Domain Orientation by Feature Modeling and Recommendation for Product Classification

Kanifnath S. Hirave
PG Student,
Department of Computer Engineering.
VPCOE,
Baramati, India
E-mail: kanifhirave29@gmail.com

Prof.D.B.Hanchate
Assistant Professor,
Department of Computer Engineering.
VPCOE,
Baramati, India
E-mail: dinesh_b_hanchate@rediffmail.com

Abstract**—** Domain Analysis says that activity occurring before system analysis provides domain model. Domain model is input to the system analysis to the designer‘s tasks. Domain analysis is the procedure of identifying, organizing, analyzing, and modeling features common to a specific domain. The complete process is very time consuming and more man power is required for it. There are various projects which require extensive domain analysis activities. In proposed method, recommended system is used to reduce human efforts of performing domain analysis. It is not easy to discover relationship between items in a large database of sales transactions but there are some algorithms for solving this problem. Data mining techniques are used to discover common features across products as well as relationships among those features .Incremental diffusive algorithm is used to extract features. Bi-Partity Distribution technique is used for feature recommendations during the domain analysis process

Keywords: *Domain Analysis, Clustering, Incremental Diffusive Clustering (IDC), Recommender systems, Bi-Paritity Distribution.*

*****

I. INTRODUCTION

Domain analysis is the process of identifying, organizing, analyzing, and modeling features common to a particular domain [3], [4]. It is conducted in primary stage of the software development life-cycle (SDLC) to produce ideas for a product, to find out similarities and differences in a domain, and to identify opportunities for reuse. It is a leading element of the software engineering process. Most domain analysis techniques, such as the feature-oriented domain analysis (FODA) [5] or the feature-oriented reuse method (FORM) [6] depend on analysts manually reviewing the existing requirement specifications or participant's product brochures and websites, and are quite labor intensive. The accomplishment of these methods is reliant on the accessibility of relevant documents or access to the existing project repositories, as well as the knowledge and capability of the domain analyst. Other approaches such as the domain analysis and reuse environment (DARE)[7] utilize data mining and information retrieval methods to provide automated support for feature identification and extraction, but have a tendency to focus their efforts on only a small requirements specifications. The extracted features are limited by the scope of the available specifications. In this paper, address these limitations through presenting a new approach for finding a larger set of applicant features. In previous methods that extract features from exclusive project repositories, this approach extracts raw feature descriptions. It analyzes the relationships between features and products, and utilizes this information to recommend features for an explicit project. This method takes as input to initial product description, examines this description, and then produces related feature recommendations utilizing Bi-Paritity distribution Techniques. Previously introduced feature recommender system is useful for limited domain with comparatively small-sized software products. Antivirus software, multimedia, and photography applications are some of the software products.

Step #1 : Enter Initial product description
Amiti Free Antivirus is an effective and easy to use free antivirus for your PC. Protects against viruses, trojans, worms and malware. Amiti Antivirus has built-in real-time memory shields, scheduling, multiple skin and translations support. Uses famous clamav antivirus engine library.

Step #2:Confirm Features
We have identified the following features from your initial product description. Please confirm
Easy to use
Protects against viruses, worms, viruses, Trojans , Malware
Real time memory shields
Scheduling, multiple skin and translations support.

Step #3: Recommended Features
Based on the features you have already selected we recommended the following three features. Please confirm
× Network intrusion detection
Real time monitoring
Web history and cookies management

Fig1: Example of feature Recommendations

In proposed system we extend this prior work by making several extra contributions. Firstly, provide a more detailed analysis of our technique for mining features from online product listings utilizing incremental diffusive clustering (IDC) [15][16] algorithm and then Bi-Partity distribution is used for feature recommendations.

The final output is a set of recommended features which is input for the requirements engineering process to help project stakeholders who define the features for a specific software product or product line. In this approach, the user can interrelate with the system to provide response on candidate features. In this example scenario, the user rejects network intrusion; however, file monitoring and web history is added to the initial product profile. Additional recommendations are then generated based on this profile

The remainder of this paper is laid out as follows: Section II describes old methods for domain analysis. Section III describes implementation details of proposed system. Section IV describes datasets. Section V describes results. Finally Section VI describes conclusions.

II.LITERATURE SURVEY

This work fills the difference between automated feature detection and recommender systems. So, this section provides a brief background survey on each of these areas.

