Chapter 17
Recommendation Systems in Requirements Discovery

Negar Hariri, Carlos Castro-Herrera, Jane Cleland-Huang, and Bamshad Mobasher

Abstract Recommendation systems offer the opportunity for supporting and enhancing a wide variety of activities in requirements engineering. We discuss several potential uses. In particular we highlight the role of recommendation systems in online forums that are used for capturing and discussing feature requests. The recommendation system is used to mitigate problems introduced when face-to-face communication is replaced with potentially high-volume online discussions. In this context, recommendation systems can be used to suggest relevant topics to stakeholders and conversely to recommend expert stakeholders for each discussion topic. We also explore the use of recommendation systems in the domain analysis process, where they can be used to recommend sets of features to include in new products.

17.1 Introduction

Requirements engineering covers a variety of different activities focused on the discovery, analysis, specification, validation, and management of software and systems requirements [30, 42, 45, 51]. The primary goal of the discovery process is to elicit and identify stakeholders' needs, wants, and desires for the software system. This can be somewhat challenging, especially when stakeholders are geographically distributed and unable to physically gather together for face-to-face meetings. Different groups of stakeholders also have differing perspectives and goals for the system, which can create conflicts and inconsistencies. This is particularly

N. Hariri (✉) • J. Cleland-Huang • B. Mobasher
School of Computing, DePaul University, Chicago, IL, USA
e-mail: nhariri@cs.depaul.edu; jhuang@cs.depaul.edu; mobasher@cs.depaul.edu

C. Castro-Herrera
GOOGLE, Chicago, IL, USA
e-mail: ccastro@google.com

M.P. Robillard et al. (eds.), *Recommendation Systems in Software Engineering*,
DOI 10.1007/978-3-642-45135-5_17, © Springer-Verlag Berlin Heidelberg 2014

troublesome if important stakeholders are missing from the requirements elicitation and negotiation process [18].

Robertson and Robertson [45] prescribe a rigorous upfront domain analysis and requirements trawling process which involve identifying and engaging stakeholders, observing users performing tasks in their natural work environments, and conducting interviews, surveys, and group brainstorming meetings. These activities are designed to discover, analyze, and prioritize requirements, to specify use-cases and business rules, and in some cases to propose and evaluate candidate design solutions. All of these activities are highly collaborative and people-intensive.

Several recent trends have significantly impacted the way we think about requirements. The move towards the globalization of software development and the dispersion of stakeholders across multiple geographical locations [13] makes communication and coordination more difficult and introduces challenges caused by diversity in language and culture, lack of engagement in the requirements discovery process, loss of informal communication between stakeholders, a reduced level of trust caused by the lack of face-to-face communication, difficulties in managing conflicts and achieving a common understanding of the requirements, ineffective decision-making meetings, and process delay introduced by the time zone differences [17].

The popularity of open source software development has also affected the requirements process. The collaborative and transparent nature of open source projects has popularized the notion of opening up the requirements elicitation process to allow a far broader set of stakeholders to contribute their ideas and suggestions using an online forum [48]. The impact has been felt even in more traditional projects.

Finally, the broad adoption of agile approaches has impacted the way in which we define requirements. As a community, we now embrace the idea that software requirements may emerge incrementally as the project progresses. This is particularly true in software-intensive projects as opposed to more traditional systems engineering projects. In this chapter we therefore focus more on the ongoing discovery of ideas and features as opposed to the specification of more traditional requirements.

Recommendation systems can potentially address many of the challenges involved in the elicitation process. In general, a recommendation system [1] identifies items of potential interest to a given user based on that user's preference profile (see Ying and Robillard [53] in Chap. 8 for more details on user profiles) or observed behavior. It is not difficult to conceive of stakeholders or even products as the target of recommendations, and users, topics, or features as the recommendable items.

In this chapter we provide an overview of several areas of the requirements process which could potentially benefit from the use of recommendation systems. We then describe two diverse applications of recommendation in greater detail. The first uses a recommendation system to support requirements discovery in online discussion forums by helping to manage and organize stakeholders' discussions, recommending discussion threads to stakeholders, and recommending

knowledgeable stakeholders for specific topics. The second application leverages the availability of detailed product information on publicly accessible websites such as Softpedia and then uses this data to learn association rules and to construct a recommendation system capable of recommending domain-specific features for a product. Both applications leverage growing trends towards moving the requirements process online through adopting social networking tools.

17.2 Recommendation Systems in Requirements Engineering

While a significant body of prior work has focused on making recommendations to support more general software engineering tasks such as finding experts to help with development tasks [37, 39, 40], keeping developers informed of stakeholders working on related tasks [52], or supporting the build process [49], there has been far less thought on how to utilize recommendation systems within the requirements discovery process. Felfernig et al. [24] presented a visionary perspective of a "Recommendation and Decision Support System," which would support individual and group activities through recommending stakeholders for quality reviews, prioritizing requirements, suggesting relevant requirements for a current task, identifying dependencies among requirements, proposing changes that could be made to a requirements artifact to maximize group agreement, and identifying sets of requirements for a future release. In other words, they envisioned a system that could assist in a wide array of tasks related to requirements engineering.

Similarly, Maalej and Thurimella [34] proposed a research agenda for recommendation systems in requirements engineering. They envisioned potential uses of recommendation systems which included recommending traceability links, relevant background information, artifacts that have changed, templates to use, past rationale decisions, requirements from previous systems, vocabulary to use, people to collaborate with, status of activities and artifacts, and priorities, among others.

In this chapter we focus on recommendation systems which have been actually implemented and evaluated in the requirements engineering domain. We avoid discussing systems which have the look-and-feel of a recommendation system, but which are purely search based and therefore do not leverage the core concepts that define a recommendation system. For example, in the requirements engineering field, researchers have developed techniques for generating (or retrieving) trace links between various artifacts, such as between requirements and source code, or between requirements and regulatory codes [3, 15, 20, 29]. However, while the end result is a ranked listing of candidate links, which may appear to take on the form of a recommendation, to the large part these approaches leverage basic information retrieval and machine learning techniques instead of the core recommendation algorithms that are the focus of this book.

Another interesting application of a recommendation system was proposed by Lim et al. [32, 33]. They developed a tool called StakeNet, which used social networking techniques to generate recommendations of project stakeholders. Based

upon an early definition of the project, they identified an initial set of stakeholder roles and associated stakeholders. They then utilized their tool to invite each of the identified stakeholders to recommend additional stakeholders and to provide a salience measure that captures the influence, legitimacy, and urgency of the recommendation. Finally they used social networking metrics to prioritize candidate stakeholders for inclusion in the project. StakeNet is unique in the way it generates recommendations. Unlike other systems, StakeNet elicits recommendations directly from users and then filters them using social network metrics. It is these filtered recommendations which are presented to the users.

17.3 Recommendation Systems in Online Forums

Wikis and forums provide community-based portals that support collaborative tasks and knowledge management activities. There are many benefits in using online forums to support requirements discovery. For example, forums create a broadly inclusive environment in which geographically distributed stakeholders can collaborate asynchronously in a virtual meeting place to explore, discuss, and specify requirements [21,47,48]. Noll [41] observed that almost all the requirements for Firefox 2.0 were discovered through online forums, wikis, and bug tracking systems. Christley and Madey [14] also pointed out that many activities that take place in open source development are supported by online forums. Laurent and Cleland-Huang [31] explored the way vendor-led open source projects conducted requirements engineering tasks using online forums. They found forums to be very effective for including large numbers of stakeholders; however, they also found that the sheer mass of data collected in the forums created a number of challenges. For example, it was often difficult for new users to find relevant discussion threads. Similarly, project managers found it difficult to extract and manage feature requests from within the forums, in order to identify specific stakeholder roles, and to understand and document feature priorities.

