A Survey on Mining Techniques for Feature Recommendations of Online Products

K. Prasanthi[1] and Shaik.Munwar [2]
Department of IT, SVEC, Tirupati, A.P.
prasanthikadirimangalam@gmail.com[1]
munwar.it@gmail.com [2]

Abstract **- Domain analysis is essential in both application development and product line development. A novel feature recommender system is presented to support the domain analysis process for reducing the human effort. The approach in this relies on data mining technique which mines product description from available online specifications to determine the common features across products as well as relationship among those features. A novel incremental diffusive clustering algorithm was proposed for feature discovery to find domain specific features. Association rule mining and k-nearest neighbor machine learning methods are used to make feature recommendations on domain analysis process. The main focus is on mining additional software repositories and directories to broaden the knowledge of the system and explore techniques to enhance the navigation and visualization of the recommending features.**

Keywords: Domain analysis, Recommender system, Clustering, Data mining, Feature extraction

IJERT

1. INTRODUCTION

Domain analysis plays an important role in the software development life cycle [1]. It is conducted in the early phases to generate ideas for a product and determine the commonalities and differences of a product within the domain. As such, it is an important component of the upfront software engineering. The most domain analysis techniques are "Feature-oriented Domain Analysis (FODA)" [2] and "Feature –Oriented Reuse Method (FORM)" [3] trust upon analyst manually reviewing the existing requirement specifications or competitors product brochures and websites. The extracted features of the products are therefore limited by the scope of the available specifications.

We address these limitations through presenting a novel approach for identifying a broader set of candidate features and then analyses the relationships between features and products, and uses the information to recommend features for a specific project. The approach takes initial product description as input, analyzes the description of that product, and generates related feature recommendations utilizing two different algorithms. The first algorithm uses the association rule mining [5] and second algorithm uses the k-nearest neighbor approach [6].

The set of recommended features is the end result which can fed into requirement engineering process to help stakeholders define features for a specific software product or product line. A feature recommender system was introduced which includes several different components. Those are

1. Screen scrapper – responsible for mining descriptors from online product specifications.
2. Incremental diffusive clustering algorithm – clusters descriptors into features and extract meaningful names for each cluster.
3. Product-by-feature matrix – generated as a by-product of the clustering process.
4. Frequent Item set Graph (FIG) - generates feature association rules [6].
5. Recommender system – utilizes both product-by-feature matrix and FIG to generate on-demand feature recommendations.

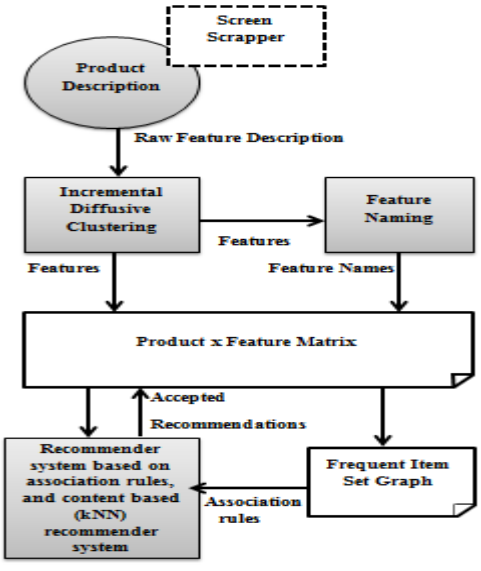

Fig (a) Constructing the Recommender System

The remainder of this paper is laid out as follows: Section 2 describes the approach for mining features from online product descriptions. Section 3 describes feature recommendations, which is then evaluated in Section 4.

2. FEATURE EXTRACTION

The Screen scraper utility was used to extract both product summary and also the list of features. The summary data was split into sentences which were added to the features extracted from the bulleted list, creating a combined list which we refer to as *feature descriptor*.

Table 1 represents a small subset of the features manually identified which shows that most of the analyzed antivirus systems include core features.

Incremental Diffusive Clustering:

Incremental Diffusive Clustering is the centerpiece of the feature discovery component of the system [7], [8], [9].

The goal of this component is to group together extracted product feature descriptors into an aggregate representation of candidate feature.

In this Incremental Diffusive Clustering (IDC) algorithm was first described and then report on a new quantitative and qualitative analysis that we conducted.

IDC contains the following

- Preprocessing
- Clustering
- Post processing
- Feature Naming

Preprocessing: Initial descriptors are first preprocessed by tokenizing them into a set of keywords, remove the commonly occurred word and then stemming the remaining keywords to their root form.

.

