AUTOMATING SOFTWARE CUSTOMIZATION VIA CROWDSOURCING USING ASSOCIATION RULE MINING AND MARKOV DECISION PROCESSES

SAEIDEH HAMIDI

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ARTS

GRADUATE PROGRAM IN MASTER OF INFORMATION
TECHNOLOGY
YORK UNIVERSITY
TORONTO, ONTARIO

JULY 2014

Abstract

As systems grow in size and complexity so do their configuration possibilities. Users of modern systems are easy to be confused and overwhelmed by the amount of choices they need to make in order to fit their systems to their exact needs. In this thesis, we propose a technique to select what information to elicit from the user so that the system can recommend the maximum number of personalized configuration items. Our method is based on constructing configuration elicitation dialogs through utilizing crowd wisdom.

A set of configuration preferences in form of association rules is first mined from a crowd configuration data set. Possible configuration elicitation dialogs are then modeled through a Markov Decision Processes (MDPs). Within the model, association rules are used to automatically infer configuration decisions based on knowledge already elicited earlier in the dialog. This way, an MDP solver can search for elicitation strategies which maximize the expected amount of automated decisions, reducing thereby elicitation effort and increasing user confidence of the result. We conclude by reporting results of a case study in which this method is applied to the privacy configuration of Facebook.

Acknowledgements

This research concludes my Master studies in School of Information Technology at York University and would have not been possible without the help and support of the people whom I would like to thank.

First and foremost, I would like to express my profound gratitude towards Prof. Liaskos for his enthusiasm in this topic and his vigilant guidance throughout my research. I am also truly grateful for his kind supervision and constant support during my study at York University.

I am also sincerely grateful to Dr. Andritsos from University of Toronto for his invaluable assistance during this research. I take the opportunity to extend my sincere thanks to Dr. Yang, Dr. Litoiu, Dr. Ali Asgary, and my committee members, who so kindly lent me their precious time.

Last but not least, I would like to express my sincere thanks to my friends and staff members at School of Information Technology for their support.

Table of Contents

List of Tables

List of Figures

Chapter 1 : Introduction

1.1 Automating Software Customization

Modern software systems exhibit functionalities in amounts and complexities that have never been seen before. As systems mature and new challenges and opportunities emerge, more features are developed and added to an already large and complex set of existing ones. Users interested to fully exploit the power of their software are invited to choose the functions they need and *configure* them in a way that perfectly aligns with their goals, preferences and capabilities [1]. But as the amount and complexity of functions grow, so does the space of configuration possibilities. Thus, when attempting to fit the functionality of their systems to their unique needs, users have to deal with an increasingly mystifying and overwhelming task.

In handling and customizing the configuration of the system, users not only need to be familiar with technical options, but they also need to be aware of the implications of each alternative. Take an email client as an example and a setting, within such system, of intervals for checking new emails with alternatives of short to long intervals. The concept of this setting (i.e. the frequency of updates of email) may be well understood by the users, but the implications of each option may not be realized; users may or may not be aware that shorter intervals impose more traffic on the network or distractions for themselves, while the longer intervals reduces their availability in exchange for lower network traffic

[2]. This lack of technical knowledge among non-expert users calls for tools and facilities to help users with customization of their systems.

The dominant practice for configuring software nowadays is through constructing and using configuration screens either during installation of the application, or after the application has been installed. Virtually every common desktop, mobile or web application has one or more "Options" or "Preferences" screens presenting to the user lists of configuration variables and allowing her to configure each by selecting the option of their choice.

But in this way, users are left alone to go through all variables and somehow decide what a good option is for each variable. Beside the huge number of variables that users have to deal with, their difficulty to choose the best option which fits to their need is another challenge. Not all users of software applications are professional users or feel comfortable with handling the settings. Studies suggest that the extent to which users are able to customize their settings to reflect their preferences depends on their computer skills and level of technical understanding the relevant settings [3, 4]. Novice users in particular may have no clue as to what choices are sensible with respect to their own needs and preferences, as they have made limited or no use of the system before. To sum up, the specific reasons why users may have difficulty coping with configuration screens seem to be that: (a) the user has to painstakingly go through each and every variable (b) user will not be confident that she has chosen the right combination of values with respect to their own needs and preferences.

As a remedy, the current practice is to offer a universal one-size-fits-all default configuration to all users, apparently with hopes that for a majority of users it will be good enough to delve into the configuration screens. Alas, research also shows that most people rarely change the default [5]. This means that either the default options are suitable for most people which obviates the need to further customize the software or, quite more credibly, that customization is difficult for the majority. Again, the assumption of users difficulty in customization is particularly true for the least skilled users, who are known to be the least likely to be able to customize the settings on their own and change the default [3].

But what if users could just consult each other? If a novice user had access to the configurations that a community (*"crowd"*) of expert users have set for themselves, this would be valuable advice for building her own configuration. Configuration could actually be in part automated: all she would have to do is to provide a minimum of configuration options and have the rest be automatically defined, based on how a like-minded group (based on her limited input) within the expert crowd has configured their own system.

Consider for example in the email client, among other things, the user has to configure the size of the font (small, medium, large etc.) and the size of the function icons. Assume further that we know that those in the expert crowd who have large fonts also have large icons. If that is the case, it is fair to expect that once the user declares that she likes large fonts, the system also knows without further asking that she probably prefers large icons – at least from whichever arbitrary default. But how can we identify a minimum set of user-provided configuration options in order to have his/her entire system configured with minimum effort?

In this study, we propose a way by which users can reduce the effort to configure their system by crowdsourcing the configuration preferences of a crowd. The exact problem set-up is explained in the following section.

1.2 Problem Definition

Configuring modern software systems is a process of assigning values to a number of parameters that the designers of the software system avail to users within various configuration facilities, such as “Options", and “Preference" screens or configuration files. In more abstract terms, a configuration problem consists of a set $\mathcal{V}$ of configuration *variables* V_i each drawing values o_j^i (*options*) from the set O_{V_i}.

Let us return to the above configuration problem from our hypothetical email client program. Among its many configuration variables (e.g. whether to use HTML, what connection security to apply, whether to include a signature in each email, etc.), assume that the email program allows the user to also adjust *font size*, *icon size* (for buttons such as “New E-Mail” and “Reply”) and whether to also *display text* under the icons. Let us denote these variables as V_{fnt}, V_{ico} and V_{disp}, respectively. The user can adjust the variables to take values from the sets: {*large*, *medium, small*}, {*large icon, small icon*} and {*yes*, *no*}, respectively. Thus, using the above formalization the domain of each variable is as follows:

$O_{V_{fnt}}= \{o_{lg}^{fnt} , o_{med}^{fnt} , o_{sm}^{fnt}\}$

$O_{V_{ico}}= \{o_{lgi}^{ico} , o_{smi}^{ico} \}$

$O_{V_{disp}}= \{o_{yes}^{disp} , o_{no}^{disp} \}$

A *configuration decision* is the (human/non-automated) act of thinking and deciding what option is more suitable or preferred for a particular variable. In this case, in order to completely solve the configuration problem, the email client user would need to make three (3) configuration decisions, thinking and deciding about each variable separately.

Nevertheless, we may have some crowd configuration preferences in the form of rules (or patterns) mined from e.g. a group of other users. Such rules would tell us that if the user decides to adjust some of her configuration variables in a specific way, then she also wants to adjust some other configuration variables in another, also specific, way. The extracted rules from the crowd are in fact association rules, which are of the following general form:

$(V_{l_1} = o.^{l_1}) \wedge (V_{l_2} = o.^{l_2}) \wedge \ldots \rightarrow (V_{r_1} = o.^{r_1}) \wedge (V_{r_2} = o.^{r_2}) \wedge \ldots$

Where V_{l_i} and V_{r_i} are respectively variables on the left-hand and right-hand side of the rule, and $o._{}^{i}$ and $o.^{r_i}$ are arbitrary values thereof, drawn, of course, from the corresponding domains.

Back to the email client example, we may be aware that if a user prefers to have large font size, she also likes to have large icon size. Furthermore we know that if she likes large icon size then she also prefers text to not be displayed under the icon, apparently to save space. Here is how we can write those rules more formally:

$(V_{fnt} = o_{lg}^{fnt}) \rightarrow (V_{ico} = o_{lgi}^{ico})$

$(V_{ico} = o_{lgi}^{ico}) \rightarrow (V_{disp} = o_{no}^{disp})$

Notice now how this knowledge, whenever available, can help us reduce configuration effort. Without the association rule, in order to have the complete set configured, the user would need to make three separate decisions – about font size, about icon size and about text display. However, if we assume presence of the association rules, we know that some choices in some variables help us infer choices in other variables, saving us from making the corresponding configuration decisions. In this example, it is obvious that if we know that the user prefers to have large font, then we also know what font she wants for the icons and, better yet, we also know if she wants text under the icons.

Given this, it is easy to see that there are more efficient and less efficient *orderings* by which the configuration decisions can be made. Given the above two configuration rules, if the user decides about icon size first, then text display, and finally font size, she will always have to make three decisions. If, on the contrary, she starts by deciding what font size she wants, then, depending on her decision on that, there is a chance that the other two decisions need not to be made, but are taken care of by the rules, instead.

More generally, given a set of configuration variables to be decided upon, and a set of crowd preference rules among them, what is the optimal sequence in which potential decisions can be ordered, so that the smallest number of decisions is eventually made on average? Knowing how to compute such a sequence of decisions could be useful for constructing intelligent configuration elicitation interfaces in which users are directed towards making more influential decisions first, allowing thereby a larger subset of variables to be configured automatically without the need for a human decision.

In the next subsection we outline how we address this problem in this thesis.

1.3 Proposed Framework

We propose a way by which the effort of users to configure their system is minimized and the fitness of the result is improved by utilizing configuration preferences of a crowd. The proposed framework is based on mining of association rules between configuration preferences and solving the sequence of configuration items based on Markov Decision Process.

Configuration preferences in the proposed technique come in the form of association rules, which are mined from the chosen configurations of a crowd of users. Given the configuration options of the crowd, the goal is to model a recommender that minimizes the effort of a new user in configuring the options. The proposed recommender will ask user to explicitly configure a minimized number of parameters, and will recommend (or predict) the rest of the options accordingly. The challenge therefore, is determining the sequence of questions to be asked from the user that will yield the most gain in predicting the rest of configurations. To solve this, a method based on combining association rule mining with Markov Decision Processes (MDPs) is proposed as follows.

The first step is mining combinations of configuration options that frequently coincide. At this stage, after acquiring ("crowdsourcing") a dataset of users' customized configurations, the association rules are mined following standard a-priori algorithm [6]. To focus on significant associations, a threshold for minimum support and confidence is decided. The result is a set of association rules that are highly suggestive by our data. These rules tell us that those of the crowd who make particular configuration choices with one or more of the configuration parameters, also make certain choices with other parameters.

The next step is the construction of the MDP model. As we will see, MDP models consist of states and actions that lead - with a certain probability - from one state to the other. In our problem, states correspond to the state of our knowledge of the preferred configuration for a particular user. Actions, on the other hand, are configuration questions posed to the users in order to learn their preferred option for a given variable, as in “how would you like to configure this parameter?”. In such a question the user has different options to respond, and she will pick either of those with a certain probability. As such the action of posing the question comes with a probability distribution of possible outcomes. This distribution is decided based on the frequency of configuration options in the crowd sourced dataset.

User response to the action naturally triggers transition from a less informed to a more informed state, in which only one more variable is known; but the association rules help us multiply the amount of knowledge we gain with each answer, effectively skipping unnecessary questions. Using an MDP-solver we can solve the MDP and obtain a policy, which describes sequences of actions that optimize expected utility - there are many such because actions have different effects. In our application, by solving the MDP through rewarding shorter paths towards more configuration knowledge, we are able to minimize the length of the sequence of questions that are required for a user to have a complete configuration. Thus the resulting policy will decide the best action to take (i.e. ask user to set a configuration parameter) in each state of the knowledge of the configurations until all the configurations are figured out.

This technique is able to reduce the amount of effort users need to dedicate in order to configure their system in practical cases. In order to evaluate this, we apply the technique using data collected from the popular social networking system, Facebook, where it is shown that configuration effort is substantially reduced on average. Furthermore, in performance experimentation, the solver ability to handle a useful size of configuration spaces within practical computation time is investigated.

1.4 Contributions

The following summarize the novel contributions of this thesis:

- We introduce a systematic method for developing adaptive configuration elicitation dialogs that save overall configuration effort and offer expert guidance to novice users.
- We utilize the wisdom of crowd of experts while keeping the proposal independent of the configuration content or the application domain and without requiring prior knowledge about the user.
- We propose a way to combine association rules and MDPs in order to globally optimize the quality of elicitation dialogues.
- We conduct a case study using real Facebook privacy configuration data to assess applicability of the approach.
- We study the performance of MDP reasoning in the configuration problem and assess the number of variables that can be supported by our approach within practical time.

1.5 Structure of the thesis

The thesis is structured as follows. In Chapter 2, background on the techniques used in our proposed framework and a review on the literature within the scope of our effort is provided. Chapter 3 contains detailed description about how the MDP and Association Rule mining can be applied in minimizing the effort of users in customizing the configurations. In Chapter 4, we present the experimental study that was performed in order to observe our proposal in practice and assess its feasibility. In Chapter 5 we evaluate the performance of our proposed technique in the context of large domains. In final chapter, Chapter 6, the thesis is concluded with a discussion on the techniques used; we also elaborate on the limitations of this study and highlight the areas in which further research is suggested.

Chapter 2 : Background and Related Work

2.1 Background

In this section we provide an overview on the association rule mining technique and Markov Decision Process (MPD) framework which are utilized in the proposed framework.

2.1.1 Association Rule Data Mining

Association Rule Mining is one of the popular data mining techniques to discover patterns between items in a dataset. In association mining, unlike other techniques of data mining (e.g. classification and clustering [7]), we are not interested in detection of the category of items, but in the relation between them. This technique was initially used for market basket analysis to find how items purchased are related [8]; the detected buying patterns would be used in marketing activities.

The input in association mining is a dataset consisting of categorical attributes. The objective is to detect strong relations between the items and to find rules that will predict the occurrence of a set of item, referred to as *itemset*, based on the occurrences of another one [7]. For example, a grocery store with electronic check-out is able to keep track of the combinations of items that customers bring in their baskets to the check-out. Table 1 depicts an example of such transactions in a hypothetical store. An association rule would detect that the majority of the customers who shopped one set of items (e.g. sausages and buns), also had another specific set of items in their basket (e.g. beer and mustard). In this example, this information may support the marketing and product placement/shelving activities of the grocery.

Association rules are expressed in the form ($X \longrightarrow Y$) (c,s), where X and Y are itemsets that often appear together (e.g. sausages, buns, mustard and pickles respectively) and c and s denote the *confidence* and *support* of the rule respectively. The rule states when X occur, Y occur with certain probability c. The level of support s shows the fraction of transactions that have both X and Y. Since the amount of association rules between all items can be huge, confidence and support are used as measures of significance and interestingness of a rule.

