Seed set $\sigma(S)$ of the current and new candidate states.

B.3.2 Genetic algorithm

Genetic Algorithms (GA) [71, 53] were proposed as a general method for solving both constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution. The algorithm iteratively generates a population $\mathbf{S}_t$ of individual candidate IM solutions S_i by the application of genetic operators, crossover and mutation, to the chromosomes in the previous generation $\mathbf{S}_{t-1}$. At each step, the genetic algorithm randomly selects individuals

Algorithm B.1 Simulated Annealing algorithm pseudocode

```
1. $s = s_0$
2. For $k = 0 : k_{max}$ (exclusive):
    (a) $T \leftarrow temperature(k/k_{max})$
    (b) Pick a random neighbor, $S_{new} \leftarrow neighbour(S)$
    (c) If $P_a(\sigma S, \sigma(S_{new}), T) > r$, $r \sim U(0,1)$
    (d) $S \leftarrow S_{new}$
3. Return $s$
```

from the current population and uses them as parents to produce the children for the next generation. Successive population generations "evolve" toward an optimal solution [2]. In GAs, the number of generations and the population size are critical parameters, in the experiments they are set proportional to problem complexity, i.e. number of nodes in the social network, so that effective computation time grows linearly. A typical GA is as follows: Start with a randomly generated population $\mathbf{S}_0 = \{S_1, \ldots, S_N\}$, each chromosome is a candidate solution encoded as a binary valued vector. Calculate the fitness $\sigma(S_i)$ of each chromosome S_i in the population S_t, the fitness allows to compute the selection probability. At each generation, create N offspring by crossover and mutation. For crossover, randomly draw a pair of parent chromosomes from the current population, according to an empirical probability distribution which is a increasing function of chromosome fitness. Selection is done “with replacement," meaning that the same chromosome can be selected more than once to become a parent. According to crossover probability p_c exchange the bits of the pair randomly to form two offsprings. If no crossover takes place, form two offspring that are exact copies of their respective parents. For mutation: Mutate each bit the two offspring at each locus with probability p_m (the mutation probability), and place the resulting chromosomes in the new population. Finally, replace the current population with the new population applying an elitist rule that preserves the best chromosomes of the previous population.

B.3.3 Greedy Search

A straightforward Greedy Search (GS) algorithm achieves a good deterministic approximation to the optimum solution of IM due to non-negativity, monotonicity and submodularity of $\sigma(\cdot)$ [77]. The influence spread is a set function $\sigma : 2^V \rightarrow \mathbb{R}$,

[2] `http://www.mathworks.com/discovery/genetic-algorithm.html`

Algorithm B.2 Proposed New Method's algorithm

```
1. Given weighted social graph $G = (V, E, W)$
2. $S_0 = \varnothing$
3. Iterate until $\triangle\sigma(v^*) = 0$
    (a) Compute $\triangle\sigma(v) = \{\sigma(S_t \cup \{v\}) - \sigma(S_t)\}$ for all $v \in V - S_t$
    (b) $v^* = \arg\max_{v \in V} \{\triangle\sigma(v)\}$
    (c) $S_{t+1} = S_t \cup \{v^*\}$
    (d) $t = t + 1$
4. Return $S_t$
```

which is non-negative, i.e. $\sigma(S \geq 0)$ for all $S \subseteq V$, monotone, i.e. $\sigma(S) \leq \sigma(T)$ for all $S \subseteq T$, and submodular, i.e. $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(T \cup \{v\}) - \sigma(T)$ for all $S \subseteq V$ and $v \in V$. For this kind of objective functions, it is known that the greedy search would achieve a solution which is at least 63% of the global optimum [77].

There are two main steps in the Greedy Search specified . First, we compute the influence spread $\sigma(v_i)$ for all nodes in the graph. Second, we compute the iterative solution generation specified in Algorithm B.2. At the beginning, the solution IM-Seed node set S is empty. At each step of the greedy search iteration, we look for the node v^* that provides the greatest increase in the influence $\triangle\sigma(v)$, until there is no increase in influence whatever the node chosen, i.e. $\triangle\sigma(v^*) = 0$. For the ease of notation, we assume that $\sigma(\emptyset) = 0$.