At the beginning, there is no any policy for domain analysis, domain analysis is conducted manually [3]. Domain analysis is carried out with help of data flow diagrams. Domain analysis can be considered as a process which is occurring previous to system analysis. [3]. Organized detection and use of cohesion across related software systems is required for successful software reuse. Domain analysis offer, a general report of the necessities of that class of structures and a set of methods for their implementation with the help of observing related software systems. FODA [5] create methods for accomplishment a domain analysis and define the products of the domain analysis process. The important technical condition for completing effective software reuses efficient detection and use of unity across related software systems. In FORM [5], inspection of a class of related systems and the cohesion of primary systems exist. It is possible to achieve a set of reference models. FORM starts with an analysis of agreement among applications in a particular domain in terms of services, operating environments, domain tools. The feature model (FM)[5] is defined as construction during the analysis is called feature model. Feature model captures commonality. Domain Analysis and Reuse Environment [6] is CASE tool which helps in domain analysis of finding and recording the similarities and differences of related software systems. DARE [6] helps to capture of domain information from experts in a domain. Captured domain information is stored in a domain catalog, which is enclosed a general architecture for the domain and domain specific components. We also studied the problem of finding out association rules amongst items in huge database of sales transactions. There are two algorithms i.e. Apriori and Apriori-TID [7, 8] algorithm for Association Rule Mining (ARM) which is well known algorithm to find Association rules which are used for affinities among items [7,8].The process of estimating items through the views of other people is called as Collaborative filtering (CF)[9] .CF technology fetches organized views of large interrelated publics on the web which supporting filtering of large amounts of data. We studied the very important part of collaborative filtering, its key uses for users of the principle and exercise of CF [9] algorithms. We also studied challenges of a CF recommendation system and evaluation of Collaborative Filtering [10].

III. SYSTEM IMPLEMENTATION

The proposed framework is organized into two phases, Offline phase and Online phase respectively. The system architecture is as shown in fig.2

Offline phase: Dataset i.e. product descriptions are collected from sourceforge.net Collected dataset is passed through the preprocessing, IDC algorithm, post processing, and creation of Product × Feature matrix. This is stage used for feature extraction.

Preprocessing

Initial feature descriptors are first preprocessed by converting them into a set of keywords; eliminating commonly occurring words i.e. stop words, and stemming the remaining keywords to their root form i.e., terms. Given the vocabulary of all such terms $T=\{t_1,t_2,..,t_w\}$each feature descriptor, f_i, is represented as a vector of terms, $v_i=\{f_{i,1},f_{i,2},..,f_{i,w}\}$ where f_{ij} is a term weight representing the number of occurrences of term t_j in the feature descriptor f_i. These term weights are then converted using a standard term frequency-inverse document frequency(tf-idf) method [13] Such that $tf\text{-}idf(f_{ij})=f_{i\ j}.log_2(D/df_j)$ where D denotes the total number of feature descriptors and df_j denotes the number of feature descriptors containing term tj. Finally the converted vector(with tf-idf weights) is normalized to a unit vector resulting in the vector $v_i=(F_{i,1}, F_{i,2}, F_{i,W})$

The following steps are then executed to identify features.

A. Granularity

In order to decide how many features or clusters to produce for a given product category, IDC [15], [16] uses a revised version of Can's metric [17] which considers the degree to which each feature descriptor distinguishes itself from other feature descriptors. The ideal number of clusters K is computed as follows:

$$K=\sum_{v_i \in F}\sum_{j=1}^{W}\frac{f_i,j}{|v\_i|}\cdot\frac{f_i,j}{N_j}=\sum_{j=1}^{w}\frac{f_{i,j}^2}{N_j}\ldots\ldots(1)$$

where Nj represents the total number of occurrences of term $t_{j.}$

B. Clustering

The feature descriptor vectors extracted in the preprocessing stage is automatically clustered using the incremental diffusive clustering algorithm (IDC) [15], [16] to extract a meaningful and cohesive set of final features. In each iteration, IDC first automatically clusters the feature descriptors, and then, it detects and keeps as it is the highest cluster. Once this cluster has been selected, its dominant terms