Table 1 - Example of transactions in a store

transaction	sausage	buns	beer	mustard
1	1	0	0	1
2	1	0	1	0
3	1	1	0	1
4	0	1	0	0
5	1	1	1	1

At the heart of the association rule mining is the detection of frequent itemsets with minimum support constraint. Subsequently, the minimum confidence constraint is used to form rules X⟶Y between two frequent itemsets X and Y. The confidence of a rule is simply the calculation of the conditional probability P(Y|X) which shows what percentage of the instances that contain itemset X also contain itemset Y. For example, the rule {sausages, buns} ⟶{mustard} has a confidence of $2/2 = 1.0$ in the database of Table 1. This means that for 100% of the transactions containing sausages and buns the rule is correct; in other words 100% of the times a customer buys sausages and buns, mustard is bought as well. This rule has a support of $2/5 = 0.4$ which means that in 40% of the transactions, the three items of sausages, buns, and mustard are purchased together.

In large datasets, the detection of all frequent itemsets with various sizes is a difficult task and efficient techniques have been proposed for mining association rules. This research entails application of one of the most established techniques, Apriori [9]. Since the details of the techniques are not the relevant to this research, the reader is referred to the literature [6, 8, 10, and 11] in this regard if interested.

2.1.2 Markov Decision Processes

Markov decision processes (MDPs) is a mathematical framework for modeling and solving sequential decision making and optimization problems under uncertainty [12]. MDPs are used for optimization problems in a wide area of disciplines such as production and inventory management [13-15]. In the proposed framework, MDPs are utilized in order to calculate the optimal sequence of configuration questions, given uncertain user responses.

MDPs are used for analysing a discrete time stochastic process: a process in which the system's state changes over stages (i.e. separate points in time) in response to actions and whose outcome is not certain. In MDPs the decision making problem is described by a finite or infinite set of states $s \in S$. Each state is a description of the system at a particular stage and holds all the necessary information to predict the next state. The information can be described through a set of variables, each combination of values of whom uniquely describes each state.

At any state, the decision-making agent chooses an action from a set of actions $a \in \mathrm{A}$. Performance of an action has various possible outcomes, each with a different probability. Such non-deterministic effects are described by a probability distribution: for each action, p_{ij}^{α} is defined for every pair of states s_i and s_j as the probability of reaching state s_i from s_j by taking an action α. Thus each action comes with a *transition matrix* containing p_{ij}^{α} in its cells. In MDPs, the probability distribution of transitions possesses the Markov property: i.e. the probability distribution of future states depends only upon the present state and the taken action, and is independent of the history that preceded it.

In MDPs each state has a value and each action comes at a cost. Functions for *reward* R: $S \times S \longrightarrow R$ and *cost* C: $A \longrightarrow R$ are used to state the immediate reward of reaching the new state and the costs associated with taking the action respectively. A reward matrix can be used to represent R, and a table can be used to assign a cost to each action. The overall

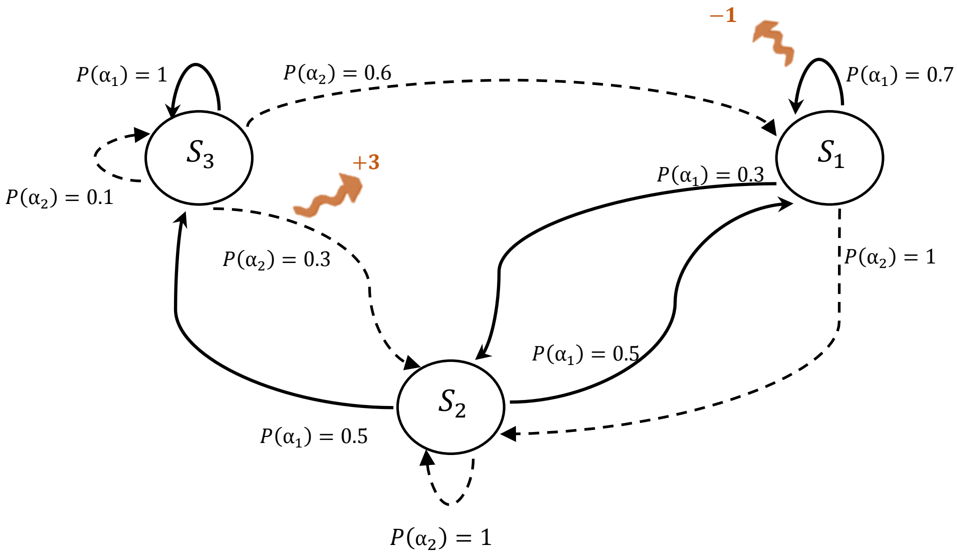


Figure 1 - Example of a simple MDP model

utility of the transition is calculated by subtracting the reward obtained by attaining the transition minus the cost incurred by performing the action.

Figure 1 illustrates a simple example of MDP model with three states S_1, S_2 and S_3. The two actions $a_1$and a_2 are shown in solid and dashed lines. In this model, any action can be taken at each state, and the system will transit to one or a number of destination states based on the probability. Taking an action might not change state of the system, such as taking action $a_1$in states S_3. Furthermore as shown in the model, there is a reward and cost associated with transition from S_3 to S_2 via action α_2 and with taking the action a_1 in state S_1 respectively.

The core problem of MDPs is to find the course of actions, called *optimal policy* π that maximizes the cumulative expected utility of each state. This optimal policy is, roughly, the result of progressively tallying up the product of probability and utility while traversing each sequence of actions and observing potential transitions actuated thereby. Optimal means that there is no other policy that can give the agent a larger expected cumulative utility. For an MDP with stationary dynamics (i.e. independent of time), with either infinite or indefinite number of stages, there always exists an optimal stationary policy [16, 17]. This implies that although the system is non-deterministic, there is always a sequence of actions that can be recommended. MDP solvers are tools that allow calculation of optimal policies given a complete MDP formulation. MDP solvers adopt various algorithms, primarily Value Iteration (VI), and Policy Iteration (PI); discussion on those algorithms are beyond the scope of this work and readers are referred to the references for details [12].

2.2 Related Work

The system that we develop in this thesis for assisting configuration can be seen a Recommendation System. Recommendation systems (RS) are a class of systems that help users in decision making among an overload of choices by predicting the best items that matches user needs. In this section first an overview on prominent techniques of implementation of RS is provided. Subsequently, we proceed to the review the related works of recommendation systems in the field of requirements engineering.

2.2.1 Recommendation Techniques

Recommendation system is a solution to help users in decision making in overload of information: they elicit the preferences of individuals and match them with best items accordingly [18, 19]. There are different types of recommender systems that vary in terms of the information used, and the algorithm that elicits the recommendations. In the simplest form, RS can be non-personalized and can suggest users the most popular/common items such as top seller items in E-commerce. In the realm of software configurations, the default options are in fact such universal recommendations. In higher level, recommendation can be a function of personalization in which users are offered items based on their preferences and constraints.

In the simplest form of personalization, *Demographic* recommendation systems, socio-demographic attributes such as age, gender, profession, and education, are used [18]. In such a system for example, movies of different genres can be suggested to users based on their age and gender.

Knowledge-Based (KB) systems are based on the heuristic rules and specific domain knowledge of the experts. KB systems make recommendations by reasoning about what products or features meet the user's requirements and be useful to her. These systems are mostly utilized in domains where items are complex and specialized, or in systems that do not have many users to collect preferences [20]. An example of such recommender in requirements engineering is the work of Romero-Mariona et al [21], in which they developed a knowledge-based system to recommend the best fit model for security requirements based on 10 fix criteria; the recommendation is based on the user rating of these criteria in his/her system. KB systems therefore, are costly due to necessity of the knowledge acquisition from domain experts and translating this knowledge into a model on which the recommender is based [18].

In *Content-Based* (CB) systems, the general principle is to identify the common features of items that are favourable to each user, and then recommend to him/her the items that share the same features. Since CB systems are based on the semantic features of items, information describing the content of items is assumed available. These features might be provided explicitly as in commercial items, or can be extracted and learned implicitly as in textual contexts. An example of such work done is a recommender for reuse of software requirements that matches a new software project description against the requirements artifacts of already completed software projects from the repository for reuse [22].

Collaborative Filtering (CF) systems [23], probably the oldest and most popular kind of RS [18, 19], are based on the rationale that users who shared similar tastes in the past, will have similar choices in the future. Hence, CF takes advantage of wisdom of crowd and recommends to the active user the items that other users with similar taste like. The similarity in taste of two users is mostly calculated based on history of users (e.g. purchase history in E-commerce). The case of software requirement recommender for reuse has been proposed by collaborative Filtering implementation as well. Lim and Finkelstein [3] introduced a system in which stakeholders will specify ratings on their initial (known) requirements, using the ratings and matching it against the repository of previous projects, the system can recommend additional potential requirements.

Both CB and CF systems have their advantages and challenges. A major issue in content-based systems is that structured information about items and/or the profile of users should be available. Furthermore if the profile of each user needs to be constructed (in order to be associated with features of items), it then requires repetitive updates since the user's attitude and preferences might change over time [24]. CF overcomes some of the limitations of CB systems and is the most prominent approach adopted by large, established e-commerce applications [18, 19]. Another advantage of CF recommendations is that they are based on the perceived usefulness of items as evaluated by the crowd, instead of content that may not be a true indicator of quality. However, CF suffers from the cold start problem at the initial phase, where a user or an item are new in the system and there is not enough information to compute the similarity and hence make the recommendations. In addition, the recommendations might be biased towards popular items, and unlike CB, the system is challenged to attend to users with unique taste [19].

Collaborative Filtering recommendation are separable into two classes: (a) *Memory based* algorithms that require preferences of all users be stored in memory at run-time and (b) *Model-based* algorithms that periodically create a predictive model offline. As a representative of memory-based CF technique, neighbourhood-based systems [19] focus on similarity between items or alternatively between users to recommend the items. Neighbours are users whose history of choices are most correlated. Pearson correlation and vector cosine similarity are commonly used similarity calculations and the weighted average of neighbours is used to make predictions. The choice of neighbours can be based on top N similar ones, or the ones that show a similarity over a specified threshold. Memory-based recommenders have long been a popular approach in huge commercial systems (e.g. Amazon) [18]. This prevalence is due to their simplicity, efficiency, and transparency of calculations, which makes them more interpretable [18, 19, 25, and 26].

In Model-based approaches, a predictive model is constructed at frequent intervals offline and later at run time, the model is readily used to make recommendations. Therefore, although the offline training phase is costly, the recommendation phase is fast [18, 19]. In order to build the predictive model in Model-based RS, various mining techniques such as Bayesian networks (e.g. in [27 - 30]), a linear regression (e.g. in [31]), and clustering (e.g. in [32, 33]) can be practiced. Association rule mining has also been used in constructing collaborative models. Fu et al. developed a system to recommend web pages by mining association rules over users' navigation histories [34]. In another study, Leung et al. proposed a CF framework using fuzzy association rules that takes advantage of product similarities in taxonomies to address data sparseness [35]. More recently, a significant amount of research has been done to model the recommendation process using more complex probabilistic models [36]. Markov Decision Processes (MDP) is one such model that can be utilized. A brief review of the relevant literature in application of MDP in recommendation systems follows.

2.2.2 MDP Based recommender

Optimal policies solved by MDPs have a direct application to recommendation systems. In such systems, the recommendation process is viewed as a sequential optimization problem of recommending items, rather than a prediction problem. An example of such application is the work of Shani et al. [37] in which Markov decision processes (MDPs) model is used to maximize the profit of the sale of an Israeli online bookstore. In their model the state space of the MDP corresponds to possible sequences of purchased items and the actions correspond to a recommendation of an item to a user; the rewards in the MDP correspond to the utility of selling an item, for example, the net profit. They show that the deployed MDP-recommender system produces a much higher profit than the system without using the recommender.

A critical challenge in the application of MDPs in practical recommenders is the rapid growth of the size of the MDP with the complexity of the problem. As the computational and representational complexity of MDPs is high, appropriate approaches and approximations [37-40] must be developed. For instance, in the proposed RS by Shani et al. [37], to control the size of MDP a few measures are taken: Firstly in definition of the states, only sequences of relatively small number of items are tracked. Secondly, only sequences that were observed in the training set are defined in the state space. This approximation is supported by the fact that for unobserved combinations, the probability of making a transition into such a state is very low.

There is an important advantage to the application of MDPs in recommendation systems: they promote the interactive process between the user and the system. In standard CF recommenders, preference elicitation from users relies on a rather limited model of interaction and users tend to collaborate only indirectly [41]. In newer generations of RS, there is more encouragement to allow conversation between user and system in order to effectively elicit explicit information from users. Such systems, referred to as *Conversational* systems [42] assist users in decision making by incorporating a more human-like interaction between the user and the system where both parties may query or provide information to the other partner [18]. An MDP-based recommender (e.g. the one proposed by Shani et al. [37]) is in fact implementation of a conversational recommendation system. A benefit of such conversational systems is that preferences can be elicited over the length of the working with the system, rather than upfront [42]. Since the preferences of users shapes and evolves gradually in the process of working with the system [43], taking a conversational approach is encouraged in recommendation system [18].

2.2.3 Recommendation systems in Requirements Engineering

The literature in the area of RSs is mostly related to E-commerce, and recommendations product- and service- configuration (e.g. 16, 45). Thus there is considerable research into recommending simple products such as books in amazon.com, as well as complex and configurable products and services such as computers [45]. The customization of complex products and the configuration technologies for mass production is in fact similar in concept to the research at hand and has received a great attention in literature [46-50].

Configuration recommenders assist users in selecting attributes and features such as customer requirements and product attributes of a complex product. Example domains of application of such configurators so far are computers [45], cars, financial services, railway stations, and complex telecommunication switches [46]. The motivation for product and service configurators is the same as ours: customers (i.e. users) are overwhelmed with the highly variant products, and get confused by the complexity of the offered options. In many cases, users are reported to have problem understanding the set of offered options in detail [46]. The other challenge that is recognized in literature is that users do not know their preferences beforehand, but rather gain better understanding of the scope of options and shape their preferences within the process of configuration [43]. In such cases, the need to support users with recommendations that are, for example, derived from preferences of similar users is realized [24].

In requirements engineering, the idea of application of RS is still young and rather unexplored. Maalej and Thurimella [51] have outlined a research agenda for recommender systems within the requirements engineering domain and proposed potential areas in which recommender technologies can be utilized to help stakeholders create, analyze, specify, and manage requirements. The body of the available studies for application of recommendations in RE are in support of activities of system analysts, and not the end users. Activities such as requirements elicitation from different stakeholders [e.g. 52, 53], enforcement of quality assurance scenarios [e.g. 54, 21], and negotiation and prioritization of requirements of different stakeholders [e.g. 55] are the mainstream for which recommendation systems have been investigated [56, 57].