B.3.4 Harmony Search

Harmony Search (HS) [48, 85] is a global search heuristic for global optimization of non-convex functions inspired in the musical improvisation process. It has been successfully applied to a variety of problems alone or hybridized with other methods. Some recent applications follow. The muzzle velocity of an electromagnetic railgun was optimized applying HS on the variables ranked by a previous orthogonal design method [27]. The optimization of an electrical transformer design [65] was achieved by a multi-objective HS endowed with crowding distance ranking and control parameter tuning by a Ricker map. Parameter tuning by HS of the proportional integral controllers of a distributed power generation system overcomes genetic algorithms and gradient descent approaches in [5]. A differential HS compares favorably with particle filter approaches for face tracking in video imagery [46]. A quasi-oppositional HS improves over fuzzified internal control models [119, 125] for the control of several standard power generation systems. The design of ensemble classifiers using HS for classifier-as-feature selection is discussed in [37]. An improved HS is successfully applied or thrust optimization of dynamic positioning of off-shore oil drilling platforms [137]. The variety of these applications show the versatility of the HS approach.

In the HS algorithm, each musician (= decision variable) plays (=generates) a note (= a value) looking for the best harmony (= global optimum) all together. Harmony is defined by some objective function which we try to optimize (minimize or maximize). Since its initial proposal, there have been variations and improvements of HS in the literature to be applied in different contexts, for instance: discrete design variables, and global optimization through competition [50, 49, 96, 84, 108]. The optimization problem is specified as follows:

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) \,|\, \mathbf{x} = [x_i \in X_i; i = 1, ..., N] \right\} \tag{B.10}$$

where $f(\mathbf{x})$ is the objective function that corresponds to the musical harmony, X_i is the range set of design variable x_i (we consider continuous design variables), and N is the number of design variables. The "harmony memory" (HM) matrix, equation B.11, is the central data structure of the algorithm containing the current state of the search, given by the preserved harmony vectors plus their harmony value $f(\mathbf{x})$,

$$HM = \left[\begin{array}{ccc} \mathbf{x}^1 & \dots & \mathbf{x}^{HMS} \\ f\left(\mathbf{x}^1\right) & \dots & f\left(\mathbf{x}^{HMS}\right) \end{array} \right], \tag{B.11}$$

ordered so that $f\left(\mathbf{x}^j\right) \geq f\left(\mathbf{x}^{j+1}\right)$, so that $f\left(\mathbf{x}^{HMS}\right)$ is the worst harmony value.

Bibliography

[1] Kimberly S Adams and Sandra L Christenson. Trust and the family 'school relationship examination of parents' teacher differences in elementary and secondary grades. *Journal of School Psychology*, 38(5):477 – 497, 2000.

[2] W.J. Adams, G.C. Hadjichristofi, and IV Davis, N.J. Calculating a node's reputation in a mobile ad hoc network. In *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International*, pages 303 – 307, april 2005.

[3] Sheikh I. Ahamed, Munirul M. Haque, Md. Endadul Hoque, Farzana Rahman, and Nilothpal Talukder. Design, analysis, and deployment of omnipresent formal trust model (ftm) with trust bootstrapping for pervasive environments. *Journal of Systems and Software*, 83(2):253 – 270, 2010. Computer Software and Applications.

[4] Samah Al-Oufi, Heung-Nam Kim, and Abdulmotaleb El Saddik. A group trust metric for identifying people of trust in online social networks. *Expert Systems with Applications*, 39(18):13173 – 13181, 2012.

[5] M.N. Ambia, H.M. Hasanien, A. Al-Durra, and S.M. Muyeen. Harmony search algorithm-based controller parameters optimization for a distributed-generation system. *Power Delivery, IEEE Transactions on*, 30(1):246–255, Feb 2015.

[6] B. Apolloni, M. Fiasch, G. Galliani, C. Zizzo, G. Caridakis, G. Siolas, S. Kollias, M. Graña, F. Barrientos, and S. San Jose. Social and smart: Towards an instance of subconscious social intelligence. *Communications in Computer and Information Science*, 384:302–311, 2013. cited By 0.