TABLE 1

A Small Sample of Features for Selection of Featured Antivirus

IJERT

	Ad-Aware TotalSecurity	Avast! Premier Antivirus	Baidu Antivirus	Bitdefender	BullGuard	CoGen Antivirus	Diyusof Antivirus	HitmanPro	JottiQ	Kaspersky	MCShield	McAfee	NANO AntiVirus	Norton	Panda	Snadar	ViriT eXplorer Lite	VIPRE Antivirus	XANA Evolution	Zone Alarm
Anti-browser exploit	1	1	1	1	0	0	1	0	0	1	1	1	1	1	1	1	0	0	1	1
Automatic external disk scanning	0	1	1	1	0	0	0	1	1	1	1	1	1	0	1	1	1	0	0	1
Anti-phishing	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0	0	1	1	0
Anti-rootkit	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0
Anti-Trojan	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
Auto USB detect	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	1	1	1	1	0
Bootable rescue disk	0	1	0	1	1	0	0	1	0	1	0	1	0	1	1	1	1	0	0	0
Compressed file scanning	1	1	0	1	1	1	1	0	0	1	0	1	0	1	1	1	0	1	1	0
Email attachment scanning	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0	1	0	1
Gaming mode	0	1	1	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1
Infected file quarantine	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1	0	0
Link scanner	0	0	0	1	0	0	1	0	0	1	0	0	1	0	1	1	1	0	0	0
On-demand scanning	0	1	1	1	0	0	0	0	1	1	0	1	0	1	1	1	1	0	1	0
Spam filter	0	1	1	1	0	1	1	1	0	1	1	0	0	0	0	0	1	0	1	0

Clustering: The feature descriptor vectors extracted in the preprocessing stage are automatically clustered using the incremental diffusive clustering algorithm.

Postprocessing: A postprocessing step is executed to optimize the identified feature which involves four tasks of

- Removing misfits,
- Recomputing centroids
- Checking for missing members, and finally
- Merging the similar features.

Feature Naming: Each feature is named by identifying the mediod, defined as the descriptor that is most representative of the feature's theme.

3. FEATURE RECOMMENDATIONS

Based upon the clusters generated using IDC, a binary product-by-feature matrix was created, which contains the complete set of recommendable features referred to as the *feature pool.*

Overview:

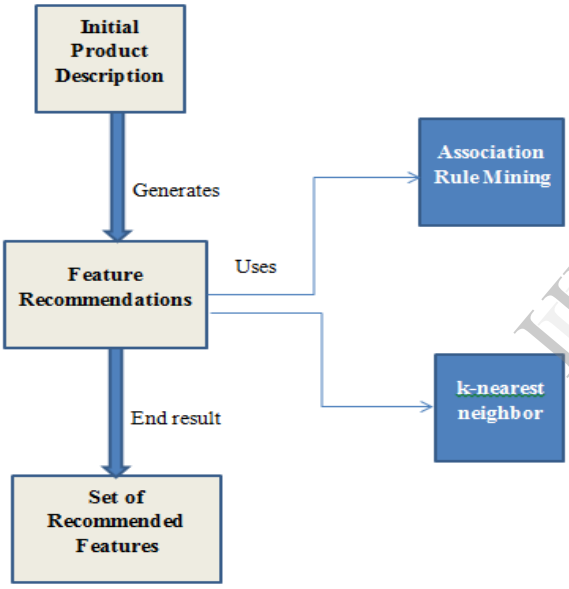

Fig (b) step by step process

Step 1: Creating Initial product description First an initial product profile is constructed or created in a format compatible with the feature model. For this, a domain analyst creates a short textual description of the product, which is then processed to match elements of the description to features in the feature pool.

This matching is accomplished by first performing basic preprocessing such as

1. Tokenization

2. Stemming

3. Removal of stop words

As a result of this step of the process, an initial product profile is appended to the product-by-feature matrix.

Step 2: Generating feature recommendations

In this step, *feature pool* is used to generate feature recommendations

Step 3: Using Association rule mining

In this step, feature recommendation uses association rule mining which is mainly used to address the problem that is "initial profile is relatively thin and contains only a few features".

Various algorithms exist for discovering the frequent item sets and association rules among them we chose to use the FP-Growth algorithm [10].

Step 4: Using kNN (k-nearest neighbor)

For this step, the augmented profile is given as input to kNN recommendation module. The product based k-nearest neighbor algorithm is used to be an efficient method for recommending features and requirements [11].

It is mainly used for producing more recommendations.

4. EVALUATION OF THE FEATURE RECOMMENDER

Evaluating a recommender system is a nontrivial task. There are several commonly used statistical methods for evaluating recommender systems [12].

Evaluating recommender systems and their algorithms is inherently difficult for several reasons.