To better understand the challenges of using social networking tools for requirements elicitation, we analyzed discussion threads and topics in the forums of several open source projects and found a high percentage of discussion threads consisting of only one or two feature requests. For example, as shown in Fig. 17.1, 59 % of Poseidon threads, 48 % of Open Bravo threads, and 42 % of Zimbra threads included only one or two posts [16]. The presence of so many small threads suggests either that a significant number of distinct discussion topics exist or that users tend to initiate redundant threads without first searching for related discussions. This phenomenon hinders the overarching goal of the forum, which is to emerge project requirements by facilitating topic-based discussions between stakeholders with similar interests. Without some sort of structuring and support, online forums tend to degenerate into question and answer style venues or, even worse, to contain large numbers of posts which lack any response or related discussion. These

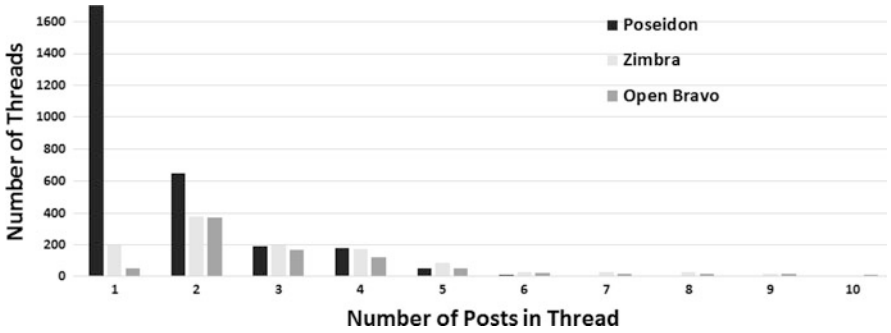

Fig. 17.1 Discussion thread sizes. Forums are characterized by numerous small threads and a couple of very large threads

problems create a rich opportunity for utilizing recommendation systems to improve stakeholder collaboration.

The idea of utilizing recommendation systems in online forums is certainly not new. Spertus et al. [50] developed a system for recommending user-created discussion groups in the Orkut social networking site. They used a collaborative filtering approach based on the *k*-nearest neighbors (kNN) strategy and compared multiple similarity metrics. Chen et al. [12] also recommended communities in the Orkut social networking site, but their approach differed from Spertus et al. in that they used multiple input sources: users' community memberships and users' textual contributions. Freyne et al. [26] explored the effect that generating early personalized recommendations had on social networking sites. In their work they generated two kinds of recommendations: recommending people to be added to a social network and recommending enhancements to a person's profile. They found that the users who received early recommendations became more engaged in the social network. Guy et al. [27] explored the recommendation of "social software items" within a social networking site. The recommended items included webpages, blog entries, wiki pages, and user communities. They experimented on moving beyond the concept of *user similarity*, to the idea of *user familiarity*, where neighborhoods were created using the user's social network. They discovered that familiarity worked better, and that it provided richer explanations for the recommendations.

17.3.1 Recommending Topics

Recommendation systems can be used to help manage the requirements process in online forums. For example, they can be used to address the problem in which stakeholders with similar interests fail to "find each other" in the forum, meaning that topic discussions are dispersed across multiple threads, preventing full and

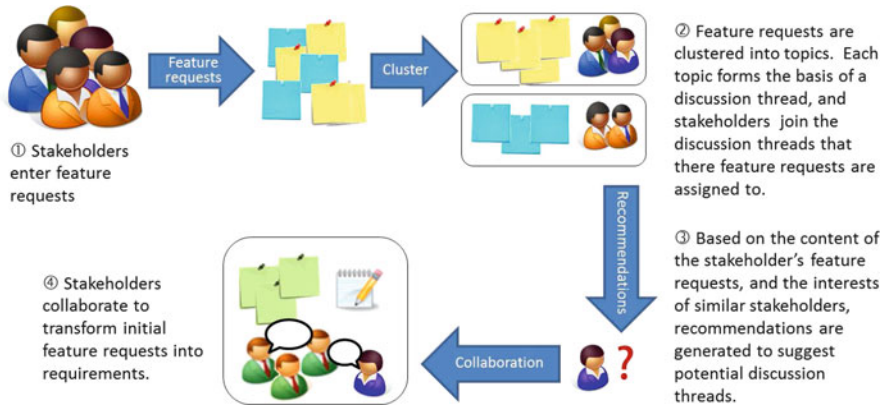

Fig. 17.2 OPCI: Organizer and promoter of collaborative ideas

rigorous exploration of specific issues. In addition, recommendation systems can be used to address the problem of unexplored topics by recommending expert stakeholders for a topic.

As an example, consider the recommendation system OPCI (Organizer and Promoter of Collaborative Ideas) [5, 7–10]. OPCI provides support for recommending stakeholders and topics and optional support for managing the actual discussion threads. If thread management is used, then OPCI proactively helps to maintain a more cohesive set of discussion topics. New posts are analyzed in order to determine if one or more existing and relevant discussion threads already exist. If so, OPCI presents these threads to the user so that they have the option of posting their feature request or comment to one of the existing threads. Furthermore, OPCI also monitors discussions in existing threads, determines when the discussion diverges into a new topic, and makes appropriate suggestions for spawning new threads. We do not elaborate on these features of OPCI further, as they are not central to the application of recommendation systems in a requirements forum. The recommendation systems described in the remainder of this section can be applied to either user-defined or automatically clustered discussion threads.

It is interesting to note that a discussion topic represents both a clustering of feature requests and a grouping of stakeholders. In the remainder of this chapter we refer to a *topic* and *discussion thread* synonymously. The general process is summarized in Fig. 17.2 which shows that OPCI assumes stakeholders' needs are collected in a web-based tool such as a wiki, forum, or bug tracker. The feature requests are then placed into discussion threads either by the users themselves or by a tool such as OPCI. A variety of recommendation algorithms are then used to generate recommendations to project stakeholders, thereby enhancing the quality of the online requirements discovery process.

OPCI uses both content-based and collaborative-based recommendation systems. The content-based recommendation utilizes the content of the discussion threads

to initially recommend similar topics to a stakeholder while the collaborative recommendation creates additional recommendations by identifying stakeholders with similar interests, and then using these similarities to generate recommendations. Content-based recommendation systems, which are particularly useful for keeping similar feature requests collocated in a single thread, are discussed in more detail later in the chapter. In the following section we focus on describing the use of collaborative recommendations. These serve the important role of cross-pollinating discussions with contributions from stakeholders with related concerns.

17.3.2 Creating User Profiles

A basic introduction to collaborative recommendation algorithms is provided by Menzies [36] in Chap. 3. The standard kNN-based algorithm with Pearson correlation assumes that each entry in the user profile represents the degree of interest that a user has in a particular item to be recommended.