[7] Arkaitz Artetxe, Eider Sanchez, Carlos Toro, Cesar Sanín, Edward Szczerbicki, Manuel Graña, and Jorge Posada. Impact of reflexive ontologies in semantic clinical decision support systems. *Cybernetics and Systems*, 44(2-3):187–203, 2013.

[8] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58 – 71, 2007. <ce:title>Software Engineering and the Semantic Web</ce:title>.

[9] B. Avants, P. Dhillon, B.M. Kandel, P.A. Cook, C.T. McMillan, M. Grossman, and J.C. Gee. Eigenanatomy improves detection power for longitudinal cortical change. *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, 15(Pt 3):206–213, 2012. cited By 3.

[10] B.B. Avants, D.J. Libon, K. Rascovsky, A. Boller, C.T. McMillan, L. Massimo, H.B. Coslett, A. Chatterjee, R.G. Gross, and M. Grossman. Sparse canonical correlation analysis relates network-level atrophy to multivariate cognitive measures in a neurodegenerative population. *NeuroImage*, 84:698–711, 2014. cited By 5.

[11] B. Babagholami-Mohamadabadi, A. Jourabloo, A. Zarghami, and S. Kasaei. A bayesian framework for sparse representation-based 3-d human pose estimation. *Signal Processing Letters, IEEE*, 21(3):297–300, March 2014.

[12] G. Bachi, M. Coscia, A. Monreale, and F. Giannotti. Classifying trust/distrust relationships in online social networks. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 552–557, 2012.

[13] Punam Bedi and Ravish Sharma. Trust based recommender system using ant colony for trust computation. *Expert Systems with Applications*, 39(1):1183 – 1190, 2012.

[14] Kamal K. Bharadwaj and Mohammad Yahya H. Al-Shamri. Fuzzy computational models for trust and reputation systems. *Electronic Commerce Research and Applications*, 8(1):37 – 47, 2009.

[15] B. Bhargava, L. Lilien, A. Rosenthal, M. Winslett, M. Sloman, T.S. Dillon, E. Chang, F.K. Hussain, W. Nejdl, D. Olmedilla, and V. Kashyap. The pudding of trust. *Intelligent Systems, IEEE*, 19(5):74 – 88, sept.-oct. 2004.

[16] Jinbo Bi, Kristin P. Bennett, Mark Embrechts, Curt M. Breneman, Minghu Song, Isabelle Guyon, and Andre Elisseeff. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:2003, 2003.

[17] Kirsimarja Blomqvist. The many faces of trust. *Scandinavian Journal of Management*, 13(3):271 – 286, 1997.

[18] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiÃ©rrez. Recommender systems survey. *Knowledge-Based Systems*, 46(0):109 – 132, 2013.

[19] Phillip Bonacich. Power and centrality; A family of measures. *American Sociological Review*, 52, 1987.

[20] Joan Borras, Antonio Moreno, and Aida Valls. Intelligent tourism recommender systems: A survey. *Expert Systems with Applications*, 41(16):7370 – 7389, 2014.

[21] Cristian E. Briguez, Maximiliano C.D. Budan, Cristhian A.D. Deagustini, Ana G. Maguitman, Marcela Capobianco, and Guillermo R. Simari. Argument-based mixed recommenders and their application to movie suggestion. *Expert Systems with Applications*, 41(14):6467 – 6482, 2014.

[22] Christopher Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):167–121, 1998.

[23] T.S. Caetano, J.J. McAuley, Li Cheng, Quoc V. Le, and A.J. Smola. Learning graph matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(6):1048–1058, June 2009.

[24] A.B. Can and B. Bhargava. Sort: A self-organizing trust model for peer-to-peer systems. *Dependable and Secure Computing, IEEE Transactions on*, 10(1):14 –27, jan.-feb. 2013.

[25] D.W. Chadwick, A.J. Young, and N.K. Cicovic. Merging and extending the pgp and pem trust models-the ice-tel trust model. *Network, IEEE*, 11(3):16 –24, may/jun 1997.