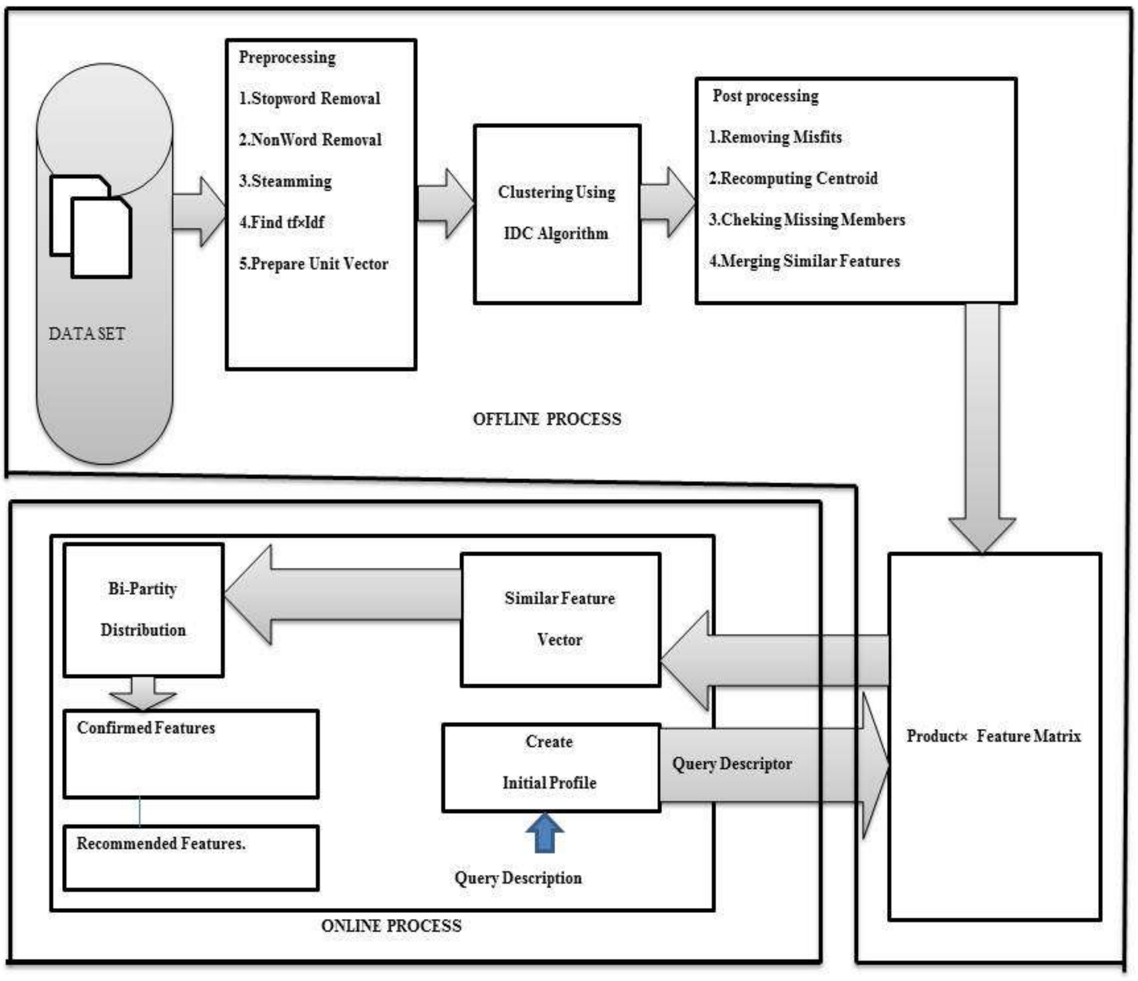


Fig: Flow of System

are recognized and removed from all clusters. In our initial version of IDC, a consensus-based spherical K-means clustering methodology [16] was used in each iteration to increase the cohesiveness of intermediate clusters. We clustered feature descriptors in each of the sourceforge categories separately, and then merged similar clusters across the categories.