In order to create a forum-based recommendation system, we construct a *user-by-discussion-thread* matrix R which captures the interest each user has in a particular discussion topic. There are two primary ways to represent this matrix. The first approach represents the degree of interest a user has in each topic, by depicting the number of posts a user has in a thread, and also the extent to which those posts represent core concepts of the thread, i.e., the similarity between the user's posts and the central theme of the thread. This results in a matrix R containing a set of continuous values [10].

Alternately, a binary matrix R can be used, in which a membership score of 1 means that the user has engaged in the discussion thread while a score of 0 means that they have not. However, we cannot assume that a score of 0 means that the user is not interested in the topic.

Switching to a binary representation of the R matrix requires a few additional changes to the similarity and prediction formulas of kNN, mainly because the concept of *average ratings* does not make sense in a binary profile. There are several binary similarity metrics available [50]; one of the most accepted formulas is the binary equivalent of the cosine similarity metric ($\cos'$), defined in Eq. (17.1).

$$\cos'(u_a, u_b) := \frac{|R_{u_a} \cap R_{u_b}|}{\sqrt{|R_{u_a}| \times |R_{u_b}|}}, \tag{17.1}$$

where R_u is the set of rated items of user u; more specifically, the membership (yes or no) of the user in the discussion threads.

Equation (17.2) has been shown to consistently return good recommendations.

$$\hat{r}'_{u,i} := \frac{\sum_{n \in nbr(u)} \text{userSim}(u, n) \times r_{n,i}}{\sum_{n \in nbr(u)} \text{userSim}(u, n)} \tag{17.2}$$

Table 17.1 Characteristics of the main datasets

Dataset	# Threads	# Posts	# Users
Second Life	50	3392	2120
Student	29	223	36
Sugar CRM	60	885	523
Railway	55	1652	132

17.3.3 Profile Augmentation with Requirements Metadata

In addition to using a binary profile, major improvements can also be achieved [7] by augmenting the user profiles with additional known attributes about the users. This metadata can be incorporated into the ratings matrix, such that $R = (r_{u,i})_{|U|\times(|A|+|I|)}$, where the first A columns indicate that a user has an interest in a known attribute a of the domain. This approach is feasible in the requirements engineering domain, where the role of specific stakeholders is often known or easily elicited. Examples of user attributes include the roles of the users in the project (e.g., a developer or project manager), their interest in system qualities (e.g., security or usability), or their interest in key functionalities or modules (e.g., calendaring functionality or payroll module). These additional attributes can be used to augment the user profile and to generate the neighborhoods of similar users.

17.3.4 Evaluation

One of the common ways to evaluate a recommendation system is based on the standard leave-one-out cross-validation experimental design. In this style of experiment, we systematically remove one known interest for each user and then evaluate the recommendation system's ability to successfully recommend it back. To illustrate the effectiveness of the three variants of recommendation systems discussed in previous sections, we present experimental results achieved by generating recommendations using the datasets described in Table 17.1.

"Second Life" is an Internet-based virtual world game in which users create avatars to explore, and interact in, a "virtual world." "Student" is a small dataset of feature requests and user interests created by 36 graduate level students at DePaul University for an Amazon-like student web portal where the students could buy and sell books. "Sugar CRM" is an open source customer relationship management system that supports campaign management, email marketing, lead management, marketing analysis, forecasting, quote management, case management, and many other features. Finally, "Railway" is a dataset of requirements and stakeholder roles mined from the public specifications of two large-scale railway systems, the Canadian Rail Operating Rules and the Standard Code of Operating rules published by the Association of American Railroads.

Results from four kNN-based variants are shown in the hit ratio graph depicted in Fig. 17.3 [7]. A hit ratio graph is particularly useful for evaluating a recommendation

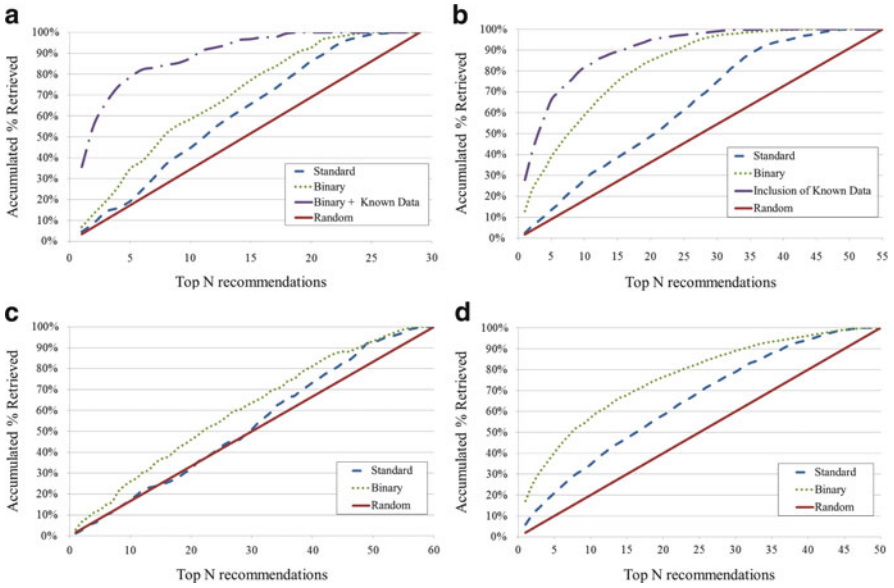

Fig. 17.3 Collaborative recommendation results using standard, binary kNN, and the augmented profiles on four datasets. (**a**) Student dataset. (**b**) Railway dataset. (**c**) Sugar dataset. (**d**) SecondLife dataset

system when the results are presented as a ranked list. Hit ratio curves plot the accumulated percentage of correctly retrieved results against the number of recommendations made. In other words, for the items that were recommended back, it shows how many were ranked as the first item shown to the user, how many were ranked as the second item, and so on. This is typically plotted in a graph and compared against a random recommendation (represented as a diagonal line). The ideal hit ratio graph for a good recommendation will show a sharp improvement over the random case for the early recommendations, indicating that those were indeed items that the user had an interest in.

In this application, binary recommendations tend to outperform the non-binary approach. Furthermore, adding additional information also significantly improves the quality of the generated recommendations, primarily because the additional knowledge increases the density of information in the users' profiles, making the selection of neighbors more reliable.

17.4 Recommending Expert Stakeholders

In addition to using recommendation systems to help stakeholders find relevant discussion threads, recommendations can be made at the project level to identify stakeholders with expertise in specific areas. These recommendations address the

commonly occurring problem of unanswered posts. In most open source projects, a significant percentage of posts never receive responses. From a requirements elicitation perspective, this suggests that certain ideas go unexplored and represents lost opportunities for truly understanding and meeting the users' needs.

The same recommendation algorithms, described in the previous sections, can be used to identify three groups of stakeholders [5, 6]:

Direct Stakeholders. Represent users who have directly contributed ideas to the topic. In other words, these are the users whose posts have been clustered together into a topic.

Indirect Stakeholders. Represent users who have contributed ideas to related topics. These stakeholders are discovered through measuring the similarity between the topics (clusters), and selecting users who have posted to closely related discussion threads.

Inferred Stakeholders. Represent users who have exhibited patterns of interest which suggest that they could potentially be interested in the topic. These users are found by a collaborative recommendation, the same binary kNN described in Eqs. (17.1) and (17.2).