[26] Ben-Jye Chang and Szu-Liang Kuo. Markov chain trust model for trust-value analysis and key management in distributed multicast manets. *Vehicular Technology, IEEE Transactions on*, 58(4):1846 –1863, may 2009.

[27] T. Chao, Y. Yan, P. Ma, M. Yang, and Y.W. Hu. Optimization of electromagnetic railgun based on orthogonal design method and harmony search algorithm. *Plasma Science, IEEE Transactions on*, 43(5):1546–1554, May 2015.

[28] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligent Research*, 16:321–357, 2002.

[29] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *Neural Networks, IEEE Transactions on*, 2(2):302–309, 1991.

[30] Xiao Cheng Chen, Run Jia Liu, and Hui you Chang. Research of collaborative filtering recommendation algorithm based on trust propagation model. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 4, pages V4–177–V4–183, 2010.

[31] Jin-Hee Cho, A. Swami, and Ing-Ray Chen. A survey on trust management for mobile ad hoc networks. *Communications Surveys Tutorials, IEEE*, 13(4):562 –583, quarter 2011.

[32] Konstantinos Christidis and Gregoris Mentzas. A topic-based recommender system for electronic marketplace platforms. *Expert Systems with Applications*, 40(11):4370 – 4379, 2013.

[33] F.C.T. Chua, H.W. Lauw, and Ee-Peng Lim. Generative models for item adoptions using social correlation. *Knowledge and Data Engineering, IEEE Transactions on*, 25(9):2036–2048, Sept 2013.

[34] Orlando Cicchello and Stefan C. Kremer. Inducing grammars from sparse data sets: A survey of algorithms and results. *J. Mach. Learn. Res.*, 4:603–632, December 2003.

[35] M. Culp and G. Michailidis. Graph-based semisupervised learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(1):174–179, Jan 2008.

[36] The Anh Dang and Emmanuel Viennet. Collaborative filtering in social networks: A community-based approach. In *Computing, Management and Telecommunications (ComManTel), 2013 International Conference on*, pages 128–133, 2013.

[37] R Diao, F Chao, T Peng, N. Snooke, and Q Shen. Feature selection inspired classifier ensemble reduction. *Cybernetics, IEEE Transactions on*, 44(8):1259–1268, Aug 2014.

[38] P. Domingos and M. Richardson. Mining the network value of customers. In *Seventh International Conference on Knowledge Discovery and Data Mining*. 2001.

[39] Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[40] K.S. Cook (Editor). *Trust in Society*, volume 2. Russell Sage Foundation Series on Trust, February 2003. New York.

[41] O. Fachrunnisa and F.K. Hussain. A methodology for maintaining trust in industrial digital ecosystems. *Industrial Electronics, IEEE Transactions on*, 60(3):1042 –1058, march 2013.

[42] Alberto Fernandez, Maria Jose del Jesus, and Francisco Herrera. On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets. *Information Sciences*, 180(8):1268 – 1291, 2010.

[43] Yu-Hsiang Fu, Chung-Yuan Huang, and Chuen-Tsai Sun. Using global diversity and local topology features to identify influential network spreaders. *Physica A: Statistical Mechanics and its Applications*, 433(0):344 – 355, 2015.

[44] Xinmao Gai, Yong Li, Yasha Chen, and Changxiang Shen. Formal definitions for trust in trusted computing. In *Ubiquitous Intelligence Computing and 7th International Conference on Autonomic Trusted Computing (UIC/ATC), 2010 7th International Conference on*, pages 305 –310, oct. 2010.

[45] S. Galhotra, A. Arora, S. Virinchi, and S. Roy. Asim: A scalable algorithm for influence maximization under the independent cascade model. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 35–36. International World Wide Web Conferences Steering Committee, 2015.

[46] M-L Gao, L-L Li, X-M Sun, and D-S Luo. Face tracking based on differential harmony search. *Computer Vision, IET*, 9(1):98–109, 2015.

[47] Salvador Garcia, Joaquin Derrac, Isaac Triguero, Cristobal J. Carmona, and Francisco Herrera. Evolutionary-based selection of generalized instances for imbalanced classification. *Knowledge-Based Systems*, 25(1):3 – 12, 2012.