C. Selecting the Best Cluster

In each iteration, IDC stimulates the best cluster to the status of a feature. Based on initial interpretations, the best cluster from a set $L=\{C_1,C_2,\ldots,C_K\}$ is one that has high levels of cohesion with wide coverage of the topics. To measure cohesion for a cluster $C_i = \{v_{i,1},v_{i,2},..,v_{i,r}\}$ with centroid $\overline{C_i}$. The similarity between the feature descriptors and their associated centroids is computed and averaged as,

$$\left(\sum_{j=1}^{r} sim\left(V_{ij},\overline{C_i}\right)\right)/r \ldots\ldots\ldots\ldots\ldots\ldots\ldots..(2)$$

While topic coverage is computed as,

$$\left(\sum_{j=1}^{r} sim\left(V_{ij},\overline{C_i}\right)\right)\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

Cohesion and topic coverage scores are added together, and the cluster with the highest combined score is selected as the best cluster and promoted to the status of a feature. Remember that if cohesion itself used to select the best cluster, the algorithm would have been biased towards small clusters.

D. Removing Dominant Terms

In order to remove dominant terms, terms exhibiting weights above a predefined threshold (0.15) in the centroid vector of the “best” cluster are selected. Since the centroids are also normalized, this threshold is held constant. These terms are then removed from all descriptors in the data set. For example, if dominant terms are identified as instant, e-mail, and encrypt, then a descriptor originally specified as Encrypt e-mail messages before transmission is reduced to messages before transmission with the word before also removed as a stop word. Future rounds of clustering are then performed on the reduced version of descriptors. This process is repeated until the targeted number of features, determined previously in step 1, has been identified.

E. Post processing

A post processing step is performed to enhance the identified features. This step involves the four tasks of

1. Removing misfits

2. Recomputing centroids,

3. Checking for missing members, and finally

4. Merging similar features.

Misfits are removed by computing the similarity score between each centroid and feature descriptor in the corresponding cluster. Descriptors whose similarity to the centroid drop below a given threshold value (set to 0.35 based on experiential observations) are removed from the cluster. Centroids are then shifted according to the remaining feature descriptors using the same SPK algorithm adopted in the initial clustering step. Missing feature descriptors are recognized by re-computing the cosine similarity between every feature descriptor not currently assigned to a cluster, and the centroid of that cluster. Any feature descriptor showing a similarity score higher than a given threshold (set to 0.35) is added to that cluster. Finally, similar clusters are merged by computing the cosine similarity between each pair of centroids. Any pair of clusters showing a score above a given threshold (set to 0.55) is finally merged into a single feature.

F. Feature Naming

Each feature is named by identifying the medoid, defined as the descriptor which is most representative of the feature’s theme. The medoid is identified by first calculating the cosine similarity between each feature descriptor and the centroid of the cluster, and then averaging all term weights in the feature descriptor vector above a certain threshold (0.1). The cosine similarity and the average of dominant term weights are then added together to produce the score of each feature descriptor. The feature descriptor scoring the highest value is selected as the medoid and the corresponding original feature descriptor (from sourceforge) is selected as the name for the feature. This approach produces relatively meaningful names. As an example, a feature based on the theme of updat, databas, automat, viru was subsequently named Virus definition update and automatic update supported.

G. Merging Category Clusters

This method includes processing each product category separately, and then merging them into a single product-by-feature matrix. This directly meets scalability issues of dealing with large numbers of features, and has the additional extensibility benefit of allowing new product categories to be added incrementally. Merging is accomplished through computing the cosine similarity between each pair of features, and then merging features exhibiting

Online phase: online phase is used for feature recommendation which is based upon the clusters generated using IDC, we create a binary product-by-feature matrix, $M(m_{ij})_{P\times F}$ where P represents the number of products, *F* is the number of identified features and m_{ij} is 1 if and only if the feature j includes a descriptor originally mined from the product i. This matrix, which contains the complete set of recommendable features referred to as the feature pool from now on, is used to generate feature recommendations in the following steps.

1. Creating Initial Product Profile

First an initial product profile is built in a format compatible with the feature model. To achieve this, the 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 tokenization, stemming, and removal of stop words, and then converting the product description p to a term vector $p=(w_{1,p},w_{2,p},\ldots,w_{n,p})$ where each dimension corresponds to a separate term. We used the standard term frequency-inverse document frequency approach [13], also known as tf-idf, which calculates the weight for each term based on the normalized term frequency and the inverse document frequency.

Once the product description is converted to term vector form, it is compared to the term vector representation of each feature in the feature pool using a standard information retrieval metric such as the cosine similarity. Features are then ranked according to their similarity to the product description, and presented to the analyst for confirmation. As a result of this step, an initial product profile is appended to the product-by-feature matrix.