In line with the idea of this work, researchers have recently emphasized the idea of integration of recommendation systems to tailor the software configuration process [36]. Autonomic features incorporated into commercial RDBMS are prominent examples of such systems which recommend database configurations (i.e., indexes, materialized views, partitions) for a given workload [58]. However the literature on database configurators are specific to this domain only and mostly centred around RDBMS's own features (i.e. query optimizers) [58]; to our knowledge, there is no generic framework or guidelines as how to construct recommendation systems for customizing system configurations that is applicable to various domains.

2.3 **Software Configuration**

The problem of configuring existing systems has been studied from a variety of angles. Wendy E. Mackay [5] offers one of the earliest and most influential studies on the problem from a Human Computer Interaction view point. In her study she observed a number of users in order to understand when and how they configure the software applications they use. Among other things, it was found that users often do not go into the trouble of customizing their software, fearing that they will waste a lot of time without being able to actually find the configuration variable they need. Moreover, a substantial number of participants simply did not know how to perform the customization. Similar results are reported elsewhere [59], where it was also found that users are more likely to engage in customization activities when doing so is easy. While the studies are quite old, there is no evidence that customization of user applications has become better; the literature on Facebook (Section 4.1.1) reveals that customization remains a difficult problem. Furthermore, one of the interesting findings of the Mackay study is the social aspect of configuration: according to the study, one of the factors that actually trigger customization effort is observing other users achieving it. In the Internet era, 23 years later, this seems to support a vision for crowdsourcing configuration.

Viewing software customization as a requirements problem has also been attempted. Goal modeling has been proposed [60, 61] in order to connect high-level user goals with low level software configurations [2]. Liaskos et al. have been exploring ways to access the variation points of a software system that are alternative to lists of configuration variables [62]. In that work users express their desires through a high-level goal-based preference language [63], and a sequence of automated reasoning steps translate the result into an appropriately behaving system. Nevertheless, for these goal-oriented approaches to be applicable, both the goal models and their mapping to software variability must be pre-established by experts. A recommender system such as the one proposed in this thesis, has the benefit that it does not require any such requirements models.

2.4 Summary

In this chapter the background knowledge necessary to understand this research was provided. The approach introduced in this research applies two established techniques: Association rules are used to mine the pattern of preferences in configuration options, and Markov Decision Processes is utilized to decide the sequence of explicit questions that are inquired from the user to predict the rest of the items.

We also explored the different recommendation techniques that apply to the case of recommending customization of software configuration. We reviewed the proposed approaches to automatic software configuration as well: exploration of the literature revealed that application of recommendation systems in the realm of requirements engineering is still in its infancy and approaches to construct a software configuration recommender independent of the domain, as we propose here, are yet to be introduced.

Chapter 3 : Solution

In this section, we describe in detail our method for calculating the minimum sequence of configuration questions to be posed to the user that is necessary to have the entire system configured, based on a crowd sourced predictions. As we saw, the proposed system can be seen as a model-based Collaborative Filtering (CF) Recommendation System (RS). The model of the proposed recommender is based on Markov Decision Processes (MDPs) informed by Association Rules that are, in turn, mined from crowd data. Roughly, the association rules allow us to skip configuration questions whose answer can be predicted; the MDP is necessary in order to calculate the optimal sequence of such questions, given stochastic user responses.

3.1 Association Rules as Configuration Preferences

The logic behind using association rules to capture common consumer preferences applies directly to the problem of configuration. As mentioned before, in the context of configuration, an association rule captures the fact that if a user selects a particular combination of options for a group of variables, she is likely to also select a certain configuration of options for a different group of variables. This knowledge comes from a data set of configurations of individual users, which we will call the *crowd data set*. In many systems, such as on-line social networking or web-based email systems, such data sets are readily available to e.g. administrators and owners.

In this case, to ensure the validity of the rules, it is assumed that the crowd data set comes from a group of expert users. Then, when a new and novice user makes use of the association rule $X \longrightarrow Y$, she is actually using the left-hand side of the rule to identify with a subgroup of experts whose preference with respect to 'X' match. Then she can use the right-hand side 'Y' as expert advice for configuring the corresponding variables. Effectively, the result is a recommender system based on association rules.

To allow calculation of such sequences, MDPs modeling is used. We describe how in the next section.

3.2 **Building the MDP**

Let us now focus on the details of modeling the explained problem as an MDP. Summarizing the discussion in the previous chapter, a fully defined MDP includes: (a) a set of variables describing the state of the domain, (b) a set of actions $\alpha \in \mathcal{A}$ which actuate transitions from one state to the other, (c) a set of transition matrices, one per action α, showing the probability of transition from one state to each of the others, after the action has been performed, (d) a reward matrix representing the reward function R and (e) a cost table representing the cost function C.

This section is dedicated to explanation of each of these components in our solution in detail. Throughout this section the case configuration options of an email client is used to better illustrate the concepts.

3.2.1 States

In our solution, each state of the MDP model represents the state of our knowledge of what options the user wants for configuration variables. Recall that a configuration problem is a set of configuration variables $V_i \in \mathcal{V}$ each drawing options o_j^i from a set O_{V_i}. The MDP states are exactly the same set $\mathcal{V}$and each variable V_i has the same domain O_{V_i}but with one important difference: for each domain O_i, containing options $o_1^i, o_2^i, \ldots$ an extra value o_0^i is added. This additional value, which we call the "*unknown*" value, denotes that we actually do not know what option the user prefers for the configuration variable. Initially all variables are set to that unknown value.

3.2.2 Actions

Each action represents a prompt to the user to make a configuration decision, i.e. a *question*. This question is associated with a configuration variable we are interested in: The system asks the user what option she prefers for that variable. Let us denote that action as $V_i?$, where V_i is the variable in question. In response to this question, the user will answer with his chosen option for that variable. In MDP terms, asking the user a question and obtaining an answer is an action which causes the system to transition from one state to another state, changing the value of the corresponding variable from o_0^i (the "unknown" value) to some other value, based on the user response. But this response (and, consequently, the resulting state) is not known with certainty. We instead have a probability distribution over possible answers which we represent using the transition matrix.

Let us return to the email client problem. For the sake of the space in our illustration of the graph, let us suppose we have only two configuration variables V_{disp}, V_{ico}. Figure 2 shows the complete MDP model for this case with nine states (numbered in the states) and two actions. In each state, our knowledge over the values of the variables is labeled inside each state. The states in which all variables are known are the final desired states. In Figure 2 such states are emphasized by bolder border. In the initial state (the middle state distinguished by dashed border in Figures 2), we do not know any of the user's decisions, so the state is $\{o_0^{disp}, o_0^{ico}\}$. If we ask the user V_{disp}? in Figure 2 (i.e. whether she wants to

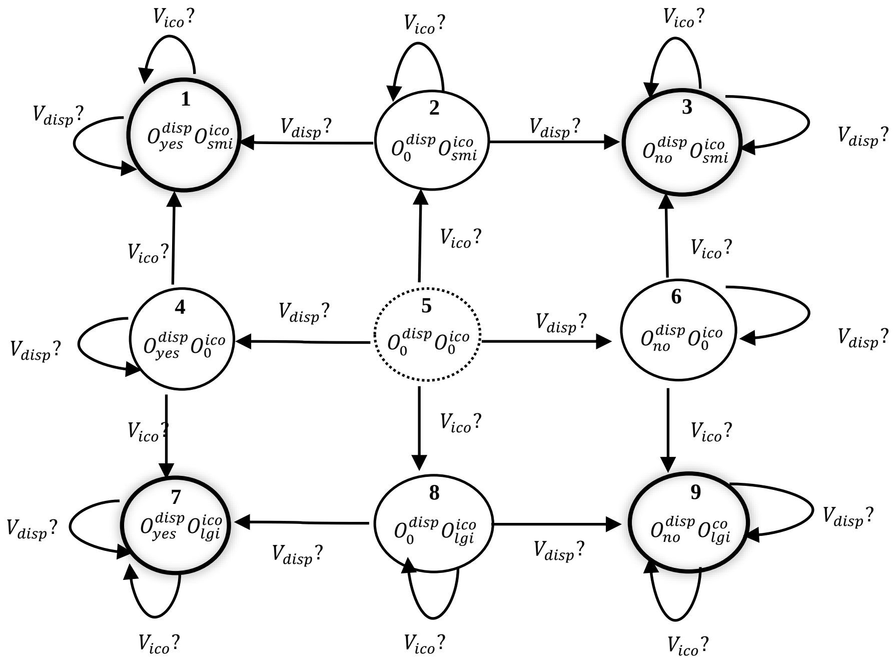


Figure 2 - The MDP model for email client

display text under the icons or not) the user will respond with one of o_{yes}^{disp} or o_{no}^{disp}. So the next possible states are $\{o_{no}^{disp}, o_{0}^{ico}\}$ or $\{o_{yes}^{disp}, o_{0}^{ico}\}$ each with its own probability which is calculated as described in the following section. If she said *yes*, then we can further ask V_{ico}*?* in Figure 2, which will lead us to one of $\{o_{yes}^{disp}, o_{lgi}^{ico}\}$ or $\{o_{yes}^{disp}, o_{smi}^{ico}\}$.

3.2.3 Transition Matrices

The transition matrix for each action V_i? is an *N*×*N* table, where *N* is the number of all possible option combinations of variables V_i, i.e. all possible states. Each cell represents the probability of transitioning from one state to the other as an outcome of performance of V_i? . In this case, the numbers are taken from the crowd data set. The idea is to measure the frequency by which a certain answer occurs, given (if applicable) all previous answers from the user.

More specifically, the probabilities for the transition matrices are calculated as follows: Firstly, it can be observed that only single-step transitions are possible. In other words, for now, only transitions from states of *n* unknown options to states of *n*−1 unknown options have non-zero probability; all other transitions have a probability of zero so value 0 is set to the appropriate cells. Note that this is bound to change once association rules are considered.

To see how the probability in the single step transition cases is calculated, let us consider a configuration problem with m variables $V_1, \ldots V_m$ and an arbitrary single-step transition within this problem from state

$$\{o_{x_1}^{1}, o_{x_2}^{2}, o_{x_3}^{3}, \ldots, o_{x_{n-1}}^{n-1}, \mathbf{o}_{\mathbf{0}}^{\mathbf{n}}, o_{0}^{n+1}, \ldots, o_{0}^{m}\}$$

to a state

$$\{o_{x_1}^{1}, o_{x_2}^{2}, o_{x_3}^{3}, \ldots, o_{x_{n-1}}^{n-1}, \mathbf{o}_{\mathbf{x_n}}^{\mathbf{n}}, o_{0}^{n+1}, \ldots, o_{0}^{m}\}$$

where x_i are non-zero values.

The probability that this transition occurs can be calculated by counting how many times crowd members select $o_{x_n}^{n}$ when they have already selected $o_{x_1}^{1}, o_{x_2}^{2}, \ldots, o_{x_{n-1}}^{n-1}$. In other words, the frequencies that correspond to the conditional probability are calculated:

$$P(V_n = o_{x_n}^{n} | V_1 = o_{x_1}^{1}, V_2 = o_{x_2}^{2}, \ldots, V_{n-1} = o_{x_{n-1}}^{n-1})$$

In cases where there are not any instances of the origin state in the sample dataset, simply the frequency of $o_{x_n}^{n}$ in the entire data set is used. The end result of this process is transition matrices which have zero in all their cells except for those which signify a one-step transition; in those cells the probability is calculated as above. Since some MDP solvers require that any action can be attempted at any state, in the state with a known variable, taking the action to ask that same variable will not change the state. The probability of such transition will be 1, since there is no other state to transfer to; the discussion on how the solver avoids such transitions proceeds.

Regarding our Email client example with the two variables, let us imagine that the distribution of the configuration options among the 1000 crowd members is as illustrated in contingency Table 2.

Table 2 - Contingency Table of preferences in Email Client Example

		V_{ico} (Size of icon)		
		o_{smi}^{ico}	o_{lgi}^{ico}	Sum
V_{disp} (Display text)	o_{yes}^{disp}	320	40	360
	o_{no}^{disp}	80	540	640
	Sum	400	600	

Based on the figures in the Table 2, 60% of users prefer large icons. Hence in the MDP graph, upon taking action V_{ico}? from the initial state $\{o_0^{disp}, o_0^{ico}\}$, the probability of transition to state $\{o_0^{disp}, o_{lgi}^{ico}\}$ equals 0.60. Similarly, out of the users who prefer large icons, which equals to 600 instances of users, only 40 users prefer the text of icons to be displayed. As mentioned before, the probability of transition between the source state and the destination state is in fact the conditional probability between the two: the probability of the latter happening given the knowledge that the former is already true. Hence the probability of transition from state $\{o_0^{disp}, o_{lgi}^{ico}\}$ to state $\{o_{yes}^{disp}, o_{lgi}^{ico}\}$ upon taking action V_{disp}? is calculated as follows:

$$P\left(\{o_{yes}^{disp}, o_{lgi}^{ico}\} \,|\, \{o_0^{disp}, o_{lgi}^{ico}\}\right) = \frac{40}{600} \cong 0.07$$

The calculation of the rest of transitions concludes the 9 x 9 matrices of transitions between 9 states. The transition matrices for the two actions as shown in Fig3 (a). The number associated with each state is shown inside each state in Figure 2, which correspond to the number of rows and columns in the transition matrix. As it can be seen, the sum of the probabilities of each row adds up to one, showing that every action can be taken in any state. It is noteworthy that although the size of the resulting matrices are big, they are highly sparse, which is an advantage in handling them for solving the policy.

3.2.4 Rewards and Costs

Recall now that in a completely defined MDP, each transition has its own reward while each action comes with a cost for performing it. The utility of a transition is calculated by subtracting the costs from the rewards. In the proposed formalization, each action comes with one (1) unit of cost. This cost is a direct representation of the effort required for a configuration decision, i.e. the effort for a human to think and decide about a configuration variable. Each transition, on the other hand, comes with as many units of reward as the number of variables whose preferred option becomes known. As such, the reward

$$\begin{matrix} 1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.8 & 0.0 & 0.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.36 & 0.0 & 0.64 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.07 & 0.0 & 0.93 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1 \end{matrix} \qquad \begin{matrix} 1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.89 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.11 & 0.0 & 0.0 \\ 0.0 & 0.4 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.6 & 0.0 \\ 0.0 & 0.0 & 0.13 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.87 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1 \end{matrix}$$

a) Transition Probability Matrices: Action V_{disp}on the Left; Action V_{ico} on the Right

$$\begin{matrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{matrix} \qquad \begin{matrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{matrix}$$

b) Reward Matrices: Action V_{disp}on the Left; Action V_{ico} on the Right

Figure 3 - Transition and reward matrices of Email client

represents the added value that we get for performing each action. As mentioned earlier, in case of a known variable V_i in any state, taking the action V_i? will necessarily result in a transition to the same state. Since such action has no reward and still bears a cost of 1, the utility of such transitions will be -1. This negative reward actually ensures that the policy does not consider this action and enquire about an already-known variable.