[48] Zong Woo Geem, Joong Hoon Kim, and GV Loganathan. A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68, 2001.

[49] Z.W. Geem. Novel derivative of harmony search algorithm for discrete design variables. *Applied mathematics and computation*, 199(1):223–230, 2008.

[50] ZW Geem, JH Kim, and GV Loganathan. Harmony search optimization: application to pipe network design. *International journal of modelling & simulation*, 22(2):125–133, 2002.

[51] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[52] Jennifer Golbeck. Computing with trust: Definition, properties, and algorithms. In *Securecomm and Workshops, 2006*, pages 1 –7, 28 2006-sept. 1 2006.

[53] D.E. Goldberg and J.H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[54] Jacob Goldenberg. Talk of the Network : A Complex Systems Look at the Underlying Process of Word-of-Mouth. *Marketing Letters*, pages 211–223, 2001.

[55] A.I. Gonzalez, M. Graña, J. Ruiz-Cabello, and A. d'Anjou. Experimental results of an evolution-based adaptation strategy for vq image filtering. *Information Sciences*, 133((3-4)):249–266, 2001.

[56] A. Goyal, F. Bonchi, and L.V.S. Lakshmanan. A data-based approach to social influence maximization. *Proceedings of the VLDB Endowment*, 5(1):73–84, 2011.

[57] A. Goyal, W. Lu, and L.V.S. Lakshmanan. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 47–48. ACM, 2011.

[58] A. Goyal, W. Lu, and L.V.S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 211–220. IEEE, 2011.

[59] M. Grana. Subconscious social computational intelligence. In R. S. Lekshmi M. S. Kumar A. Bonato G. S. Sundara Krishnan, R. Anitha and M. Graña (eds), editors, *Computational Intelligence, Cyber Security and Computational Models, Proceedings of ICC3*, number 246 in Advances in Intelligent

Systems and Computing, pages 15–21. Springer Verlag, December 2013 20013.

[60] M. Grana, B. Apolloni, M. M. Fiasche, G. Galliani, C. Zizzo, G. Caridakis, G. Siolas, S. Kollias, F. Barriento, and S. San Jose. Social and smart: towards an instance of subconscious social intelligence. In H. Papadopoulos L. Iliadis and C. Jayne (Eds.), editors, *EANN 2013, Part II*, number 384 in CCIS, pages 302–311. Springer Verlag, 2013.

[61] M. Grana, I. Marques, A. Savio, and B. Apolloni. A domestic application of intelligent social computing: the sands project. In A Herero et al. (eds), editor, *International Joint Conference SOCO'13-CISIS'13-ICEUTE'13*, number 239 in Advances in Intelligent Systems and Computing, pages 221–228. Springer Verlag, 2013.

[62] M. Grana and I. Rebollo. Instances of subconscious social intelligent computing. In *Computational Aspects of Social Networks (CASoN), 2013 Fifth International Conference on*, pages 74–78, Aug 2013.

[63] Manuel Grana, J. David Nunez-Gonzalez, and Bruno Apolloni. A discussion on trust requirements for a social network of eahoukers. In *HAIS 2013, Proceedings*, pages 540–547. Springer Berlin Heidelberg, 2013.

[64] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[65] H.V. H. Ayala, L. Dos Santos Coelho, V.C. Mariani, M. V. Ferreira Da Luz, and J.V. Leite. Harmony search approach based on ricker map for multi-objective transformer design optimization. *Magnetics, IEEE Transactions on*, 51(3):1–4, March 2015.

[66] Daojing He, Chun Chen, S. Chan, Jiajun Bu, and A.V. Vasilakos. A distributed trust evaluation model and its application scenarios for medical sensor networks. *Information Technology in Biomedicine, IEEE Transactions on*, 16(6):1164 –1175, nov. 2012.

[67] Mehdi Heidari, Masoud Asadpour, and Hesham Faili. Smg: Fast scalable greedy algorithm for influence maximization in social networks. *Physica A: Statistical Mechanics and its Applications*, 420(0):124 – 133, 2015.

[68] Julia Heidemann, Mathias Klier, and Florian Probst. Online social networks: A survey of a global phenomenon. *Computer Networks*, 56(18):3866 – 3878, 2012. The {WEB} we live in.