- First, different algorithms may be better or worse on different data sets. Many collaborative filtering algorithms have been designed specifically for data sets where there are many more users than items .Such algorithms may be entirely inappropriate in a domain where there are many more items than users.
- The second reason that evaluation is difficult is that the goals for which an evaluation is performed may differ.

IJERT

5. RELATED WORK

In the year 1990, K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson, The *feature-oriented* concept is based on the emphasis placed by the method on identifying those features a user commonly expects in applications in a domain. This method, which is based on a study of other domain analysis approaches, defines both the products and the process of domain analysis. The report also provides a comprehensive example to illustrate the application of the FODA method to a representative class of software systems [2].

Domain engineering: An encompassing process which includes domain analysis and the subsequent construction of components, methods, and tools that address the problems of system/subsystem development through the application of the domain analysis products.

Domain model: A definition of the functions, objects, data, and relationships in a domain.

Feature: A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems [American 85].

Software architecture: The high-level packaging structure of functions and data, their interfaces and control, to support the implementation of applications in a domain.

Software reuse: The process of implementing new software systems using existing software information.

In the year 1998, K.C. Kang, S. Kim, J. Lee, K. Kim, G.J. Kim, and E. Shin, FORM (Feature-Oriented Reuse Method) [3] is a systematic method that looks for and captures commonalties and differences of applications in a domain in terms of "features" and using the analysis results to develop domain architectures and components. The model that captures the commonalties and differences is called the “feature model” and it is used to support both engineering of reusable domain artifacts and development of applications using the domain artifacts. Once a domain is described and explained in terms of common and different “units” of computation, they are used to construct different “feasible” configurations of reusable architectures.

The use of "features" is motivated by the fact that customers and engineers often speak of product characteristics in terms of “features the product has and/or delivers”. They communicate requirements or functions in terms of features and, to them, features are distinctively identifiable functional abstractions

In the year 2007, J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, Collaborative filtering (CF) is the process of filtering or evaluating items through the opinions of other people [5]. CF technology brings together the opinions of large interconnected communities on the web, supporting filtering of substantial quantities of data. CF takes its roots from something humans have been doing for centuries - sharing opinions with others.

As a formal area of research, collaborative filtering got its start as a means to handle the shifting nature of text repositories. As content bases grew from mostly "official" content, such as libraries and corporate document sets, to "informal" content such as discussion lists and e-mail archives. Pure content-based techniques were often inadequate at helping users find the documents they wanted.

IJERT

6. CONCLUSION

A novel feature recommender system was presented to support the domain analysis process. This system was used in the early phases of the software development life cycle. Quantitative experiments and qualitative analysis are used to evaluate the performance of the recommender system. This evaluation shows that recommender system is capable of making viable feature recommendation. A novel incremental Diffusive Clustering algorithm used to extract features from online product descriptions.

7.REFERENCES

[1] G. Arango and R. Prieto-Diaz, Domain Analysis: Acquisition of Reusable Information for Software construction. IEEE CS Press May 1989.
[2] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Technical Report CMU/SEI-90-TR-021, Software Eng. Inst., 1990.
[3] K.C. Kang, S. Kim, J. Lee, K. Kim, G.J. Kim, and E. Shin, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," Annals of Software Eng., vol. 5, pp. 143-168, 1998.
[4] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th Int'l Conf. Very Large Data Bases.
[5] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative Filtering Recommender Systems," The Adaptive Web, pp. 291-324,Springer, 2007.
[6] J. Sandvig, B. Mobasher, and R. Burke, "Robustness of Collaborative Recommendation Based on Association Rule Mining," Proc. ACM Conf. Recommender Systems, 2007.
[7] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli, "On-Demand Feature Recommendations Derived from Mining Public Software Repositories,"Proc. 33rd Int'l Conf. Software Eng., p. 10, May 2011.
[8] C. Duan, J. Cleland-Huang, and B. Mobasher, "A Consensus Based Approach to Constrained Clustering of Software Requirements," Proc. 17th ACM Conf.
[9] Information and Knowledge Management, pp. 1073-1082, 2008.
[10] J. Cleland-Huang and B. Mobasher, "Automated Detection of Recurring Faults in Problem Anomoly Reports," SERC Report to Lockheed Martin, 2009.
[11] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '00), June 2000.
[12] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, "A Recommender System for Requirements Elicitation in Large- Scale Software Projects," Proc. ACM Symp. Applied Computing, pp. 1419-1426, 2009
[13] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl, "Evaluating Collaborative Filtering Recommender Systems," ACM Trans. Information Systems, vol. 22, pp. 5-23, 2004.

IJERT