Back to the email client example, a transition from $\{o_0^{disp}, o_0^{ico}\}$, to $\{o_0^{disp}, o_{lgi}^{ico}\}$as a result of asking the question V_{ico}? comes with a reward of 1.0, because we found out about one variable, and a cost of 1.0 because one question was asked. Figure 3 (b) depicts the net reward of the email client. Notice that due to the cost of each action, the net reward value for actions that remain in the same state are -1 and the rest equal 0. If, however, we were somehow able to allow a transition from $\{o_0^{disp}, o_0^{ico}\}$, to $\{o_{no}^{disp}, o_{lgi}^{ico}\}$ for the same one question, then a reward of 2 is considered, because the options for two variables became known, while the cost remains 1, as one question was asked. As such the net utility would be 1 instead of 0. In the present initial state of the transition matrix, however, such transitions are impossible; this will be changed in the next section, though.

3.3 Revising the Transition Matrices

In order to take advantage of the association rules and find decision sequences with better utility, a second pass to the transition matrices is performed, which, if relevant association rules exist, allows us to replace one step transitions (transitions in which just one option is learnt) with multi step ones, i.e. transitions in which more than one options are learnt, yielding, hence, higher rewards. More specifically the procedure is done as follows.

Firstly, the association rules are reviewed. Of all the rules that were discovered during the mining phase, those that exceed certain high confidence *c* and support *s* threshold are filtered. Then a pre-processing step is performed in which rules that have identical left-hand side are merged. Such are rules of the form $V_x = o^x \rightarrow V_y = o^y$ and form $V_x = o^x \rightarrow V_z = o^z$, which mean that if we observe that $V_x = o^x$is a configuration decision then it can be assumed that both $V_y = o^y$and $V_y = o^y$are likely configuration decisions too. So the two rules are merged in one: $V_x = o^x \rightarrow V_y = o^y, V_z = o^z$.

The complete algorithm is shown in Figure 4. The subsequent steps are best described through reference to a directed graph. The set of nodes of the graph is the set of all possible configurations of $\mathcal{V}$; thus, each node is a state. Edges represent transitions. Initially, only the edges are added to the graph that are one step transitions, i.e. transitions from states in which $n-1$ variables are known to states in which n variables are known. Each edge from state S to state S' is labeled with a reward value of 1, a cost value of 1, a probability and the action it corresponds to.

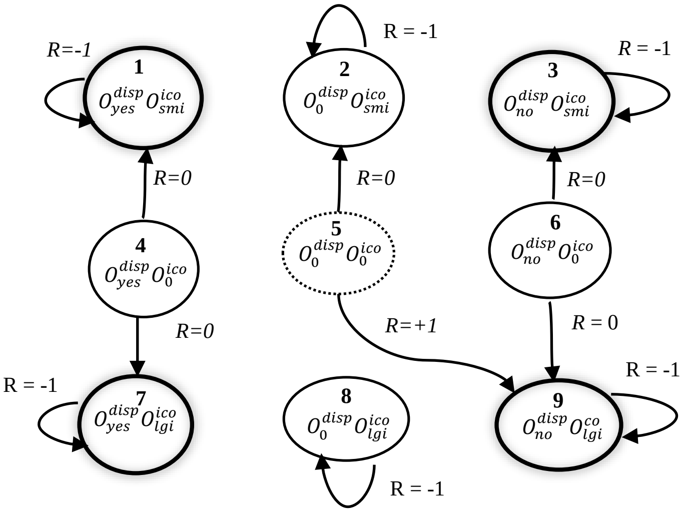
Subsequently, the edges are updated based on association rules as follows. Firstly, the states that correspond to the left and the right side of the association rule have to be found. However there is not only one such pair, if the system involves more variables than the ones emerging in the rule. If there are any variable(s) missing in the rule, that variable can be unknown or hold any value and the association rule still applies. Hence, for each association rule, all pairs of states S and S' are found such that: (a) the left hand side of the rule satisfies both S and S', (b) each variable of the right hand side of the rule appears with values exactly the same as in the rule in S, and with all values set to unknown in S' and (c) all other variables that are not mentioned in the rule are set to the same values for both S and S'.

```
INPUT: a set of configuration variables, their domains, a set of association rules
OUTPUT: A Markov decision process for optimal configuration sequences

// Build a graph of possible transitions
// Each node represents a state of knowing the option of each variable
For each possible combination of options
    Create a node in the graph

// Pre-process rules
For each set $m$ of association rules of the form $(L \rightarrow r_1, L \rightarrow r_2 \ , \ldots L \rightarrow r_m)$
    Merge into one rule of the form $(L \rightarrow r_1 \ldots r_m)$
// Add initial links
For each node $S$
    Let $N$ be the number of cases in the crowd data set.
    Let $c(S)$ be the num. of records in the crowd data set that satisfy $S$.
    Let $c(o_x^v)$ be the num. of records in the crowd data such that $V = o_x^v$.
    For each variable $V$ such that $V = o_x^v$ in $S$
        For each possible value $o_x^v$ of $V$
            Find node $S'$ that is the same as $S$ except for $V = o_x^v$ in $S'$
                Add an edge $e$ from $S'$ to $S$
                Weigh $e$ as follows
                    $e.reward$ = 1
                    If $(c(S) > 0)$ then
                        $e.probability$ = $c(S')$ / $c(S)$
                    Else
                        $e.probability$ = $c(o_x^v)$ / $N$
                    $e.action$ = $V_\propto$?

//Update the graph based on association rules
For each association rule $(l_1 \ldots l_n \rightarrow r_1 \ldots r_m)$, $r_i$ and $l_i$ being of the form $V_i = o_x^v$.
    For each pair of states $S$ and $S'$ such that:
        (a) Variables of clauses $l_1 \ldots l_n$ have known and the same options in $S$ and $S'$
        (b) Variable of clauses $r_1 \ldots r_m$ has unknown option in state $S$, known in $S'$
        (c) Variables not mentioned have same values in both $S$ and $S'$
        For each incoming edge $e$ to $S$
            Change the destination from $S$ to $S'$
            $e.reward$ = $e.reward$ + m

// Transform graph into MDP components
For each action $\propto \in A$
    Create an empty transition matrix for $\propto$, $T_a$.
    Create an empty reward matrix for $\propto$, $R_a$.
    For each edge in the graph from $S_i$ to $S_j$ such that $e.action = \propto$
        $T_\propto[i,j] = e.probability$
        $T_\propto[i,j] = e.reward$
```

Figure 4 - MDP construction algorithm

Secondly, for each such pair < *S, S'*> that is associated with the left and the right side of the association rule, all edges that are targeting S will be moved so that they now target *S*'. Moreover, the reward of each of those edges is increased by the difference in the number of variables predicted by the association rule. Intuitively, the rule causes S to become inaccessible: whenever we get a user response that leads us to *S*, thanks to the information in the rule, we can actually "shortcut" to *S*', ignoring *S*. While the cost stays the same, reward increases by the amount of unknowns that is discovered by taking the shortcut.

The resulting graph is ready to be transformed into transition and reward matrices. These are constructed initially to be zero matrices. Then for each edge labeled with the action at hand, the corresponding cell is set with the probability or the reward label of the edge, for transition and reward matrices, respectively. In the email client example, where we have one association rule of confidence 0.9, the graph can be revisited to shortcut the relevant states. Since in this example only two variables are involved, there is only one such pair of states <$\{o_0^{disp}\, o_{lgi}^{ico}\}$, $\{o_{no}^{disp}\, o_{lgi}^{ico}\}$> that relate to the association rule. For this one pair, since the transit between two states in the initial graph (Figure 2) corresponds to action V_{disp}?, only the transition matrix of this action will be modified and the transitions of action V_{icp}? remain untouched. Figure 5 shows the revised graph after the incorporation of the association rules. In Figure 5 for the sake of space and clarity only transitions and rewards of action V_{disp}? are displayed. Notice how the previous edge from state 5 to state 8, is now



R=-1
1
$O^{disp}_{yes}O^{ico}_{smi}$
R = -1
2
$O^{disp}_{0}O^{ico}_{smi}$
R = -1
3
$O^{disp}_{no}O^{ico}_{smi}$
R=0
R=0
R=0
4
$O^{disp}_{yes}O^{ico}_{0}$
5
$O^{disp}_{0}O^{ico}_{0}$
6
$O^{disp}_{no}O^{ico}_{0}$
R=0
R=+1
R = 0
R = -1
R = -1
7
$O^{disp}_{yes}O^{ico}_{lgi}$
8
$O^{disp}_{0}O^{ico}_{lgi}$
9
$O^{disp}_{no}O^{co}_{lgi}$
R = -1

Figure 5 - Revised MDP graph

changed to target state 9. The reward of this edge is also increased by one (i.e. the number of variables predicted in the association rule).

3.4 Policies and Decision Sequences

Given the above complete MDP formalization, an MDP solver is able to calculate a *policy* which maximizes the expected utility. Such policy offers a recommendation of which action should be performed when the system is at each of its possible states. In order to illustrate the policies with our email editor example, let us again suppose the system involves three (3) variables V_{disp}, V_{ico}, and V_{fnt}. In this case, policies can be represented using a tree structure as in Figure 6, where a policy for presenting configuration decision for the earlier email client problem is presented. The small circular elements represent states (labels are provided only for a few for the interest of space) and the boxes represent actions, i.e. questions.

As it is obvious in the Figure 6, each state is connected with exactly one action, the one that the policy recommends when the system is in that state. So in a case in which we only know that the user prefers to display text $\{o_0^{fnt}, o_0^{ico}, o_{yes}^{disp}\}$ the policy recommends that action V_{ico} be performed, i.e. ask the user what icon size to use. This way each path from the root of the tree structure (i.e. a state in which we do not know anything about the user's preferred configurations) to one of its leaves (i.e. a state in which we know the options for all variables), represents a unique sequence of questions to be asked to the user. Which path exactly is going to be followed depends on the answers the user will give to each question.

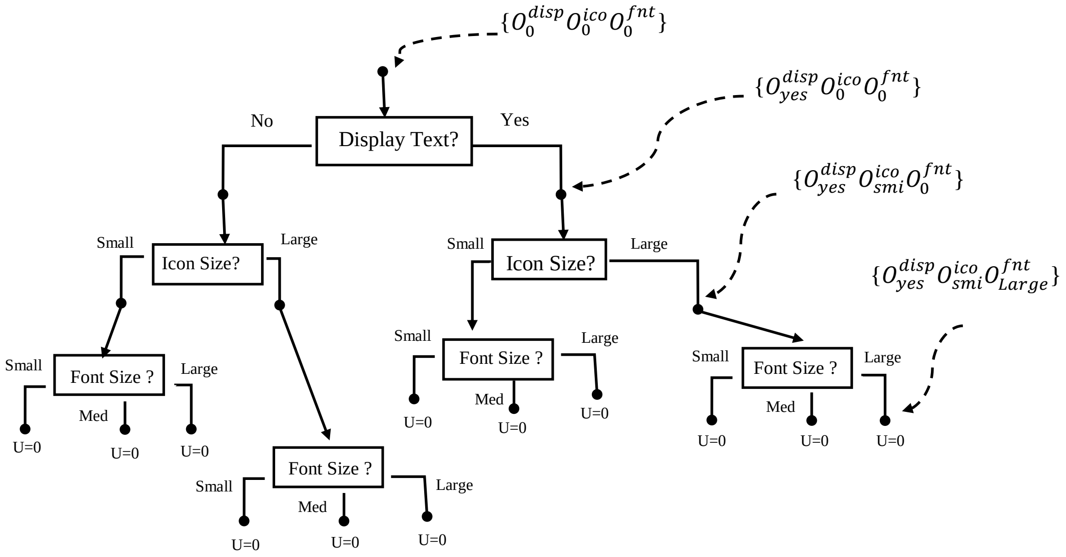


Figure 6 - A policy tree

The policy comes with an expected utility, which recursively at each step results from multiplying the utility of a transition and the probability of getting the transition and adding it to a total sum. Recall that the utility aspect results from subtracting cost from reward and is a representation of the configuration effort that is saved by running the configuration wizard. As seen before, in the initial situation in which every action transfers to another state where just one more variable is known, the total utility remains zero. As seen in Figure 6, where a policy coming out of the unprocessed MDP is presented, whatever benefit is gained by learning the preferred option for a particular variable, is paid by bothering the user with yet another decision task.

However after processing the MDP with the crowd configuration preferences the picture is quite different. Suppose in case of the email client with three (3) variables V_{fnt}, V_{ico}, and V_{disp}, the following association rules emerge:

$(V_{fnt} = o_{lg}^{fnt}) \rightarrow (V_{ico} = o_{lgi}^{ico})$

$(V_{ico} = o_{lgi}^{ico}) \rightarrow (V_{disp} = o_{no}^{disp})$

Having updated the configuration matrices based on the association rules, the MDP solver will return optimal policies with positive expected utility. One such policy can be seen in Figure 7. Weighing the benefit of a specific answer in the font size question, the policy recommends $V_{fnt}?$ to be the first question. If the user answers o_{lg}^{fnt}, which happens with a certain probability, the process is lead to a final state $\{o_{lg}^{fnt}, o_{lgi}^{ico}, o_{no}^{disp}\}$ in which all variable options are actually known. The reward for that transition in which three variables are known equals three (3) and the cost of making a decision equals one (1). As such the utility equals two (2). Thus, the difference between this policy and the one of Figure 6, which did not consider association rules, is that now several branches can lead to a positive utility value, depending on user behavior. The MDP solver moreover guarantees that the policy tree it returns offers, on average, the highest utility, i.e. the most configuration knowledge with the least decision effort.