While different approaches are possible, it can be particularly effective to use a hybrid recommendation system to identify and recommend expert stakeholders. In the hybrid approach, the text of the unanswered posts is first analyzed and then a content-based recommendation system is used to recommend users that contributed posts with similar content. This is achieved by clustering posts and then identifying stakeholders whose posts are placed in the same cluster (i.e., topic) as the unanswered post. The identified stakeholders are then used as the input to a collaborative recommendation algorithm so that additional users, who might be able to respond to the post, are identified. For this, the binary kNN algorithm described in Sect. 17.3.2 can be used.

The effectiveness of the hybrid recommendation system is illustrated through an experiment that simulates unanswered posts by examining each discussion thread in turn, identifying the first post of the thread, temporarily removing all other posts, and then running the hybrid recommendation to see if the authors (stakeholders) of the removed responses could be identified and recommended back.

Table 17.2 and Fig. 17.4 show the results of this experiment in terms of precision, recall, F_2 measure, and hit ratio graphs (for the collaborative part), compared to a random recommendation. There are several interesting observations. First, the content recommendation returns fairly good precision, recall, and F_2 values and clearly outperforms the random recommender. Second, the collaborative recommender outperforms the content-based recommender in terms of recall but achieves low precision. This is explained by the fact that the collaborative recommendation system outputs a much larger list of suggested users. Because a human user is not interested in so many recommendations, it is important to evaluate the ranking of the recommended users. This is depicted in the hit ratio graphs in Fig. 17.4, which show that the correct users tend to be returned before the incorrect ones. Furthermore, there is a limit to the number of users who can be identified through

Table 17.2 Performance of the recommendation of relevant users in terms of precision, recall, and F_2-measure for six open source forums

Dataset	Content Based recommendation				Collaborative Based recommendation				
	Observed		Random		Observed		Random		
	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	F_2
7-Zip	39.11%	48.14%	1.55%	1.90%	3.16%	71.77%	0.81%	18.36%	0.03
Alliance1	9.53%	27.01%	0.62%	1.77%	2.19%	42.13%	0.46%	8.78%	0.02
KeePas	23.87%	42.93%	1.20%	2.15%	3.50%	75.51%	0.69%	14.83%	0.03
MiKTe	15.60%	26.27%	1.11%	1.86%	4.15%	78.11%	0.82%	15.39%	0.03
Notepad	20.50%	37.14%	0.80%	1.45%	2.60%	70.61%	0.50%	13.68%	0.02
phpMyAd	23.15%	49.04%	0.79%	1.69%	5.69%	76.45%	0.41%	5.46%	0.02
RSS Ban	13.79%	14.29%	2.23%	2.31%	13.89%	10.42%	1.92%	1.44%	0.02

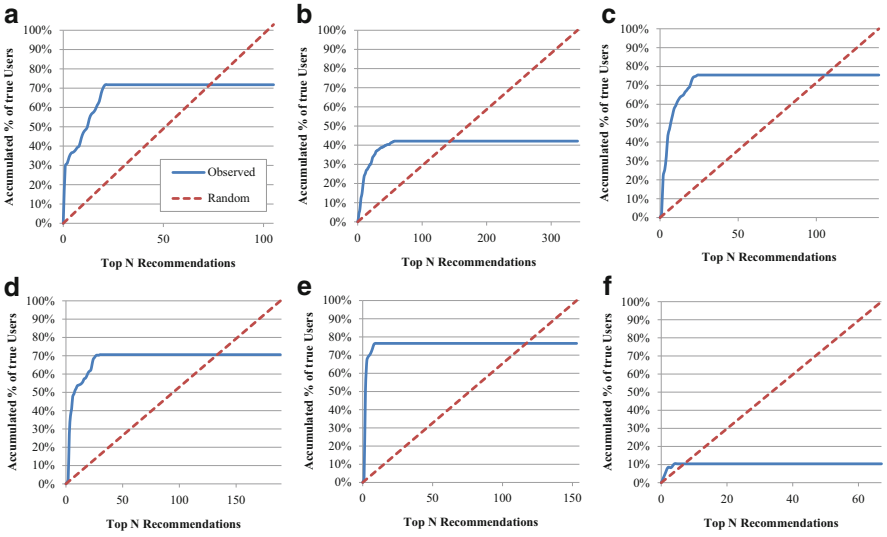

Fig. 17.4 Performance of the recommender of relevant users in terms of hit ratio for six open source forums. (**a**) 7-Zip. (**b**) Alliance. (**c**) KeyPass. (**d**) NotePad++. (**e**) PHP My Admin. (**f**) RSS Bandit

collaborative recommendations (shown as the gap between the two lines in the hit ratio graph at the maximum number of recommendations). This occurs because the collaborative recommendation was constrained to only make recommendations to users who belonged to at least three discussion threads. This restriction ensures the quality of the recommendations but also highlights the problem that we are trying to address: the fact that some users create a post that never gets answered and as a result stop participating in the open source project. A more detailed explanation of these experiments and results can be found in the related work of Castro-Herrera [5].

17.5 Feature Recommendation

The second type of recommendation system we explore in this chapter is designed for use in the domain analysis process [19]. Domain analysis is conducted in early phases of a project and involves analyzing existing software systems from

the same domain in order to better understand their common and variable parts. Domain analysis supports requirements discovery processes by identifying features commonly included in software products operating in a specific domain.

Domain analysis techniques require analysts to review documentation from existing systems in order to manually, or semi-manually, extract, organize, and model features in the domain. For example, the Domain Analysis and Reuse Environment (DARE) [25] uses semiautomated tools to extract domain vocabulary from text sources and then identifies common domain entities, functions, and objects, by clustering around related words and phrases. Chen et al. [11] manually constructed requirements relationship graphs (RRGs) from several different requirements specifications and then used clustering techniques to merge them into a single domain tree. Alves et al. [2] utilized the vector space model (VSM) and latent semantic analysis (LSA) to determine the similarity between requirements and generate an association matrix which is then clustered. A merging step is then executed to create the entire domain feature model. The primary limitations of these approaches are their reliance upon existing requirements specifications and the constraints associated with mining features from only a small handful of specifications. All of these techniques have one thing in common, which is the need for an existing set of requirements specifications. As requirements represent closely guarded intellectual property, these domain analysis techniques are often only available to organizations with existing products in the targeted domain.

On the other hand, the advent of online product repositories means that partial descriptions of hundreds of thousands of products are now available in the public domain. These product descriptions can be used in place of actual requirements to construct a recommendation system. In this section we explain how hundreds of thousands of partial product descriptions can be used to learn association rules and generate feature recommendations.

The approach utilizes data mining and machine learning methods to mine software features from online software product repositories and to infer relationships among those features. The inferred affinities are then used to train a recommender system which generates feature recommendations for a given project.

Figure 17.5 represents the overall process, consisting of an initial training phase followed by a usage phase. In the training phase, features are extracted from online product descriptions and the feature recommender is trained, while in the usage phase, the trained system makes recommendations based on an initial description of the product provided by a requirements analyst or other users of the system.

The training phase involves mining product specifications from online software product repositories. For example, feature descriptors could be retrieved from Softpedia which contains a large collection of software products. In the second step, the raw feature descriptors are fed to a clustering algorithm which groups them into features and generates an appropriate name for each feature. Finally, in the third step, a product-by-feature matrix and a feature itemset graph (FIG) based on the relationships between products and the mined features are both constructed.