[69] Lance J. Hoffman, Kim Lawson-Jenkins, and Jeremy Blum. Trust beyond security: an expanded trust model. *Commun. ACM*, 49(7):94–101, July 2006.

[70] R.R. Hoffman, M. Johnson, J.M. Bradshaw, and A. Underbrink. Trust in automation. *Intelligent Systems, IEEE*, 28(1):84–88, 2013.

[71] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[72] G. . Huang, Q. . Zhu, and C. . Siew. Extreme learning machine: A new learning scheme of feedforward neural networks. In *IEEE International Conference on Neural Networks - Conference Proceedings*, volume 2, pages 985–990, 2004. Cited By (since 1996):113.

[73] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70:489 – 501, 2006.

[74] Chunxiao Jiang, Yan Chen, and K.J.R. Liu. Graphical evolutionary game for information diffusion over social networks. *Selected Topics in Signal Processing, IEEE Journal of*, 8(4):524–536, Aug 2014.

[75] Q. Jiang, G. Song, C. Gao, Y. Wang, W. Si, and K. Xie. Simulated annealing based influence maximization in social networks. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[76] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[77] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[78] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986, 1984.

[79] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):pp. 671–680, 1983.

[80] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD 2008*, 2008.

[81] Nikolay Korovaiko and Alex Thomo. Predictingtrust from user ratings. *Procedia Computer Science*, 10(0):263 – 271, 2012. <ce:title>ANT 2012 and MobiWIS 2012</ce:title>.

[82] Bartosz Krawczyk, Michal Wozniak, and Gerald Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14, Part C(0):554 – 562, 2014.

[83] Jong-Ryul Lee and Chin-Wan Chung. A query approach for influence maximization on specific users in social networks. *Knowledge and Data Engineering, IEEE Transactions on*, 27(2):340–353, Feb 2015.

[84] K.S. Lee and Z.W. Geem. A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9):781–798, 2004.

[85] K.S. Lee and Z.W. Geem. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194(36-38):3902 – 3933, 2005.

[86] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.

[87] Haifeng Liu, Ee-Peng Lim, Hady W. Lauw, Minh-Tam Le, Aixin Sun, Jaideep Srivastava, and Young Ae Kim. Predicting trusts among users of online communities – an epinions case study. In *EC'08, Proceedings of the 9th ACM conference on Electronic commerce*, pages 310–319. ACM, 2008.

[88] Wei Liu, Jun Wang, and Shih-Fu Chang. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9):2624–2638, Sept 2012.

[89] Victoria Lopez, Alberto Fernandez, Salvador Garcia, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250(0):113 – 141, 2013.

[90] Victoria Lopez, Alberto Fernandez, and Francisco Herrera. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257(0):1 – 13, 2014.

[91] Victoria Lopez, Alberto Fernandez, Jose G. Moreno-Torres, and Francisco Herrera. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585 – 6608, 2012.

[92] Dunia Lopez-Pintado. Diffusion in complex social networks. *Games and Economic Behavior*, 62(2):573 – 590, 2008.

[93] Z. Lu, W. Zhang, W. Wu, J. Kim, and B. Fu. The complexity of influence maximization problem in the deterministic linear threshold model. *Journal of combinatorial optimization*, 24(3):374–378, 2012.

[94] M.S. Lund, B. Solhaug, and K. Stlen. Evolution in relation to risk and trust management. *Computer*, 43(5):49 –55, may 2010.

[95] Ji Ma and M.A. Orgun. Trust management and trust theory revision. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(3):451 –460, may 2006.

[96] M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. *Applied mathematics and computation*, 188(2):1567–1579, 2007.

[97] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11:19–60, March 2010.

[98] D.W. Manchala. E-commerce trust metrics and models. *Internet Computing, IEEE*, 4(2):36 –44, mar/apr 2000.

[99] Antonio Maratea, Alfredo Petrosino, and Mario Manzo. Adjusted f-measure and kernel scaling for imbalanced data learning. *Information Sciences*, 257(0):331 – 341, 2014.