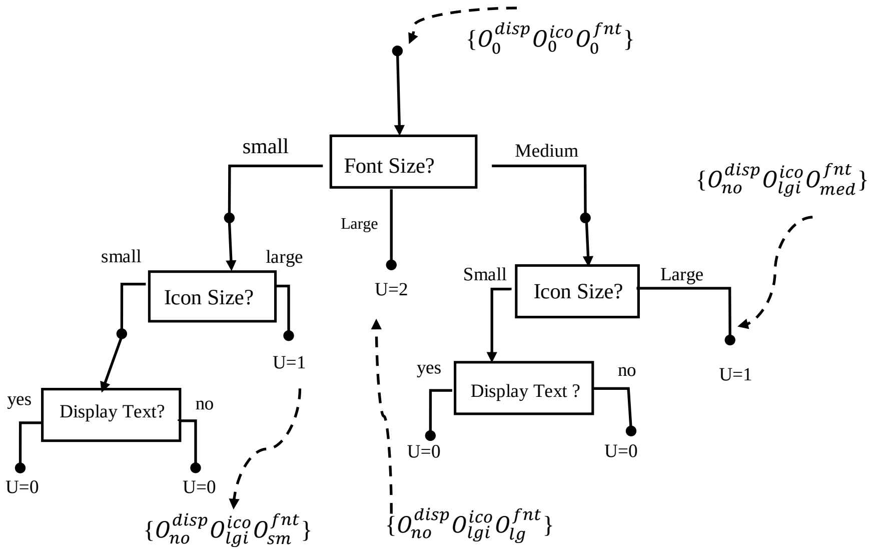

$\{O_0^{disp} O_0^{ico} O_0^{fnt}\}$
small
Font Size?
Medium
$\{O_{no}^{disp} O_{lgi}^{ico} O_{med}^{fnt}\}$
Large
small
large
Icon Size?
U=2
Small
Icon Size?
Large
U=1
yes
Display Text ?
no
U=1
yes
Display Text?
no
U=0
U=0
U=0
U=0
$\{O_{no}^{disp} O_{lgi}^{ico} O_{sm}^{fnt}\}$
$\{O_{no}^{disp} O_{lgi}^{ico} O_{lg}^{fnt}\}$

Figure 7 - Updated policy tree

3.5 Summary and Discussion

In this chapter, the proposed method for construction of the model in collaborative recommender was explained in detail. For the model of the recommender, we use MDPs to solve the optimal configuration dialogue possibilities in our conversational system. In the MDP model for our configuration problem, each action is a question that the elicitation agent can ask the user. The content of each question is a configuration variable and its answers are possible options for the variable. The user will respond with a preferred option with a certain probability. The user’s response leads the system from a state in which the elicitation agent knows the preferred options for a smaller subset of the configuration variables, to a state in which it knows them for a larger subset.

We aim to decide the optimal sequence of dialogues with the criterion to minimize the number of interactions between the user and the system and thereby minimize user effort. To this end, association rules are used to bypass transitions that are inferred by the crow data. The solver will find optimal policies for the elicitation agent, i.e. what question to ask given the answers already provided by the user. The optimality of such policy is based on the utility model for actions and states, which reward questions whose answers are likely to help us perform more configurations automatically.

In order to propose a generic framework that is domain-independent, we assume that no prior information about the users and their preferences is available beforehand. With the absence of any user profile, preferences need to be elicited in the process of recommendation. Therefore we treated the problem of customization of configuration items as a sequential decision making process through conversational process with the user: the program inquires explicit user’s preference for a subset of item attributes in the course of normal recommendation dialogues and the user responds.

To find the best sequence of questions to ask, MDPs are an appropriate choice due to many reasons: MDPs solve the sequence of questions by incorporating the probabilities and benefits of knowledge over the preference of already known items into consideration. The optimal policy, for instance, might decide to inquire the user about a configuration variable, which compared to another variable, is able to predict fewer (unknown) variables; this decision is supported by the fact that the variable with less immediate reward can lead to more likely or more predicted variables in subsequent steps. In other words, the MDP solver will search for a global optimum.

Therefore in our proposed model, using the crowd preference for configuration variables, the MDP model is constructed and the optimal policy is solved off-line to determine the sequence of questions to inquire the user about the preference of variables. At the time of making the recommendation, the already-solved policy is readily used to decide the sequence of questions (i.e. in dialogues).

Chapter 4 : Case Study

In order to evaluate the applicability of the proposed technique in a real-world system, we applied the technique to a subset of configuration variables of the social networking tool Facebook. The data set we used comes from a crowd of York University students who shared their Facebook configuration options with us. The goals of this evaluation include finding whether extraction of sensible rules is at all possible and whether the policies that result from our MDP-based approach do really save configuration effort. This chapter presents the results of this study in detail.

4.1 Case Study: Facebook

Facebook[1], the biggest Social Network Site (SNS) today, is reported to accommodate more than 1.28 billion monthly active users (April 2014). Its users spend substantial time interacting with it and uploading large amount of sensitive information [64-65]. In an SNS such as Facebook users can (1) construct profiles with customized visibility and degrees of information sharing, (2) define the list of their connections as friends within the network, (3) avail in the personal profile area a variety of material (photos, text, information etc.) to be viewed and commented upon by other users who have been given access to, and (4)

[1] www.Facebook.com

interact with the network of friends and the users in the network (e.g. view their information, or communicate) [66].

Facebook offers a set of configuration variables which the users can configure in order to meet their privacy and other system use needs. To facilitate the management of information sharing, Facebook allows each user to customize fine-grained access control of others to one's personal data: For instance, users can specify the exact connections who are entitled to see their posts, pictures, events, etc. Moreover, to facilitate the access control, Facebook allows users to group their friends in different lists in order to customize the access levels of the shared information based on their differing relationships.

The aforementioned variables are mostly configured using the traditional approach: a set of default options is chosen upon creation of a new account. At any time, users can adjust them by going through a set of configuration screens, where the variables are presented together with possible options. Given the personal material on Facebook is often of sensitive nature, how each user can control the access levels of others to her data is crucial. In fact, privacy in Facebook has been a subject for intense research [37, 64, 65-71] as well as discussion in popular media. Many studies on Facebook have acknowledged the difficulty that users face in configurations of the settings [64, 67-71]. Research studies indicate that not all users are aware of the implications of the privacy settings [17, 71] and that users tend to overestimate the strength of their chosen privacy in Facebook [72]. Moreover the default setting is mostly inclined to share information more broadly [3], which is alarming since a great percentage of users are reported to keep the default settings [72]. Hence, the problem of privacy configuration options for Facebook offers us a good example of why assisting users to configure their systems becomes more and more important.

The current literature on supporting the customization of Facebook is mostly based on the notion of automatic detection of friend lists to predict the friend list(s) that a connection falls under [72, 73]. These approaches use clustering techniques and use information such as similarities between two users (e.g. gender, location, and company), extent of interaction between them, and their common connections to classify friends [74]. Such commonalities form the basis for sharing configurations too. One of the aforementioned studies proposes a wizard that requires the user to decide upon the access level of a subset of friends and hence allocate the same access to the rest of the members of that cluster [72]. This proposal can be the basis for extending this repertoire of Facebook configuration approaches with a solution that does not depend on the attitudes of Facebook friends but of an arbitrary set of expert users.

4.1.1 Configuration items in Facebook

In this case study, in order to demonstrate the application the proposed approach, a small set of nine (9) items in Facebook configuration variables were chosen. These items with their relevant domain values are shown in Table 3. It is noteworthy that the location of the variables changes fairly often in Facebook interface, but in April 2014, the items for this study were located under the following three sections: Privacy, Timeline & Tagging, and Friends. The domain of variables (i.e. the set of possible values) for these configurations vary from static options to customized dynamic lists: The domain of settings for 1.c and 2.d consists of static options (such as "Yes" and "No"), while the rest refer to access levels of different connections in the network and their domain consist of a subset of five possible values ("Public", "Friends of Friends", "Friends", "Custom", "Only me"). Under "Custom" settings, users can specify any subset of all their connections in one's network; in this study this option is treated as a coarse-grained category without looking at the form of those lists.

Table 3 - Facebook Privacy Configuration Options

1. General Privacy
    - a) *Who can see future posts?*
    - b) *Who can look you up using the email address or phone number you provided?*
    - c) *Do you want other search engines to link to your timeline?*

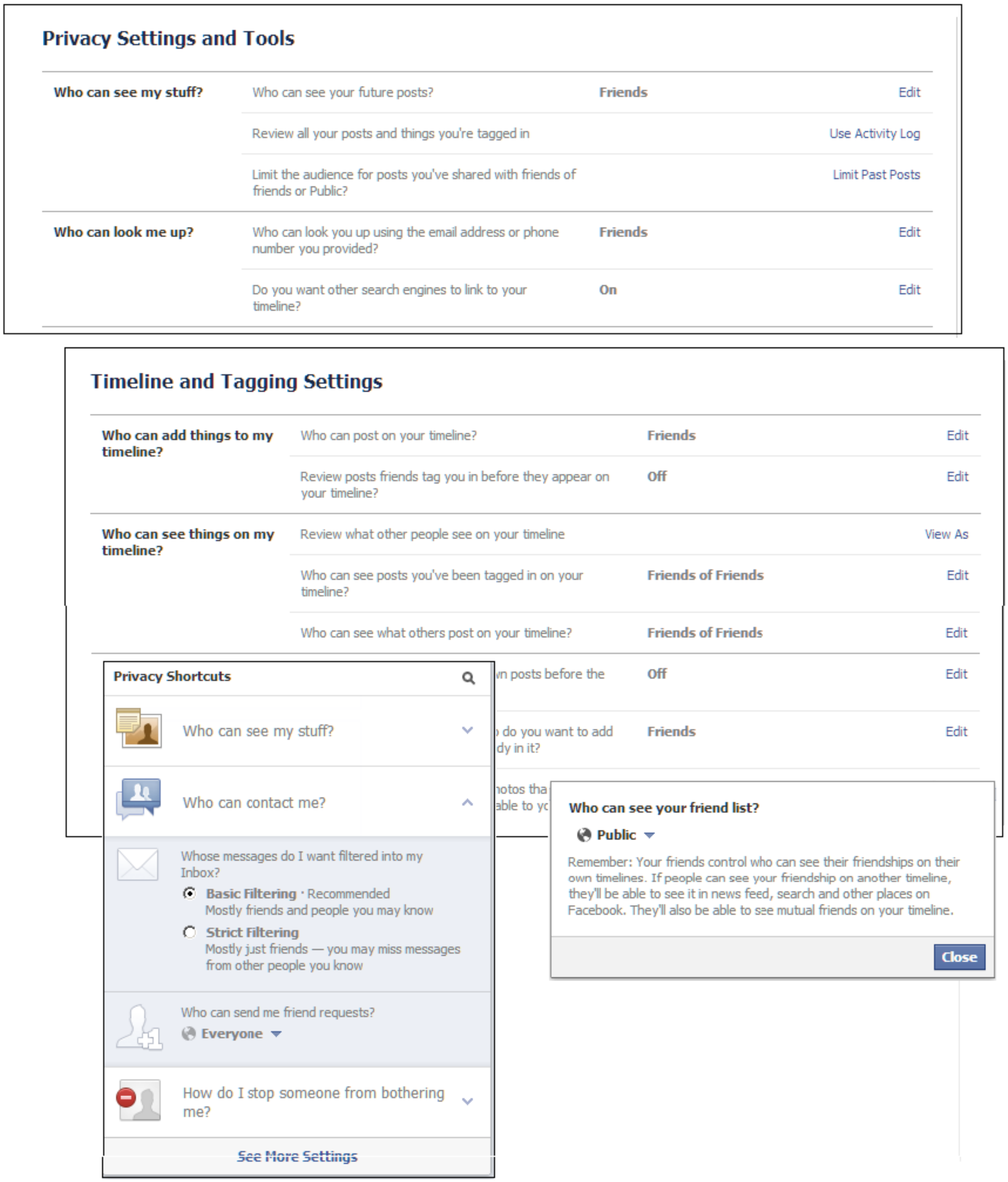
2. Timeline and Tagging
    - a) *Who can post on your timeline?*
    - b) *Who can see what others post on your timeline?*
    - c) *Who can see posts you've been tagged in on your timeline?*
    - d) *Review posts friends tag you in before they appear on your timeline?*

3. Friend List Related
    - a) *Who can send you friend requests?*
    - b) *Who can see friend list?*

4.2 Data Collection

To test the feasibility of approach in real settings, the data regarding users' preferences for configuration items of Facebook was collected from undergraduate students of the School of Information Technology during March and April of 2013. The participants in the experiment were attending Prof. Liaskos's class on Human Computer Interaction and they were given bonus marks for participation.

Since the literature reports on a huge gap between users' expectation (ideal setting) and what they actually implement in their account [72], we decided to collect both actual and self-reported configuration information. Users were initially invited to participate in an online questionnaire, where they were asked about their preferred options for the nine configuration items in Table 3. Only when the questionnaire was fully submitted, in order to collect their real adopted customization, participants were asked to take screenshots of the configuration screens that contain the variables in question and send them over to the researchers – after erasing any personal information may exist in those screenshots. Figure 8 shows a sample screenshots of a user. The result is a data set of 45 users amenable to association rule mining and application to this research framework.



Privacy Settings and Tools

Who can see my stuff?	Who can see your future posts?	Friends	Edit
	Review all your posts and things you're tagged in		Use Activity Log
	Limit the audience for posts you've shared with friends of friends or Public?		Limit Past Posts
Who can look me up?	Who can look you up using the email address or phone number you provided?	Friends	Edit
	Do you want other search engines to link to your timeline?	On	Edit

Timeline and Tagging Settings

Who can add things to my timeline?	Who can post on your timeline?	Friends	Edit
	Review posts friends tag you in before they appear on your timeline?	Off	Edit
Who can see things on my timeline?	Review what other people see on your timeline		View As
	Who can see posts you've been tagged in on your timeline?	Friends of Friends	Edit
	Who can see what others post on your timeline?	Friends of Friends	Edit
	n posts before the	Off	Edit
	do you want to add dy in it?	Friends	Edit
	hotos tha able to yo		

Privacy Shortcuts

Who can see my stuff?

Who can contact me?

Whose messages do I want filtered into my Inbox?

● **Basic Filtering** · Recommended
Mostly friends and people you may know

○ **Strict Filtering**
Mostly just friends — you may miss messages from other people you know

Who can send me friend requests?
Everyone

How do I stop someone from bothering me?

See More Settings

Who can see your friend list?

Public

Remember: Your friends control who can see their friendships on their own timelines. If people can see your friendship on another timeline, they'll be able to see it in news feed, search and other places on Facebook. They'll also be able to see mutual friends on your timeline.

Figure 8 - Sample snapshots of Facebook configurations

4.3 Goals and Method

The research questions we are looking to answer in this study are as follows:

A. Do any association rules of substantial significance emerge?

In order to answer this, using a popular data-mining software, we mine the association rules with varying levels of confidence and support. Then we simply qualitatively examine the results of association rule mining to see if any significant number of patterns is detected in the preference crowd.

B. How much configuration effort on average does this technique result to?

To address this question, we investigate how many configurations on average a user has to explicitly customize so that the rest of the configuration items can be automatically configured. The recommended items are hence the effort that has been saved. To find the number and sequence of items that needs to explicitly be questioned, the following steps are performed:

i. Firstly, the MDP policy is solved for once given the 45 sample data. Then, with the optimal policy at hand, for each user, we start from the initial state in which all variables are unknown and do the subsequent step.

ii. At any given state, the solved policy of the MDP model in step (i) determines the next question to ask the user. To simulate the user response to this question, his/her chosen option for this configuration variable is simply looked up from the collected dataset. Hence, knowledge over the value of the latest item will transfer us to a new state, in which one or more additional variables are now known.

iii. Step (ii) is repeated until a state is reached in which all variables are known. The configuration at hand, thus, represents the recommended configuration and the number of steps represents the total effort.

iv. Total effort in step (iii) is calculated and averaged over the 45 instances.