The trained recommender system can be used to generate recommendations based on an initial textual description of the product provided by the requirements

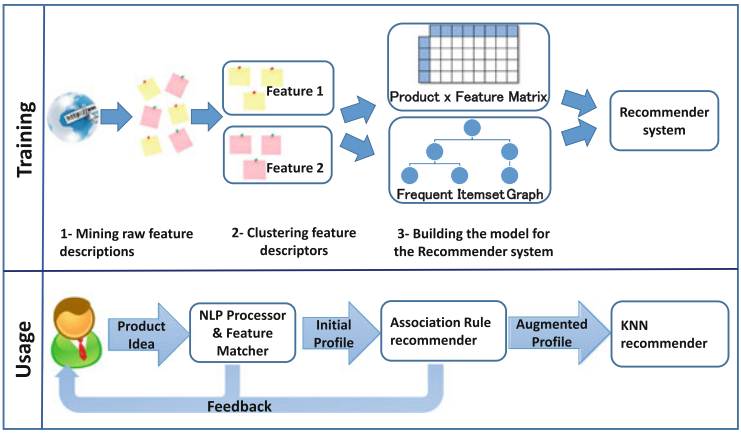

Fig. 17.5 Feature extraction and recommendation

or domain analyst. Basic information retrieval methods are used to match this description to a set of related features in the product-by-feature matrix. These features are presented to the user for confirmation. The quality of the final recommendations can be improved through utilizing a two-phase process. In the first phase, association rule mining, as discussed by Menzies [36] in Chap. 3, is used to augment the initial product profile. This is particularly effective given that we have found that association rules tend to produce relatively accurate, but incomplete recommendations. Finally, the augmented profile is used in a standard kNN approach to generate an additional set of feature recommendations. As we will later show, recommendations produced from the association rule recommender are usually complementary to the features recommended by the kNN recommender, and furthermore, augmenting the initial sparse product profile helps the kNN recommender to produce more accurate recommendations.

Figure 17.6 illustrates a feature recommendation scenario for an anti-virus product. An initial product description is first mapped to four features in the recommender's knowledge base related to *spam detection*, *disk scans*, *virus definitions*, and *virus databases*, which serve as seeds for generating feature recommendations. The product profile is then augmented by association rule mining with recommended features such as *network intrusion detection* and *real-time file monitoring*. Finally (although not shown in the figure), the kNN recommender system makes an additional set of recommendations.

17.5.1 Feature Mining

Features must initially be mined from product specifications. For the examples presented in this chapter, we utilized 117,265 different products from 21 Softpedia

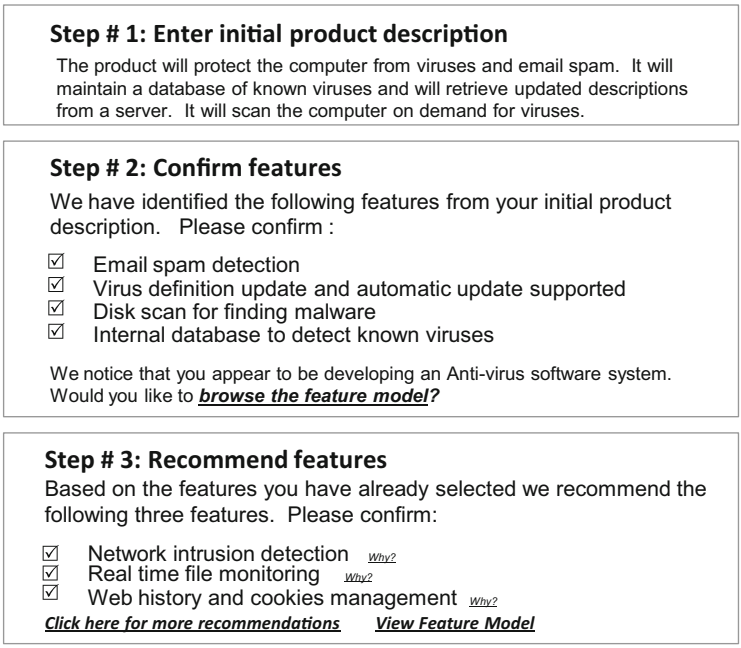

Fig. 17.6 An example usage scenario

categories. Product descriptions are parsed into sentences to form *feature descriptors* and then preprocessed using standard information retrieval techniques such as stemming and stop-word removal. Each feature descriptor is then transformed into a vector space representation using the TF–IDF approach.

As many products contain similar features, and these features are described in slightly different ways, the descriptors must be clustered into coherent clusters such that each cluster corresponds to a software feature.

The similarity of a pair of feature descriptors can be measured by computing the cosine similarity of their corresponding TF–IDF vectors. This similarity measure can be used by any conventional clustering algorithm such as K-means [35], K-medoid [35], or spherical K-means [22] to group similar feature descriptors. In our system we used the *incremental diffusive clustering* (IDC) approach [23] to group the feature descriptors into 1,135 clusters. This algorithm uses a heuristic approach to determine the number of clusters. Based on our previous studies, this algorithm tends to outperform other algorithms, including K-means, spherical K-means, and latent Dirichlet allocation (LDA) [4] for clustering requirements [28].

It is important to present a comprehensible recommendation to the user. To this end, each feature needs to be meaningfully named. One approach uses the *medoid* as the name. The medoid is defined as the descriptor that is most representative of the feature's theme. The medoid is identified by first computing the cosine similarity between each descriptor and the centroid of the cluster and then summing up the

different weighted values in the descriptor's term vector for all values above a certain threshold (0.1). Both scores are normalized and then added together for each descriptor. The descriptor scoring the highest value is selected as the feature name. This approach produces quite meaningful names. As an example, a feature based on the theme of *updat, databas, automat, viru* might subsequently be named *Virus definition update and automatic update supported*.

17.5.2 Feature Recommendation Algorithm

The goal of the feature recommendation module is to provide recommendations for a project with a given set of initial features. The feature recommender can be trained by creating a binary product-by-feature matrix, $M := (m_{i,j})_{P \times F}$, where P represents the number of products (117,265), F is the number of identified features (1,135), and $m_{i,j}$ is 1 if and only if the feature j includes a descriptor originally mined from the product i. Having this matrix, various collaborative filtering methods, including neighborhood-based techniques such as user-based kNN and item-based kNN as well as matrix factorization approaches such as BPRMF [43], can be exploited to produce recommendations. For a new product p with a set of features F_p, the recommendation algorithm computes a recommendation score for each of the features which are not in F_p and presents to the users the top N (where N is the number of recommendations) features with the highest recommendation scores.

To compare different recommendation algorithms, we describe the results of applying a fivefold cross-validation experiment [44]. For each product p in the test data, $L = 3$ features are randomly selected and used to represent the product profile. One of the remaining features, f_t, is randomly selected as the target feature, and each recommendation algorithm is evaluated based on its predictive power in recommending the target feature.

Hit ratio results for three algorithms at different sizes of recommendations are shown in Fig. 17.7. As can be seen, although BPRMF returns good performance when $N > 45$, it does not perform well at higher ranks. Assuming that the user is likely to look at the first ten recommendations, user-base kNN returns the best performance in comparison with the other two methods.