[100] Ion Marques, Manuel Graña, Anna Kamińska-Chuchmała, and Bruno Apolloni. An experiment of subconscious intelligent social computing on household appliances. *Neurocomputing*, in press, 2014.

[101] A. Mayoue, Q. Barthelemy, S. Onis, and A. Larue. Preprocessing for classification of sparse data: Application to trajectory recognition. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 37–40, Aug 2012.

[102] Rezvan Mohamadi-Baghmolaei, Niloofar Mozafari, and Ali Hamzeh. Trust based latency aware influence maximization in social networks. *Engineering Applications of Artificial Intelligence*, 41(0):195 – 206, 2015.

[103] A. Monteserin and A. Amandi. Whom should i persuade during a negotiation? an approach based on social influence maximization. *Decision Support Systems*, 77:1 – 20, 2015.

[104] A. Nazemian, H. Gholami, and F. Taghiyareh. An improved model of trust-aware recommender systems using distrust metric. In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pages 1079–1084, Aug 2012.

[105] Surya Nepal, Sanat Kumar Bista, and Cecile Paris. Behavior-based propagation of trust in social networks with restricted and anonymous participation. *Computational Intelligence*, pages n/a–n/a, 2014.

[106] Mark Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[107] Viet-An Nguyen, Ee-Peng Lim, Jing Jiang, and Aixin Sun. To trust or not to trust? predicting online trusts using trust antecedent framework. In *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pages 896 – 901. IEEE, 2009.

[108] MGH Omran and M Mahdavi. Global-best harmony search. *Applied Mathematics and Computation*, 198(2):643–656, 2008.

[109] G. Palubeckis. Fast simulated annealing for single-row equidistant facility layout. *Applied Mathematics and Computation*, 263:287 – 301, 2015.

[110] N. Pathak, A. Banerjee, and J. Srivastava. A generalized linear threshold model for multiple cascades. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 965–970. IEEE, 2010.

[111] Sancheng Peng, Min Wu, Guojun Wang, and Shui Yu. Containing smartphone worm propagation with an influence maximization algorithm. *Computer Networks*, 74, Part B(0):103 – 113, 2014. Special Issue on Mobile Computing for Content/Service-Oriented Networking Architecture.

[112] BA Prakash, D Chakrabarti, NC Valler, M Faloutsos, and C Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. *Knowledge and information systems*, 33(3):549–575, 2012.

[113] A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[114] Khadije Rahimkhani, Abolfazl Aleahmad, Maseud Rahgozar, and Ali Moeini. A fast algorithm for finding most influential people based on the linear threshold model. *Expert Systems with Applications*, 42(3):1353 – 1361, 2015.

[115] Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. Super mediator. a new centrality measure of node importance for information diffusion over social network. *Information Sciences*, (0):–, 2015.

[116] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Andres Folleco. An empirical study of the classification performance of learners on imbalanced and noisy software quality data. *Information Sciences*, 259(0):571 – 595, 2014.

[117] N. Shadbolt. A matter of trust. *Intelligent Systems, IEEE*, 17(1):2–3, 2002.

[118] Shang Shang, Sanjeev R Kulkarni, Paul W Cuff, and Pan Hui. A randomwalk based model incorporating social information for recommendations. In *Machine Learning for Signal Processing (MLSP), 2012 IEEE International Workshop on*, pages 1–6. IEEE, 2012.

[119] C.K. Shiva and V. Mukherjee. Comparative performance assessment of a novel quasi-oppositional harmony search algorithm and internal model control method for automatic generation control of power systems. *Generation, Transmission Distribution, IET*, 9(11):1137–1150, 2015.

[120] G. Song, X. Zhou, Y. Wang, and K. Xie. Influence maximization on large-scale mobile social network: A divide-and-conquer method. *Parallel and Distributed Systems, IEEE Transactions on*, 26(5):1379–1392, May 2015.

[121] Junqing Sun, Zhaohao Sun, Yuanzhe Li, and Shuliang Zhao. A strategic model of trust management in web services. *Physics Procedia*, 24, Part B(0):1560 – 1566, 2012. <ce:title>International Conference on Applied Physics and Industrial Engineering 2012</ce:title>.