C. What is the average accuracy of the automated inference that this technique implies? In other words, by automatically predicting a configuration option based on another one, how accurate are the predictions?

i. To investigate the accuracy of the recommendations, K-fold cross validation (K=9) technique is used on the sample dataset.

ii. In each fold, using the 40 instances as the training data, the optimal policy is once solved.

iii. For each 5 instances of test data, the final state of the MDP (similar to step ii in question B) is found. This final state captures the value of any predictions made, as well as the values inquired from the user (i.e. looked up from the data set). For any predicted variable, the predicted value is matched against the real user's choice from the available dataset. The accuracy of predictions for each test user is hence calculated as the ratio of the number of variables correctly predicted out of all predictions made.

iv. The accuracy of each fold is calculated by averaging over the rates for 5 users.

v. The total accuracy equals the average over the accuracy rates of 9 folds as calculated in step iv.

4.4 Results

4.4.1 Participants and Responses

The participants consisted of 35 male and 10 female undergraduate students of School of Information Technology in York University. The majority of users (62%) were Facebook members for at least 5 years. Only one participant was a member for a duration of less than a year. Regarding the extent of use, about 9 percent of users declared to use Facebook less frequent than once a month, while the majority of 62.2% use it for at least once a day. Amongst all respondents, the most common responses for the time spent on the site each day were less than 10 minutes (53%) and between 10 and 30 minutes (20%). The users on average had more than 300 connections in their friend list (median=250); 3 users had less than 50 connections and 2 users had more than 1000 friends in their network.

Our comparison between the questionnaires and the snapshots confirm the gap between real configurations implemented, and what users think as reported by literature [72]. When the users were inquired about the settings, out of 9 items, they believe on average they have changed 70% of the settings, while the real implemented change from the defaults is 60%.

The reasons underlying this cleavage and the decision on which one truly reflects the preferences is debatable. As argued by Liu et al. [72], it can be assumed that the reported settings reflect user's ideal options and that they have failed to truly implement their preference as expressed in the questionnaire in the real settings. On the other hand, self-reporting always runs the risk of lesser reliability than direct observation of snapshots, as it is unknown how mindfully participants have responded. For the rest of the analysis, we take the data from snapshots into the account which we know are reliable. From here on, we refer to the snapshots as *preference of users.*

4.4.2 Findings

Main Results

The association rules in the data at hand were mined using Weka[2]. Different thresholds for significance as well as support were used for this experiment. By trying significance threshold = 0.9 and support 1, 738 association rules were mined. The maximum significance (1) yielded as many as 675 association rules. By increasing the support level to 0.15, still as many as 165 rules were retrieved. The above means that, with regards to question (A) of the research, association rules exist to a great extent in the collected sample data set. It is worth noting that on average only 39% (median=34%) of the configuration

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

samples corresponds to the default, meaning that the presence of association rules are the result of active configuration effort.

With respect to configuration effort – question (B) of research goals – if each of the users had followed the proposed technique to configure their system they would take on average 6.6 steps (max 9, min 4); meaning that at least two configuration options on average (27.7%) would be decided automatically. Although, in general, effort savings is bound to depend on the amount and quality of correlations within the data set, we find that these results constitute good evidence of the practicality of the proposed technique.

In terms of evaluation of results addressed in question (C), i.e. the precision of configuration predictions, the experiment was conducted with confidence level of 0.1 and support of 0.15. The K-fold cross validation (k=9) shows a precision of 75% in the recommended items, calculated as explained above. The fact that this precision score is away from values such as 50% or lower, tells us that the configuration agent preforms sensible decisions based on the data and not e.g. randomly/arbitrarily.

Note that this accuracy measurement depends on the degree by which we believe that each configuration in our data set accurately reflects the preference of the user and truly meets his requirement. In this particular study, as there is no reliable criterion of “trueness” of each configuration or of Facebook expertise, the accuracy measures only imply that the resulting configuration policy generally reflects user’s selections, with a necessary margin of error. However, other than an indication of basic sanity of the approach, we should not view this result as a measure of success of the recommendation. To see why this is so, suppose a user in our test data is a novice one and has not made good choices in customizing his system. Matching the user's choice with the recommended one to measure the accuracy would hence imply that we expect the recommendation system to still recommend the bad options. Otherwise, especially if the association rules come from an expert data set, configuration recommendations to a novice user are actually an indication of usefulness of the method. A more comprehensive approach to measure the success of a recommendation requires assessment of a few success criteria in recommendation systems; we discuss the challenges of the evaluation of the RS in the last chapter.

Other Findings

Since we were given the opportunity to access real configuration data in this study, we performed more data exploration in order to better understand relevant configuration behaviors and detect possible influences to the result of our technique. Firstly, no significant relation was detected between the extent of activity in Facebook and the tendency to change the default settings. For this experiment, variables such as frequency and duration of application usage, number of friends, and duration of membership were assumed to imply extent of activity. We found that majority of users (median 66%) have made the privacy stricter. On average, users kept 3 out of 9 default value of the configuration items and customized the rest; in only 31% of user instances, a looser setting for at least one configuration item was chosen, whereas all users have made at least one item tighter. Figure 9 illustrates the frequency of configuration preference of the crowd in comparison with the default option. On the right number of configuration items left at the default options is illustrated, whereas on the left the number of items customized to tighter setting is shown. As the skewness of the histograms shows, greater number of items are customized to tighter settings.

Deeper analysis suggests that users who have customized their configurations to tighter measures (compared to the default options) may enjoy higher precision of predictions. Kendall's tau_b Correlation test suggest that the (discretized) precision of predictions is significantly correlated with the number of items that are tighter than the default (p<0.05, τ =.27). The number of the items left as the default is also, although with less significance, negatively correlated with the precision of recommendations (p<0.1, τ =-.24).

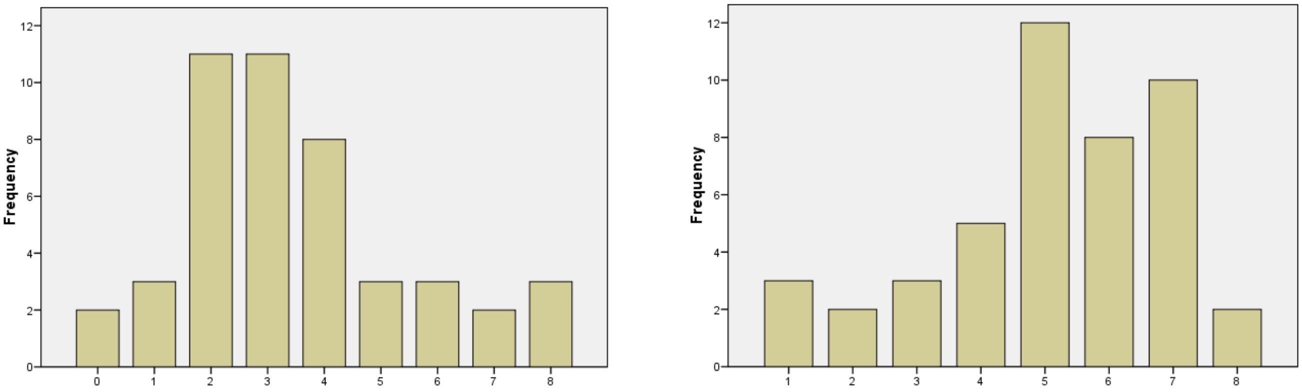


Figure 9 - Facebook configuration of crowd in comparison with default option

(a) number of items equal to default option on the left, (b) number of items tighter than default on the right

In other words, in our experiment the accuracy of prediction is higher for people who have changed the defaults to tighter measures. This can be due to the fact that the recommendations of collaborative systems are biased towards the popular trend. As shown by the earlier discussion, the majority of our crowd seem to be privacy-cautious and have changed the defaults to tighter options; hence it is reasonable to see that users who diverge from the popular pattern of the crowd and prefer looser settings will receive lower accuracy in recommendations.

4.5 Threats to Validity

In this study, we aimed to measure the accuracy of predictions as one of the means to evaluate our proposed recommendation system. In this section, we investigate the internal and external validity of this study and explore potential threats to validity with regards to the measurement of accuracy.

The dataset of our experiment was collected from the crowd through a questionnaire as well as screenshots of current system configurations. By taking the screen shots of the user's system into the account instead of reported values that might be subject to error and forgetfulness we avoid one major internal validity threat: what we measure as configuration is actual. Since the screenshots are real customizations and of the active accounts, we ensure the accuracy and reliability of the data and hence we did not need any data cleaning process to exclude wrong or conflicting data entries.

In our proposed approach, we aimed to recommend the configurations based on customizations of reference expert crowd. In the absence of crowd of experts, in whose selected settings we have faith, we had to dispense with a sample of invalidated users. Since the degree of ability of sample users in handling their configurations is unknown, the accuracy results are interpreted in a conservative way.

Regarding the external validity, although the approach to solving the research problem in this study ensures the independency of the solution from the domain, future experiments are needed to confirm the applicability of the solution in other kinds of systems and configuration aspects. We nevertheless feel that the huge popularity of Facebook and the fact that privacy is a common concern across user groups and domains, the configuration set we used is fairly representative. With respect to the choice of subjects, there is, obviously the threat of selection bias of non-random sampling since all participants were recruited from one undergraduate class at the university. All users in the experiment were IT students and shared similar biography (age, place of living) to a great extent, and hence are likely to share similar requirements and goals in customizing their configuration. Thus, although our study does not rely on inferential statistics, a more heterogeneous sample may need to be sought in the future.

4.6 Summary

In this chapter the proposed solution to personalization of configurations was applied in a real life application setting. In order to put our idea into test, the preferences of a crowd of 45 users in customization of the configurations in Facebook was collected. The results confirm that first and foremost, patterns of association with high significance would merge in the crowd, which is a requirement in our proposed approach. Using the discovered rules, on average users were able to skip 27.7% of the steps necessary to have their whole configurations figured out. Furthermore the recommendations show a good rate of precision, which we interpret as a good sanity indicator for our technique. As discussed this measure alone should not be considered as a measure of success of the recommendation. We discuss the limit to this measure in the last chapter.

Chapter 5 : Performance

In the previous chapter, the proposed framework was evaluated in a case study to assess the accuracy of the generated recommendations. In this chapter, we investigate the scalability of the method and evaluate the performance of the recommender with regards to larger sizes of configuration domains.

5.1 State Space

While the proposed MDP formulation allows for good quality and fine-grained analysis, MDP solving is unfortunately a computationally hard problem [26, 75]. As such, it is important to explore how the performance of the MDP solver varies as the number of configuration items and hence state space increases.

One of the main factors that affect MDP solving performance is the number of included variables and the sizes of their domain. In the absence of an actual configuration data set of sufficient size for the performance experiments, and without visible loss of generalizability, we chose to use a publicly available benchmark data-set for association rules. In particular, UCI KDD's archived Mushrooms dataset [3] was used, which has been shown to generate long patterns with high confidence. In its full size, the Mushrooms dataset is a multivariate dataset that contains 8416 instances with 23 categorical attributes

[3] https://archive.ics.uci.edu/ml/datasets/Mushroom

and domain size ranging from 2 to 12 categorical values. In the context of software configuration, this data set was used as if it was a complete configuration data set: each attribute is assumed to be one configuration variable and each value a configuration option. Consecutively, each instance (i.e. each row of the dataset) hypothetically stands for the configuration values chosen by a user.

The MDP solver we chose is the “Markov Decision Processes (MDP) Toolbox” offered by MathWorks Inc[4]. The MDP toolbox was run in Matlab R2012b. The input to the MDP solver is sparse matrices that are generated by a Java program and are based on the extracted association rules from Weka. The java program prepares the input matrices of the MDP solver and the algorithm is discussed in Chapter 3 (Figure 4). The experiment was conducted on a Core(TM) i5 CPU M450 2.40 GHz with 4.00 GB RAM under Windows 7.

To test the performance of the MDP solver for various numbers of configuration items, a subset of varying numbers of attributes in the dataset was randomly chosen. For each subset of attributes, the top 500 association rules of confidence 100% and minimum support of 10% were extracted. In Table 4, the time to solve the policy is depicted with respect to the

[4] http://www.mathworks.com/matlabcentral/fileexchange/25786-markov-decision-processes–mdp–toolbox

number of states. The number of states is the product of all domain sizes of all considered variables.

To get a sense of configuration problems that can be solved if such numbers of states are available, Table 4 also illustrates time in relation to the equivalent number of binary configuration variables. Specifically, N binary variables generate a space of 3^N states, and, as such, if M is the number of states, the equivalent binary variables are $log_3(M)$. Note that the base is 3 instead of 2, because one more value (the "unknown" value) is added in the domain. Figure 10 repeats the figures of Table 4 for the time to solve the MDP versus the number of dichotomous configuration items in a performance curve. As expected, the complexity of solving the problem grows exponentially with the number of variables.

Table 4 - Performance vs. State number

Number of States	Binary Equivalent	Time (s)
625	5.9	0
3125	7.3	0
10000	8.4	8
12500	8.6	1
30000	9.4	1
49000	9.8	5
50000	9.8	10
90000	10.4	31
150000	10.8	64
270000	11.4	220
450000	11.8	645
2116800	13.3	14484
3300000	13.7	31161
4410000	13.9	55579

5.2 Range of Associations

Another question we explored is whether the number of the association rules would affect the performance. To find this, a fixed subset of nine (9) attributes (equivalent to 97.2K states) in the Mushroom dataset was experimented on. The number of generated rules can be controlled via varying support levels; we hence experimented with support ranging from 0.3 to 0.6, while keeping a fixed significance level of 1.0. This way, the more the support level threshold is increased, the less rules are mined that exceed the threshold. Table 5 illustrates the time to solve MDP with regards to the support level and number of extracted association rules. The results do not seem to indicate any pattern; they instead show a rather consistent performance regardless of the range rules.