17.5.3 Addressing the Cold Start Problem

One of the problems frequently experienced with typical recommender systems, including systems based on collaborative filtering, is the product cold start problem which occurs when not enough is known about a product to make useful personalized recommendations. This situation can typically arise when the user's description of a product matches very few features in the database. This makes it difficult to find

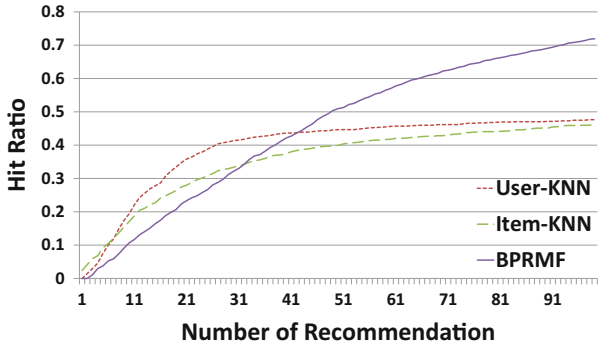

Fig. 17.7 Hit ratio comparison of different recommendation algorithms

a good neighborhood for the product, and as this neighborhood is the basis for the prediction of item scores, the recommendations accuracy is negatively affected. The solution to this problem usually involves a form of bootstrapping to enrich the initial product profiles. In the feature recommendation system, *association rule mining* can be used to enhance the initial product profile and then the discovered rules can be integrated into a hybrid recommendation framework using the kNN. Given an initial small profile for a target product, a preliminary set of recommendations are generated through association rule mining. These features are then shown to the user, and those accepted by the user are added to the product profile. The standard kNN approach is then applied on this augmented profile to generate additional recommendations.

Association rule mining is described in more detail by Menzies [36] in Chap. 3. Association rules identify groups of items based on patterns of co-occurrence across transactions. In this context each product is viewed as a "transaction," and association rules are generated among sets of features that commonly occur together among a significant number of products. The sets of features that satisfy a predefined support threshold are generally referred to as *frequent item sets*; however, in the context of domain analysis we refer to them as *frequent feature sets*.

Association rules are used to address the cold-start problem by augmenting an initially sparse profile. When a partial profile is matched against the antecedent of a discovered rule, the items on the right-hand side of the matching rules are sorted according to the confidence values for the rule, and the top ranked items from this list form the recommendation set. In order to reduce the search time, the frequent item sets can be stored in a directed acyclic graph, called a frequent itemset graph (FIG) [38, 46].

The graph is organized into levels from 0 to k, where k is the maximum size among all discovered frequent item sets. Each node at depth d in the graph corresponds to a frequent item set I of size d and is linked to item sets of size $d + 1$ that contain I at the next level. The root node at level 0 corresponds to the empty

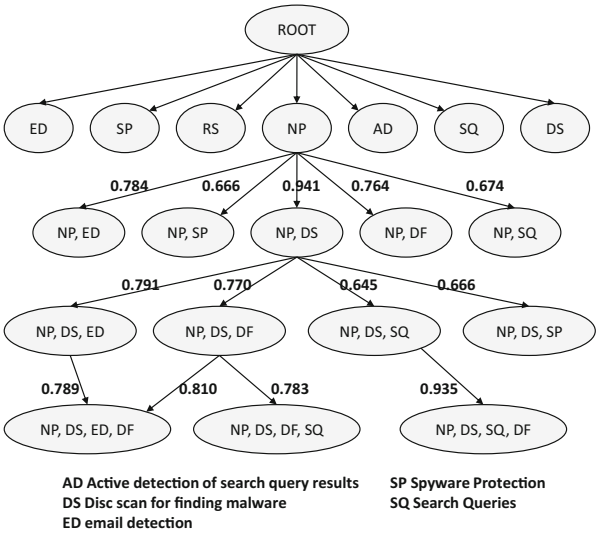

Fig. 17.8 A subset of a frequent itemset graph for the *Network Intrusion Protection* augmented with confidence scores

items set. Each node also stores the support value of the corresponding frequent item set.

Given an initial profile comprised of a set of features f, the algorithm performs a depth-first search of the graph to level $|f|$. Each candidate recommendation r is a feature contained in a frequent itemset $f \cup \{r\}$ at level $|f + 1|$. For each such child node of f, the feature r is added to the recommendation set if the support ratio $\sigma(f \cup \{r\})/\sigma\{f\}$, which is the confidence of the association rule $f \Rightarrow \{r\}$, is greater than or equal to a pre-specified minimum confidence threshold. This process is repeated for each subset of the initial itemset f, and conflicting candidate recommendations are resolved by retaining the highest confidence values. The recommended features corresponding to rules with highest confidence are shown to the user, and those accepted by the user are added to the product profile.

Figure 17.8 shows a small subset of frequent feature sets mined from the anti-virus software features. The displayed itemset graph shows features associated with *network intrusion detection*. For example, one rule specified in this graph states that if *network intrusion detection* (NP) and *disk scan for finding malware* (DS) features are found in a product, then we have a confidence of 0.791 that the product will also contain an *email detection* feature.

The effect of association rule mining on the performance of the system can be shown experimentally. For illustrative purposes we conducted a fivefold cross-validation experiment. In each of the five runs, one of the folds served as a testing set while the frequent itemset graph was generated from the remaining folds. For each product in the test set, $L = 3$ features were selected and the remaining features are removed from the profile. The frequent itemset graph was then used to

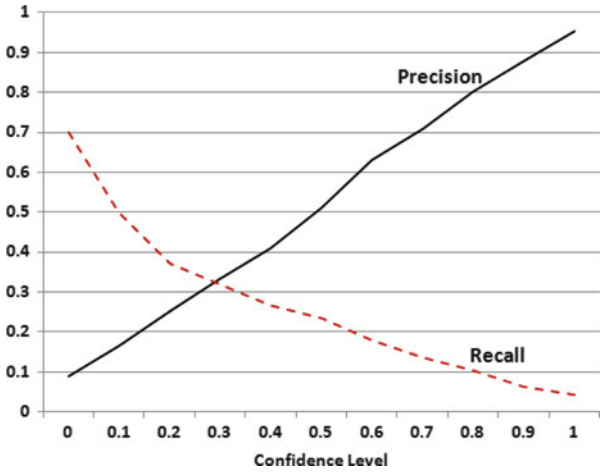

Fig. 17.9 Precision and recall at different levels of confidence

generate recommendations, and those recommendations with confidence scores of 0.2 or higher were recommended to the user. The produced recommendations can be evaluated in terms of precision and coverage, where precision is defined as the fraction of recommended items that are originally part of the product profile, and coverage as the fraction of profiles which receives recommendations.

Figure 17.9 shows the precision and recall for different levels of confidence. For example, at a confidence level of 0.2, the precision of the generated recommendations is 25 % and recall is 37 %. The association rule approach can therefore achieve high precision in its recommendations, but at the cost of lower recall. These observations support our earlier claims that association rule mining can be useful for identifying a small set of previously unused features with a high degree of precision.

To simulate the step in which the user evaluates the initial recommendations, the correct recommendations can be automatically accepted, and the incorrect ones rejected based on the known data stored in the product-by-feature matrix. These accepted recommendations are then used to augment the initial product profile of size $L = 3$, and the augmented profile is given as input to the kNN recommender to generate more recommendations. We label this hybrid method as kNN+.