[122] Yan Lindsay Sun, Wei Yu, Zhu Han, and K.J.R. Liu. Information theoretic framework of trust modeling and evaluation for ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):305 – 317, feb. 2006.

[123] Jiliang Tang, Huiji Gao, Xia Hu, and Huan Liu. Exploiting homophily effect for trust prediction. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 53–62, New York, NY, USA, 2013. ACM.

[124] Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. Exploiting local and global social context for recommendation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 2712–2718. AAAI Press, 2013.

[125] M. Tarkeshwar and V. Mukherjee. Quasi-oppositional harmony search algorithm and fuzzy logic controller for load frequency stabilisation of an isolated hybrid power system. *Generation, Transmission Distribution, IET*, 9(5):427–444, 2015.

[126] Alvaro Tejeda-Lorente, Carlos Porcel, Eduardo Peis, Rosa Sanz, and Enrique Herrera-Viedma. A quality based recommender system to disseminate information in a university digital library. *Information Sciences*, 261(0):52 – 69, 2014.

[127] G. Theodorakopoulos and J.S. Baras. On trust models and trust evaluation metrics for ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):318 – 328, feb. 2006.

[128] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, September 2001.

[129] Carlos Toro, Eider Sanchez, Eduardo Carrasco, Leonardo Mancilla-Amaya, Cesar Sanín, Edward Szczerbicki, Manuel Graña, Patricia Bonachela, Carlos Parra, Gloria Bueno, and Frank Guijarro. Using set of experience knowledge structure to extend a rule set of clinical decision support system for alzheimer's disease diagnosis. *Cybernetics and Systems*, 43(2):81–95, 2012.

[130] Sandra A. Vannoy and Prashant Palvia. The social influence model of technology adoption. *Commun. ACM*, 53(6):149–153, June 2010.

[131] V. Vapnik. *Statistical learning theory*. Wiley-Interscience, 1998.

[132] P. Victor, C. Cornelis, M.D. Cock, and A.M. Teredesai. Trust- and distrust-based recommendations for controversial reviews. *Intelligent Systems, IEEE*, 26(1):48–55, 2011.

[133] Jinghua Wang, Jane You, Qin Li, and Yong Xu. Extract minimum positive and maximum negative features for imbalanced binary classification. *Pattern Recognition*, 45(3):1136 – 1145, 2012.

[134] Xiaofeng Wang, Ling Liu, and Jinshu Su. Rlm: A general model for trust representation and aggregation. *Services Computing, IEEE Transactions on*, 5(1):131 –143, jan.-march 2012.

[135] Daoxi Xiu and Zhaoyu Liu. A formal definition for trust in distributed systems. In Jianying Zhou, Javier Lopez, RobertH. Deng, and Feng Bao, editors, *Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 482–489. Springer Berlin Heidelberg, 2005.

[136] Xin Xu, Zhenhua Huang, D. Graves, and W. Pedrycz. A clustering-based graph laplacian framework for value function approximation in reinforcement learning. *Cybernetics, IEEE Transactions on*, 44(12):2613–2625, Dec 2014.

[137] P. Yadav, R. Kumar, S.K. Panda, and C.S. Chang. Optimal thrust allocation for semisubmersible oil rig platforms using improved harmony search algorithm. *Oceanic Engineering, IEEE Journal of*, 39(3):526–539, July 2014.

[138] Xiwang Yang, Yang Guo, Yong Liu, and Harald Steck. A survey of collaborative filtering based social recommender systems. *Computer Communications*, 41(0):1 – 10, 2014.

[139] Haoxi Zhang, Cesar Sanín, and Edward Szczerbicki. Implementing fuzzy logic to generate user profile in decisional dna television: The concept and initial case study. *Cybernetics and Systems*, 44(2-3):275–283, 2013.

[140] Ping Zhang, A. Durresi, and L. Barolli. Survey of trust management on various networks. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2011 International Conference on*, pages 219 –226, 30 2011-july 2 2011.

[141] Maciej Zieba, Jakub M. Tomczak, Marek Lubicz, and Jerzy Swiatek. Boosted {SVM} for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients. *Applied Soft Computing*, 14, Part A(0):99 – 108, 2014.