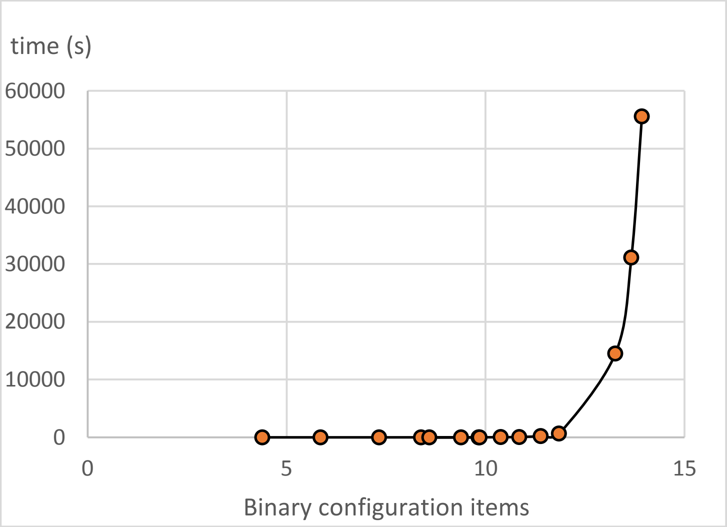


Figure 10 - Performace vs. number of binary variables

Table 5 - Performance vs. Number of Rules

Support (%)	# Rules	Time (s)
0.30	318	28.7
0.35	157	28.6
0.40	97	29.1
0.45	85	29.5
0.50	76	28.3
0.55	33	28.3
0.60	10	30.5

5.3 **Threats to Validity**

In the absence of real configuration data for larger domain size, for the performance experiment, we adopted Mushrooms Dataset, a benchmark dataset of gilled mushrooms. The rationale is to use a dataset that demonstrably contains some amount of association rules, which is the premise of our technique. Nevertheless, whether either the structure of the data (e.g. sizes of attribute domains) or the structure of the resulting association rules are not typical of configuration problems is currently unknown and poses a threat to our internal validity. Obviously real configuration data sets (which are not easy to find) would be more preferred. Nonetheless, it is not visible to us how exactly the structure of either the domains or the resulting rules would impact performance, state space and action set size being equal.

5.4 Summary and Discussion

In this chapter, we explored the performance of the proposed approach by increasing the number of the variables involved. We chose a publicly known dataset on gilled mushrooms and used it as a dataset of configuration items to simulate this experiment. Our experiment showed that the performance of the proposed approach is independent of the range of associations in the dataset. This may also indicate that the performance is independent of the nature of the domain; however this assumption needs future investigation. The increase in the number and size of the variables however, proved to affect the performance exponentially.

Our technique showed to be able to handle the equivalent of as many as 14 binary variables, which seems to be applicable to smaller problems - such as the Facebook case. Furthermore, the measured performance in this chapter addresses construction of the model, which given that it is being done offline seems affordable. Otherwise, as other model-based recommendation systems, the online performance in making the recommendation is the fastest possible, since there is no calculation involved.

To improve scalability, simpler techniques can be conceptualized for addressing the problem of saving configuration effort that should be expected to scale much better than the one we presented. For example, one can traverse the association rules in order to rank configuration options with respect to how influential they are, i.e. how long chains of automatic configurations they trigger. The ranking of the corresponding variables can then be the sequence of questions to ask the user. Simple heuristic-based approaches like this, although scalable and definitely a subject for future experimental research, do not guarantee optimality. This is due to the fact that we e.g. do not take into account the probability of other options being chosen. These approaches do not result in the advanced adaptive conversational recommender we propose, either.

Thus, for our technique we opted for sophistication and good quality of result rather than scalability using more trivial methods. Nevertheless, it is certainly desirable that our approach supports larger numbers of configuration variables and options in the future, while maintaining the same good quality of results (optimality). In the next chapter we discuss an approximation to modeling the states in order to decrease the size of the space.

Chapter 6 : Conclusion, Limitations and Opportunities

6.1 Contribution

As modern software systems increase in size and complexity, the process of configuring them becomes more cumbersome and difficult to organize. Users, especially novice ones, have the additional trouble of not knowing which system configuration, among the thousands possible, is more suitable for their unique needs. Hence, assisting users in intelligent customization of software configurations can increase system usability and allows non-expert users to fully benefit from the applications.

In this work, a method was presented to assist configuration of software systems by consulting the configurations that a crowd of expert users has already in place. First the expert data set is minded for crowd configuration preferences, which come in form of association rules. Such rules indicate that users who make certain choices for a subset of the available configuration variables are also likely to make specific choices for other variables outside that subset. The resulting association rules allow us to automatically set certain configuration variables given the available knowledge about other configuration variables. The objective is then to configure the entire set of variables in a way that the number of variables that are automatically set is maximum. This both reduces the configuration effort and ensures that the system is more expertly configured. The highlight of the proposed method is the fact that it is generic; it is independent of the knowledge of the domain and can be applied in any field.

Thus, our proposed framework has the following benefits. Firstly, it is a generic recommendation system for assisting software configuration, a type of recommendation system which, to our knowledge, has not been studied before. Secondly, it features an interactive dialogue-based model for interwoven elicitation and construction of user preferences when no information about the user is available beforehand. Thirdly, through the use of MDPs we are able to globally optimize the length of the user-machine dialogues and, as such, minimize configuration effort.

6.2 Discussion

6.2.1 Assumptions

Our approach adopts a certain stance with respect to what a configuration problem is and how it is modeled. Firstly, we assume that the configuration of the software system is represented through configuration variables with finite discrete domains. If this is not the default format of configuration variables a transformation to a compatible format is necessary. For example, continuous variables need to be brought to a discrete format. Secondly, the proposal focuses on sequential configuration of a set of variables at the same time, as it would happen when, for example, installing or using a system for the first time or when the user or administrator decides to perform a general reconfiguration of the system. Finally, the proposed approach was developed with the configuration of common personal software systems in mind. Such are systems of wide popularity among large numbers of users who might be non-technical and not computer experts. Social networking applications, blogs, email clients or calendar applications are examples of such systems. Theoretically, as long as the previous conditions are met, our approach can be applied to business and specialized software applications as well, though this needs to be further validated.

6.2.2 Choice of Recommendation System

The recommendation system we proposed is based on mining association rules and using MDPs. However, there exist different recommendation techniques, as discussed in the related works. So why do we develop our own in this thesis? In choosing the approach to the recommender, our aim is to propose a system that is independent of the domain. Knowledge-based systems are hence not a good choice since they need deep knowledge specific to the domain. In Content-based systems, information about the nature of the configuration items and the user's interests are required. In the problem definition of this work, we choose to be content-agnostic and focus on the social aspect of configuration. As such we propose a type of Collaborative Filtering system.

In the area of collaborative filtering systems, neighbourhood models are a popular choice, which can be thought as potentially applicable to software configuration. Nonetheless, Neighbourhood models need a measure of user similarity. Normally in Neighbourhood models the history of the transaction of each user with the system is used as an indication of his/her interests. In the problem at hand, in order to propose a generic framework, we assume no such information about users is available. In the absence of any information to serve as a similarity measure between users, a model is required to be able to infer the preferences based on minimum input from users. The literature does not seem to investigate how finding this minimum is possible in a way that fits to our problem.

In the proposed solution instead, users are required to answer a smaller subset of the configuration items; the explicit choice of the user would then be indicative of the user preference and can be matched against the crowd. This way through complex probabilistic models (i.e. MDPs), the model-based recommender will be able to predict as many items as possible based on minimum input from users.

6.2.3 Configuration Dialogues

In our work, the problem of customization of configuration items is considered as a sequential decision making process through a conversational process with the user. How can such a conversational processes be implemented in practice; i.e. how would the interface look like? While we prefer to leave interface design issues outside the scope of this thesis it is easy to see some possibilities. The concept of the "wizard", for example, is commonplace nowadays for facilitating the configuration of software systems: in a sequence of screens the user is guided from one variable to the other so that a baseline of options is elicited from her. Although such wizards are used widely for a variety of tasks (installations, configurations, etc.), their design seems to be based on developer intuition and domain expertise. The optimal policy that is generated in our technique could instead offer a hint as to how the variables should be ordered, helping developers come up with reasonable wizard designs. A systematic and generic way for designing wizards could not be found in the literature and clearly more research needs to be done on conceptualizing, organizing (in e.g. hierarchies) and communicating configuration options.

6.2.4 Why MDPs

In deciding the best sequence of configuration questions to ask, MDPs are very appropriate for a number of reasons. MDPs not only help find the shortest path (of questions to ask user), but also incorporate the probabilities and benefits of taking actions into consideration. Most importantly, in MDPs the long-term effects of each recommendation, and not only the immediate reward will be taken into account. The optimal policy for instance, might recommend to enquire the user about a configuration variable, which compared to another variable, will apply to fewer association rules and hence have lower immediate predictions of unasked variables, but leads to more likely or more predicted variables in subsequent steps. As we discussed above, techniques based on heuristics which take into account either only probability or some form of utility may be easier to conceptualize but may not always give the global optimum.

Furthermore, our proposed MDP model could easily incorporate information about users such as age and gender to take advantage of any patterns of preference in customization of software. In the Facebook case study for instance, biographic information such as the location, gender and age are reported to influence the trends of usage [76, 77] and hence are suspected to affect the customization of the settings. If any of these variables appear in the association rules, they can be added as extra variables in the definition of states without any need for addition of the unknown value as in the system variables. Neither do we add these in the actions variables, since for any user this information is readily available. In this research since we aimed at proposing a generic work that can be applied in any application, we restricted ourselves to the system variables only.

6.3 Challenges, Limitations and Opportunities

6.3.1 Dealing with rare requirements.

One possible drawback to collaborative systems is the potential bias of results towards the popular items and inability to serve users with rare requirements [18]. We believe mining of interesting rare patterns as well as frequent patterns is the solution that can rectify this challenge. A rare pattern is an infrequent pattern with a support that is below (or far below) minimum threshold. The high confidence, yet low support of such patterns may highlight exceptional behavior in the data. Literature are galore that discuss the various measures (e.g. *All_confidence*, *Max_confidence,* and *Cosine* [7]) that can be used for mining interesting patterns. Using such measures, we can augment our support–confidence framework to allow discovery of strong pattern relationships regardless of the frequency.

Without thorough mining of all interesting patterns of an expert crowd with diverse requirements, the results of our method would be a collaborative recommender based on the social trails associated with the configuration: the user will be advised of the attributes or complete configurations that are common. Provided further investigation is considered to generate association rules of rare patterns that would embrace rare tastes as well, they will be incorporated in the MDP model not any different than the popular rules. This way our recommender is not at disadvantage from the point of view of serving different tastes.

6.3.2 Evaluation Challenges

The evaluation of the recommendation systems is not an easy task. Most recommender systems are tested on historical data for accuracy measurement: The recommender is trained using historical data, and tested to see whether the recommendations accurately predict the user's choices [37, 18]. However in our work, the accuracy depends on the degree by which each configuration in the data set is believed to accurately predict the true configuration desire of the user and the true fit to user requirements. Hence in training the recommender model, the selection of the expert crowd in whose customizations we have confidence is crucial.

Beside the challenge in truly measuring the accuracy, this measure is insufficient to evaluate a good recommendation engine since users expect more of a recommendation system than just an exact anticipation of their tastes [78]. In a handbook on evaluation of recommendation systems, Shani and Gunawardana have explored a range of properties, other than accuracy, that are decisive in success of an RS; instances of such measures include: cold start, confidence, trust of users, novelty of the recommendations, and utility of recommendations to users or system. They have also emphasized that the subset of properties on which the system should be assessed depend on the context and usage of the RS and differs from system to system [78]. In the domain of configuration of software in particular, our goal is to increase user awareness of the solutions/options which might have been previously unknown. As such, similar to the work of Shani et al. [37], we believe that the use of measures such as satisfaction or utility of user upon receiving the recommendations is promising.

As the initial study to introduce a generic framework for configuration customization, in this thesis we conducted an offline experiment to evaluate prediction accuracy only. However more direct user studies can be performed in the future. In such studies, small group of users could go through the MDP-based recommender dialogues that are generated based on configurations of validated expert crowd. Upon receiving the customizations, users can be enquired how they like the new configuration and how the new customization has influenced their experience. We believe a true indicator of success of recommendations in the context of configuration customization would be how the recommendations are accepted by users and facilitates user interaction with the system.

6.3.3 Performance

While MDPs provide us with an optimal answer, they are known to be of high complexity due to the nature of the computation they perform [26, 75]. It is, first, noteworthy that the MDP solving is performed at design-time and offline, which means that longer computation times are affordable to some extent. The performance experimentation shows that configuration problems of up to the equivalent of fourteen (14) binary variables can be computed within practical time limits. While this is still useful – e.g. it gave us an answer to the Facebook privacy configuration problem – it is fair to desire solutions to problems with more variables.

To improve the scalability of the recommender model for larger problems, a departure from MDPs towards a simplification that is less computationally hard could help improve performance. For example, algorithms that compute rankings of variables based on the degree by which they influence other variables can be explored. As we discussed, simple heuristic-based approaches like this, although scalable and definitely a subject for future experimental research, do not guarantee optimality; such an output would disregard the probabilistic aspect and would be insensitive to user answers compared to MDP-based advanced adaptive conversational recommender.

Regarding the improvement in MDP model, since the main challenge seems to be unmanageable growth of the model, measures should be taken in order to control state space. One possible solution is reducing the space size by exclusion of states whose probability of occurrence is very low. In the explained process in Chapter 3, we consider all the possible combinations of the configuration variables as a state of the MDP model. However not all combinations are likely to happen. In our email client example, it would be rare to see one using very large font size and very small icon sizes. Such approximation

[37] can be considered to include only states that were observed in the crowd with a certain threshold. In other words, we are ignoring the states whose probability of incoming edges are low.

Another solution to handle larger number of variables in MDPs modelling is breaking a large configuration problem into smaller ones based on configuration aspects, i.e. subsets of configuration variables that are conceptually coherent. The privacy configurations of Facebook offers a good example of such an aspect; it was indeed fairly easy for us to separate it from other configuration variables of Facebook. This way we also achieve another improvement in the process of eliciting user responses: in the current framework the sequence of actions to be asked from user is only decided by the solution of MDP. Provided the configuration items entail a variety of areas, the questions posed to the user may be arbitrary, jumping from one subject to another. Dividing the configuration items to groups that are close in topic, would prevent such irregular switch in the topics of questions. More application attempts to other systems will hopefully reveal whether such division into sub-problems has a practical merit.

6.3.4 The role of the configuration items

Moving on to another potential research topic, it would be interesting to explore alternatives to avoid posing harder questions to users and to emphasize on the customization of relatively more important items. Right now the importance of the settings, or the difficulty they pose to the user are not taken into account and all configuration items are treated the same. It would be desirable if user's difficulty of the handling the configuration plays a role in the sequence; it would be more desirable if the recommender can predict the rather difficult options than the easier one. This can perhaps be applied by allocating heavier cost to questions that are deemed difficult for users and granting more rewards to states in which the important configuration items are figured out.

As a final note, the proposed framework is concerned with configuration variables that are finite (actually: of a small domain size) and discrete. Continuous variables can be discretized to fit to this model. In that case, automatic configuration of a continuous variable would imply setting a characteristic parameter of the interval that resulted from discretization – so while values of the font size interval [15pt, 25pt] map to discrete value *large*, when we, conversely, need to automatically set the discrete value *large* to the continuous variable font size, e.g. the average of the interval (20pt) can be set. Whether such discretization techniques are adequate, or whether continuous optimization techniques need to be introduced, is a matter of empirical investigation with different kinds of systems and configuration aspects and problems.