The hybrid recommender can also be evaluated using the cross validation experiment, with the small modification that the left-out item is selected from the set of features that are not part of the augmented profile. Figure 17.10 compares the hit ratio results of the user-based kNN approach with the kNN+ method. As can be seen, the quality of recommendations is significantly improved when association rules are used to augment the product profile before running the kNN algorithm. This difference represents a 0.1 improvement in hit ratio at rank of 20.

These results demonstrate the viability of using recommendation systems to recommend features during the domain analysis process.

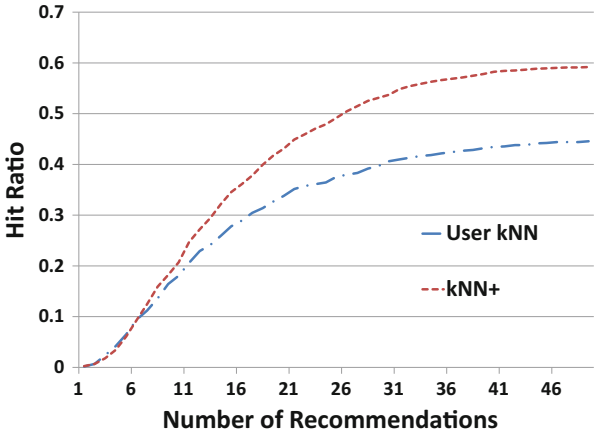

Fig. 17.10 Comparison of hit ratio for the hybrid method and the user-based kNN

17.6 Conclusion

In this chapter we have highlighted two specific applications of recommendation system in the requirements engineering domain. However, as pointed out earlier by Felfernig et al. [24] and Maalej and Thurimella [34] the potential exists for a far broader set of applications. The requirements engineering process has many of the characteristics of fields in which recommendation system technology has had significant impact. Its people-intensive upfront activities, and large quantities of data in the form of formal requirements, feature requests, and other very informal discussion posts, create an environment which can clearly benefit from social media tools such as recommendation systems. As such, the two in-depth examples we have described in this chapter serve as a proof of concept for the potential that exists to use recommendation systems to support a wide range of requirements-related activities in the future.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE T. Knowl. Data En. **17**(6), 734–749 (2005). doi:10.1109/TKDE.2005.99
2. Alves, V., Schwanninger, C., Barbosa, L., Rashid, A., Sawyer, P., Rayson, P., Pohl, K., Rummler, A.: An exploratory study of information retrieval techniques in domain analysis. In: Proceedings of the International Software Product Lines Conference, pp. 67–76 (2008). doi:10.1109/SPLC.2008.18
3. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E.: Recovering traceability links between code and documentation. IEEE T. Software Eng. **28**(10), 970–983 (2002). doi:10.1109/TSE.2002.1041053

4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
5. Castro-Herrera, C.: A hybrid recommender system for finding relevant users in open source forums. In: Proceedings of the International Workshop on Managing Requirements Knowledge, pp. 41–50 (2010). doi:10.1109/MARK.2010.5623811
6. Castro-Herrera, C., Cleland-Huang, J.: Utilizing recommender systems to support software requirements elicitation. In: Proceedings of the International Workshop on Recommendation Systems for Software Engineering, pp. 6–10 (2010). doi:10.1145/1808920.1808922
7. Castro-Herrera, C., Cleland-Huang, J., Mobasher, B.: Enhancing stakeholder profiles to improve recommendations in online requirements elicitation. In: Proceedings of the IEEE International Requirements Engineering Conference, pp. 37–46 (2009a). doi:10.1109/RE.2009.20
8. Castro-Herrera, C., Cleland-Huang, J., Mobasher, B.: A recommender system for dynamically evolving online forums. In: Proceedings of the ACM Conference on Recommender Systems, pp. 213–216 (2009b). doi:10.1145/1639714.1639751
9. Castro-Herrera, C., Duan, C., Cleland-Huang, J., Mobasher, B.: Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. In: Proceedings of the IEEE International Requirements Engineering Conference, pp. 165–168 (2008). doi:10.1109/RE.2008.47
10. Castro-Herrera, C., Duan, C., Cleland-Huang, J., Mobasher, B.: A recommender system for requirements elicitation in large-scale software projects. In: Proceedings of the ACM SIGAPP Symposium on Applied Computing, pp. 1419–1426 (2009c). doi:10.1145/1529282.1529601
11. Chen, K., Zhang, W., Zhao, H., Mei, H.: An approach to constructing feature models based on requirements clustering. In: Proceedings of the IEEE International Requirements Engineering Conference, pp. 31–40 (2005). doi:10.1109/RE.2005.9
12. Chen, W.Y., Zhang, D., Chang, E.Y.: Combinational collaborative filtering for personalized community recommendation. In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 115–123 (2008). doi:10.1145/1401890.1401909
13. Cheng, B.H.C., Atlee, J.M.: Research directions in requirements engineering. In: Proceedings of the Future of Software Engineering, pp. 285–303 (2007). doi:10.1109/FOSE.2007.17
14. Christley, S., Madey, G.: Analysis of activity in the open source software development community. In: Proceedings of the Hawaii International Conference on Systems Science, pp. 166b:1–166b:10 (2007). doi:10.1109/HICSS.2007.74
15. Cleland-Huang, J., Czauderna, A., Gibiec, M., Emenecker, J.: A machine learning approach for tracing regulatory codes to product specific requirements. In: Proceedings of the ACM/IEEE International Conference on Software Engineering, vol. 1, pp. 155–164 (2010). doi:10.1145/1806799.1806825
16. Cleland-Huang, J., Dumitru, H., Duan, C., Castro-Herrera, C.: Automated support for managing feature requests in open forums. Commun. ACM **52**(10), 68–74 (2009). doi:10.1145/1562764.1562784
17. Damian, D., Zowghi, D.: The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In: Proceedings of the IEEE Joint International Conference on Requirements Engineering, pp. 319–330 (2002). doi:10.1109/ICRE.2002.1048545
18. Davis, A.M., Tubío, Ó.D., Hickey, A.M., Juzgado, N.J., Moreno, A.M.: Effectiveness of requirements elicitation techniques: empirical results derived from a systematic review. In: Proceedings of the IEEE International Requirements Engineering Conference, pp. 176–185 (2006). doi:10.1109/RE.2006.17
19. Davril, J.M., Delfosse, E., Hariri, N., Acher, M., Cleland-Huang, J., Heymans, P.: Feature model extraction from large collections of informal product descriptions. In: Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 290–300 (2013). doi:10.1145/2491411.2491455
20. De Lucia, A., Fasano, F., Oliveto, R., Tortora, G.: Enhancing an artefact management system with traceability recovery features. In: Proceedings of the IEEE International Conference on Software Maintenance, pp. 306–315 (2004). doi:10.1109/ICSM.2004.1357816

21. Decker, B., Ras, E., Rech, J., Jaubert, P., Rieth, M.: Wiki-based stakeholder participation in requirements engineering. IEEE Software **24**(2), 28–35 (2007). doi:10.1109/MS.2007.60
22. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Mach. Learn. **42**(1–2), 143–175 (2001). doi:10.1023/A:1007612920971
23. Dumitru, H., Gibiec, M., Hariri, N., Cleland-Huang, J., Mobasher, B., Castro-Herrera, C., Mirakhorli, M.: On-demand feature recommendations derived from mining public product descriptions. In: Proceedings of the ACM/IEEE International Conference on Software Engineering, pp. 181–190 (2011). doi:10.1145/1985793.1985819
24. Felfernig, A., Schubert, M., Mandl, M., Ricci, F., Maalej, W.: Recommendation and decision technologies for requirements engineering. In: Proceedings of the International Workshop on Recommendation Systems for Software Engineering, pp. 11–15 (2010). doi:10.1145/1808920.1808923
25. Frakes, W.B., Prieto-Diaz, R., Fox, C.J.: DARE: domain analysis and reuse environment. Ann. Software Eng. **5**(1), 125–141 (1998). doi:10.1023/A:1018972323770
26. Freyne, J., Jacovi, M., Guy, I., Geyer, W.: Increasing engagement through early recommender intervention. In: Proceedings of the ACM Conference on Recommender Systems, pp. 85–92 (2009). doi:10.1145/1639714.1639730
27. Guy, I., Zwerdling, N., Carmel, D., Ronen, I., Uziel, E., Yogev, S., Ofek-Koifman, S.: Personalized recommendation of social software items based on social relations. In: Proceedings of the ACM Conference on Recommender Systems, pp. 53–60 (2009). doi:10.1145/1639714.1639725
28. Hariri, N., Castro-Herrera, C., Mirakhorli, M., Cleland-Huang, J., Mobasher, B.: Supporting domain analysis through mining and recommending features from online product listings. IEEE T. Software Eng. **39**(12): 1736–1752 (2013). doi:10.1109/TSE.2013.39.
29. Hayes, J.H., Dekhtyar, A., Sundaram, S.K.: Advancing candidate link generation for requirements tracing: the study of methods. IEEE T. Software Eng. **32**(1), 4–19 (2006). doi:10.1109/TSE.2006.3
30. Hull, E., Jackson, K., Dick, J.: Requirements Engineering. 2nd edn. Springer, Heidelberg (2005). doi:10.1007/b138335
31. Laurent, P., Cleland-Huang, J.: Lessons learned from open source projects for facilitating online requirements processes. In: Proceedings of the International Working Conference on Requirements Engineering. Lecture Notes in Computer Science, vol. 5512, pp. 240–255 (2009). doi:10.1007/978-3-642-02050-6_21
32. Lim, S., Quercia, D., Finkelstein, A.: StakeNet: using social networks to analyse the stakeholders of large-scale software projects. In: Proceedings of the ACM/IEEE International Conference on Software Engineering, pp. 295–304 (2010). doi:10.1145/1806799.1806844
33. Lim, S.L., Damian, D., Ishikawa, F., Finkelstein, A.: Using Web 2.0 for stakeholder analysis: StakeSource and its application in ten industrial projects. In: Maalej, W., Thurimella, A. (eds.) Managing Requirements Knowledge, Chap. 10, pp. 221–242. Springer, Heidelberg (2013). doi:10.1007/978-3-642-34419-0_10
34. Maalej, W., Thurimella, A.: Towards a research agenda for recommendation systems in requirements engineering. In: Proceedings of the International Workshop on Managing Requirements Knowledge, pp. 32–39 (2009). doi:10.1109/MARK.2009.12
35. Manning, C.D., Raghavan, P., Schutze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
36. Menzies, T.: Data mining: a tutorial. In: Robillard, M., Maalej, W., Walker, R.J., Zimmermann, T. (eds.) Recommendation Systems in Software Engineering. Springer, Heidelberg, Chap. 3 (2014)
37. Minto, S., Murphy, G.C.: Recommending emergent teams. In: Proceedings of the International Workshop on Mining Software Repositories, pp. 5:1–5:8 (2007). doi:10.1109/MSR.2007.27
38. Mobasher, B., Dai, H., Luo, T., Nakagawa, M.: Effective personalization based on association rule discovery from web usage data. In: Proceedings of the ACM Workshop on Web Information and Data Management, pp. 9–5 (2001). doi:10.1145/502932.502935

39. Mockus, A., Herbsleb, J.D.: Expertise browser: a quantitative approach to identifying expertise. In: Proceedings of the ACM/IEEE International Conference on Software Engineering, pp. 503–512 (2002). doi:10.1145/581339.581401
40. Moraes, A., Silva, E., da Trindade, C., Barbosa, Y., Meira, S.: Recommending experts using communication history. In: Proceedings of the International Workshop on Recommendation Systems for Software Engineering, pp. 41–45 (2010). doi:10.1145/1808920.1808929
41. Noll, J.: Requirements acquisition in open source development: Firefox 2.0. In: Proceedings of the IFIP World Computer Conference, IFIP: International Federation for Information Processing, vol. 275, pp. 69–79. Springer, Heidelberg (2008). doi:10.1007/978-0-387-09684-1_6
42. Pressman, R.: Software Engineering: A Practitioner's Approach. 7th edn. McGraw-Hill, New York (2009)
43. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence, pp. 452–461 (2009)
44. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.): Recommender Systems Handbook. Springer, New York (2011). doi:10.1007/978-0-387-85820-3
45. Robertson, S., Robertson, J.: Mastering the Requirements Process. Addison-Wesley, Reading, MA (1999)
46. Sandvig, J.J., Mobasher, B., Burke, R.: Robustness of collaborative recommendation based on association rule mining. In: Proceedings of the ACM Conference on Recommender Systems, pp. 105–112 (2007). doi:10.1145/1297231.1297249
47. Scacchi, W.: Understanding the requirements for developing open source software systems. IEE Proc. Software **149**(1), 24–39 (2002). doi:10.1049/ip-sen:20020202
48. Scacchi, W.: Free/open source software development: recent research results and emerging opportunities. In: Companion Papers to the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 459–468 (2007). doi:10.1145/1295014.1295019
49. Schröter, A., Kwan, I., Panjer, L.D., Damian, D.: Chat to succeed. In: Proceedings of the International Workshop on Recommendation Systems for Software Engineering, pp. 43–44 (2008). doi:10.1145/1454247.1454263
50. Spertus, E., Sahami, M., Buyukkokten, O.: Evaluating similarity measures: a large-scale study in the Orkut social network. In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 678–684 (2005). doi:10.1145/1081870.1081956
51. Thayer, R.H., Dorfman, M.: Software Requirements Engineering. 2nd edn. Wiley, New York (1997)
52. Xiang, P.F., Ying, A.T.T., Cheng, P., Dang, Y.B., Ehrlich, K., Helander, M.E., Matchen, P.M., Empere, A., Tarr, P.L., Williams, C., Yang, S.X.: Ensemble: a recommendation tool for promoting communication in software teams. In: Proceedings of the International Workshop on Recommendation Systems for Software Engineering (2008). doi:10.1145/1454247.1454259
53. Ying, A.T.T., Robillard, M.: Developer profiles for recommendation systems. In: Robillard, M., Maalej, W., Walker, R.J., Zimmermann, T. (eds.) Recommendation Systems in Software Engineering. Springer, Heidelberg, Chap. 8 (2014)

http://www.springer.com/978-3-642-45134-8

Recommendation Systems in Software Engineering

Robillard, M.P.; Maalej, W.; Walker, R.J.; Zimmermann, Th.
(Eds.)

2014, XIII, 562 p. 109 illus., Hardcover
ISBN: 978-3-642-45134-8