References

[1] Hui, Bowen, Sotirios Liaskos, and John Mylopoulos. "Requirements analysis for customizable software: A goals-skills-preferences framework." *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pp. 117-126. IEEE, 2003.

[2] Liaskos, Sotirios, Alexei Lapouchnian, Yiqiao Wang, Yijun Yu, and Steve Easterbrook. "Configuring common personal software: a requirements-driven approach." In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on Requirements Engineering (RE'05),* pp. 9-18. IEEE, 2005.

[3] Hargittai, Eszter. "Facebook privacy settings: Who cares?" *First Monday* 15, no. 8 (2010).

[4] Hargittai, Eszter. "Digital na (t) ives? Variation in internet skills and uses among members of the "Net generation"*." *Sociological Inquiry* 80, no. 1 (2010): 92-113.

[5] Mackay, Wendy E. "Triggers and barriers to customizing software." In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 153-160. ACM, 1991.

[6] Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." In *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487-499. 1994.

[7] Jiawei Han, Micheline Kamber, Jian Pei. *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann. 2011.

[8] Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." In *ACM SIGMOD Record*, vol. 22, no. 2, pp. 207-216. ACM, 1993.

[9] Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." In *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487-499. 1994.

[10] Hipp, Jochen, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. "Algorithms for association rule mining—a general survey and comparison." *ACM SIGKDD Explorations Newsletter* 2, no. 1 (2000): 58-64.

[11] Zaki, Mohammed Javeed. "Scalable algorithms for association mining." *Knowledge and Data Engineering, IEEE Transactions on Knowledge and Data Mining,* vol 12, no. 3 (2000): 372-390.

[12] Boutilier, Craig, Thomas Dean, and Steve Hanks. "Decision-theoretic planning: Structural assumptions and computational leverage." *arXiv preprint arXiv*:1105.5460 (2011).

[13] White, Douglas J. "Real applications of Markov decision processes." *Interfaces15*, no. 6 (1985): 73-83.

[14] White, Douglas J. "A survey of applications of Markov decision processes." *Journal of the Operational Research Society* (1993): 1073-1096.

[15] Altman, Eitan. "Applications of Markov decision processes in communication networks." In *Handbook of Markov decision processes*, pp. 489-536. Springer US, 2002.

[16] Choudhary, Alok K., Jennifer A. Harding, and Manoj K. Tiwari. "Data mining in manufacturing: a review based on the kind of knowledge." *Journal of Intelligent Manufacturing* 20, no. 5 (2009): 501-521.

[17] Debatin, Bernhard, Jennette P. Lovejoy, Ann-Kathrin Horn, and Brittany N. Hughes. "Facebook and online privacy: Attitudes, behaviors, and unintended consequences." *Journal of Computer-Mediated Communication* 15, no. 1 (2009): 83-108.

[18] Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook." In *Recommender Systems Handbook*, pp. 1-35. Springer US, 2011.

[19] Desrosiers, Christian, and George Karypis. "A comprehensive survey of neighborhood-based recommendation methods." In *Recommender systems handbook*, pp. 107-144. Springer US, 2011.

[20] Felfernig, Alexander, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker. "Developing constraint-based recommenders." In *Recommender systems handbook*, pp.

[21] Romero-Mariona, Jose, Hadar Ziv, and Debra J. Richardson. "SRRS: a recommendation system for security requirements." In *Proceedings of the 2008 international workshop on Recommendation systems for software engineering*, pp. 50-52. ACM, 2008.

[22] Dumitru, Horatiu, Marek Gibiec, Negar Hariri, Jane Cleland-Huang, Bamshad Mobasher, Carlos Castro-Herrera, and Mehdi Mirakhorli. "On-demand feature recommendations derived from mining public product descriptions." In *Software Engineering (ICSE), 2011 33rd International Conference*, pp. 181-190. IEEE, 2011.

[23] Schafer, J. Ben, Dan Frankowski, Jon Herlocker, and Shilad Sen. "Collaborative filtering recommender systems." In *The adaptive web*, pp. 291-324. Springer Berlin Heidelberg, 2007.

[24] Tiihonen, Juha, and Alexander Felfernig. "Towards recommending configurable offerings." *International Journal of Mass Customisation* 3, no. 4 (2010): 389-406.

[25] Koren, Yehuda, and Robert Bell. "Advances in collaborative filtering." In *Recommender Systems Handbook*, pp. 145-186. Springer US, 2011.

[26] Su, Xiaoyuan, and Taghi M. Khoshgoftaar. "A survey of collaborative filtering techniques." *Advances in artificial intelligence* 2009 (2009): 4.

[27] Chien, Yung-Hsin, and Edward I. George. "A bayesian model for collaborative filtering." In *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics. San Francisco: Morgan Kaufman Publishers*,[http://uncertainty99. microsoft.com/proceedings. htm], 1999.

[28] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal of machine Learning research* 3 (2003): 993-1022.

[29] Breese, John S., David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43-52. Morgan Kaufmann Publishers Inc., 1998.

[30] Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual Web search engine." *Computer networks and ISDN systems* 30, no. 1 (1998): 107-117.

[31] Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. "Item-based collaborative filtering recommendation algorithms." In *Proceedings of the 10th international conference on World Wide Web*, pp. 285-295. ACM, 2001.

[32] Buckley, Chris, and Gerard Salton. "Optimization of relevance feedback weights." In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 351-357. ACM, 1995.

[33] Cohen, William W., Robert E. Schapire, and Yoram Singer. "Learning to order things." *arXiv preprint arXiv*:1105.5464 (2011).

[34] Fu, Xiaobin, Jay Budzik, and Kristian J. Hammond. "Mining navigation history for recommendation." In *Proceedings of the 5th international conference on Intelligent user interfaces*, pp. 106-112. ACM, 2000.

[35] Leung, Cane Wing-ki, Stephen Chi-fai Chan, and Fu-lai Chung. "A collaborative filtering framework based on fuzzy association rules and multiple-level similarity." *Knowledge and Information Systems* 10, no. 3 (2006): 357-381.

[36] Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." *Knowledge and Data Engineering, IEEE Transactions* 17, no. 6 (2005): 734-749.

[37] Shani, Guy, Ronen I. Brafman, and David Heckerman. "An MDP-based recommender system." In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pp. 453-460. Morgan Kaufmann Publishers Inc., 2002.

[38] Hauskrecht, Milos. "Incremental methods for computing bounds in partially observable Markov decision processes." In *AAAI/IAAI*, pp. 734-739. 1997.

[39] Poupart, Pascal, and Craig Boutilier. "VDCBPI: an Approximate Scalable Algorithm for Large POMDPs." In *NIPS*. 2004.

[40] Kearns, Michael, Yishay Mansour, and Andrew Y. Ng. "A sparse sampling algorithm for near-optimal planning in large Markov decision processes." *Machine Learning* 49, no. 2-3 (2002): 193-208.

[41] Carenini, Giuseppe, Jocelyin Smith, and David Poole. "Towards more conversational and collaborative recommender systems." In *Proceedings of the 8th international conference on Intelligent user interfaces*, pp. 12-18. ACM, 2003.

[42] Mahmood, Tariq, and Francesco Ricci. "Improving recommender systems with adaptive conversational strategies." In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pp. 73-82. ACM, 2009.

[43] Häubl, Gerald, and Kyle B. Murray. "Preference construction and persistence in digital marketplaces: The role of electronic recommendation agents." *Journal of Consumer Psychology* 13, no. 1 (2003): 75-91.

[44] Hansen, Torben, Christian Scheer, Johannes Gutenberg, and Peter Loos. "Product Configurators in Electronic Commerce–Extension of the Configurator Concept towards Customer Recommendation." In *Proceedings of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization (MCP)*. 2003.

[45] Cöster, Rickard, Andreas Gustavsson, Tomas Olsson, and Åsa Rudström. "Enhancing web-based configuration with recommendations and cluster-based help." In *Proceedings of the AH'2002 Workshop on Recommendation and Personalization in e-Commerce*. 2002.

[46] Felfernig, Alexander, Monika Mandl, Juha Tiihonen, Monika Schubert, and Gerhard Leitner. "Personalized user interfaces for product configuration." In *Proceedings of the 15th international conference on Intelligent user interfaces*, pp. 317-320. ACM, 2010.

[47] Mittal, Sanjay, and Felix Frayman. "Towards a Generic Model of Configuration Tasks." In I*JCAI*, vol. 89, pp. 1395-1401. 1989.

[48] Fleischanderl, Gerhard, Gerhard E. Friedrich, Alois Haselböck, Herwig Schreiner, and Markus Stumptner. "Configuring large systems using generative constraint satisfaction." *IEEE Intelligent Systems* 13, no. 4 (1998): 59-68.

[49] Sabin, Daniel, and Rainer Weigel. "Product configuration frameworks-a survey." *IEEE intelligent systems* 13, no. 4 (1998): 42-49.

[50] Stumptner, Markus. "An overview of knowledge-based configuration." *AI Communications* 10, no. 2 (1997): 111-125.

[51] Maalej, Walid, and Anil Kumar Thurimella. "Towards a research agenda for recommendation systems in requirements engineering." In *Managing Requirements Knowledge (MARK)*, 2009 Second International Workshop on, pp. 32-39. IEEE, 2009.

[52] Castro-Herrera, Carlos, Jane Cleland-Huang, and Bamshad Mobasher. "Enhancing stakeholder profiles to improve recommendations in online requirements elicitation." In *Requirements Engineering Conference*, 2009. RE'09. 17th IEEE International, pp. 37-46. IEEE, 2009.

[53] Lim, Soo Ling, and Anthony Finkelstein. "StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation." *Software Engineering, IEEE Transactions on Software Engineering* 38, no. 3 (2012): 707-735.

[54] Lim, Soo Ling, Daniele Quercia, and Anthony Finkelstein. "StakeSource: harnessing the power of crowdsourcing and social networks in stakeholder

analysis." In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, vol 2, pp. 239-242. ACM, 2010.

[55] Duan, Chuan, Paula Laurent, Jane Cleland-Huang, and Charles Kwiatkowski. "Towards automated requirements prioritization and triage." *Requirements Engineering* 14, no. 2 (2009): 73-89.

[56] Felfernig, A., G. Ninaus, H. Grabner, F. Reinfrank, L. Weninger, D. Pagano, and W. Maalej. "An Overview of Recommender Systems in Requirements Engineering (preprint version)."

[57] Mobasher, Bamshad, and Jane Cleland-Huang. "Recommender systems in requirements engineering." *AI Magazine* 32, no. 3 (2011): 81-89.

[58] Consens, Mariano P., Denilson Barbosa, Adrian Teisanu, and Laurent Mignet. "Goals and benchmarks for autonomic configuration recommenders." In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 239-250. ACM, 2005.

[59] Page, Stanley R., Todd J. Johnsgard, Uhl Albert, and C. Dennis Allen. "User customization of a word processor." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 340-346. ACM, 1996.

[60] Dardenne, Anne, Axel Van Lamsweerde, and Stephen Fickas. "Goal-directed requirements acquisition." *Science of computer programming* 20, no. 1 (1993): 3-50.

[61] Mylopoulos, John, Lawrence Chung, Stephen Liao, Huaiqing Wang, and Eric Yu. "Exploring alternatives during requirements analysis." *Software, IEEE* 18, no. 1 (2001): 92-96.

[62] Liaskos, Sotirios, Shakil M. Khan, Marin Litoiu, Marina Daoud Jungblut, Vyacheslav Rogozhkin, and John Mylopoulos. "Behavioral adaptation of information systems through goal models." *Information Systems* 37, no. 8 (2012): 767-783.

[63] Liaskos, Sotirios, Sheila A. McIlraith, Shirin Sohrabi, and John Mylopoulos. "Representing and reasoning about preferences in requirements engineering." *Requirements Engineering* 16, no. 3 (2011): 227-249.

[64] Gross, Ralph, and Alessandro Acquisti. "Information revelation and privacy in online social networks." In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pp. 71-80. ACM, 2005.

[65] Johnson, Maritza, Serge Egelman, and Steven M. Bellovin. "Facebook and privacy: it's complicated." In *Proceedings of the eighth symposium on usable privacy and security*, p. 9. ACM, 2012.

[66] Lampe, Cliff, Nicole B. Ellison, and Charles Steinfield. "Changes in use and perception of Facebook." In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pp. 721-730. ACM, 2008.

[67] Fang, Lujun, and Kristen LeFevre. "Privacy wizards for social networking sites." In *Proceedings of the 19th international conference on World wide web*, pp. 351-360. ACM, 2010.

[68] Lipford, Heather Richter, Andrew Besmer, and Jason Watson. "Understanding Privacy Settings in Facebook with an Audience View." *UPSEC* 8 (2008): 1-8.

[69] Strater, Katherine, and Heather Richter Lipford. "Strategies and struggles with privacy in an online social networking community." In *Proceedings of the 22nd*

British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction, vol 1, pp. 111-119. British Computer Society, 2008.

[70] Church, Luke, Jonathan Anderson, Joseph Bonneau, and Frank Stajano. "Privacy stories: confidence in privacy behaviors through end user programming." In *SOUPS*. 2009.

[71] Acquisti, Alessandro, and Ralph Gross. "Imagined communities: Awareness, information sharing, and privacy on the Facebook." In *Privacy enhancing technologies*, pp. 36-58. Springer Berlin Heidelberg, 2006.

[72] Liu, Yabing, Krishna P. Gummadi, Balachander Krishnamurthy, and Alan Mislove. "Analyzing Facebook privacy settings: user expectations vs. reality." In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 61-70. ACM, 2011.

[73] Jones, Simon, and Eamonn O'Neill. "Feasibility of structural network clustering for group-based privacy control in social networks." In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, p. 9. ACM, 2010.

[74] McNee, Sean M., John Riedl, and Joseph A. Konstan. "Being accurate is not enough: how accuracy metrics have hurt recommender systems." In *CHI'06 extended abstracts on Human factors in computing systems*, pp. 1097-1101. ACM, 2006.

[75] White III, Chelsea C. "A survey of solution techniques for the partially observed Markov decision process." *Annals of Operations Research* 32, no. 1 (1991): 215-230.

[76] Joinson, Adam N. "Looking at, looking up or keeping up with people?: motives and use of Facebook." In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 1027-1036. ACM, 2008.

[77] Pempek, Tiffany A., Yevdokiya A. Yermolayeva, and Sandra L. Calvert. "College students' social networking experiences on Facebook." *Journal of Applied Developmental Psychology* 30, no. 3 (2009): 227-238.

[78] Shani, Guy, and Asela Gunawardana. "Evaluating recommendation systems." In *Recommender systems handbook*, pp. 257-297. Springer US, 2011