2. Feature Recommendations Using Bi-Parity Distribution

1. Calculate cosine similarity between term vector and feature pool.

$$Sim\ (dj,dk) = \frac{\sum_{i=1}^{n} f_{id_j}\cdot f_{id_k}}{\sqrt{\sum(f_i,d_j)^2}\ \cdot\sqrt{\sum(f_i,d_k)^2}} \ldots\ldots\ldots(4)$$

Where ,

   a. f_i is ith feature,

   b. d_j is jth descriptor,

   c. d_k is kth descriptor.

2. Rank features with respect to their weights.

3. Find fractions of number of occurrences of each feature to the total number .of occurrences.

4. Calculate Point Of Distribution (POD).

$$POD= \frac{\sum_{i=1}^{n} f_i(w)}{|f|} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$

Where,

w is weight associated with feature.

5. Partition the ranked feature vector around POD.

6. Label Upper partition as Confirmed features and Lower partition as Recommended features

IV. DATASET

Proposed method evaluated in the context of four different data sets collected from sourceforge[1,] sourceforge[2] sourceforge[3,] sourceforge[4]. Dataset is stored in text file.

1. Antivirus product descriptions extracted from sourceforge[1]
2. Multimedia video product descriptions extracted from sourceforge[2]
3. I-pod application product descriptions extracted from sourceforge[3]
4. Photography product descriptions extracted from sourceforge[4]

[1.]http://sourceforge.net/directory/os:windows/freshness:recently-updated/?q=antivirus

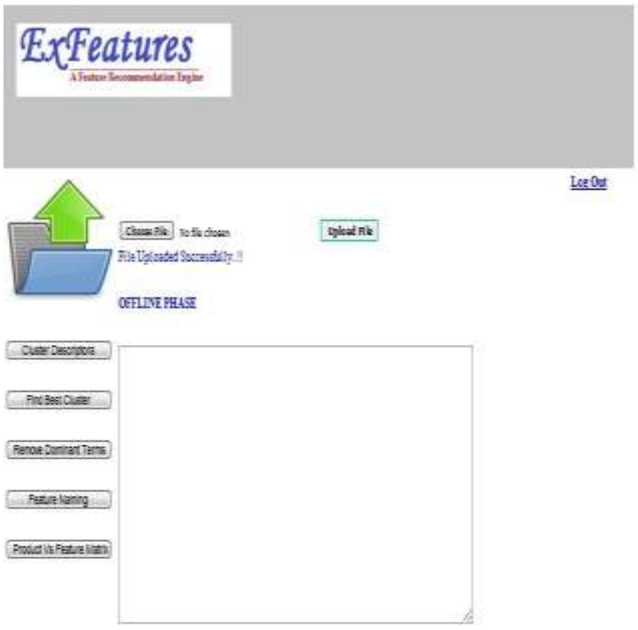
[2].http://sourceforge.net/directory/os:windows/freshness:recently-updated/?q=multimedia

[3].http://sourceforge.net/directory/os:windows/freshness:recently-updated/?q=ipod%20application

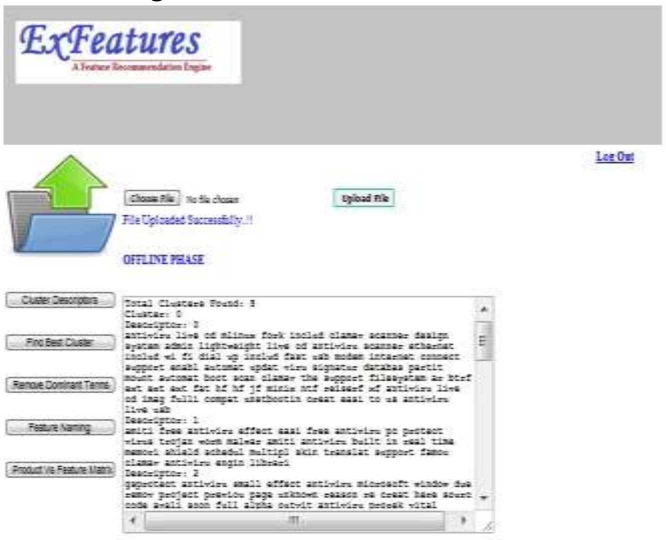
[4.]http://sourceforge.net/directory/os:windows/freshness:recently-updated/?q=ipod%20application

V. RESULTS

5.1 Upload Dataset

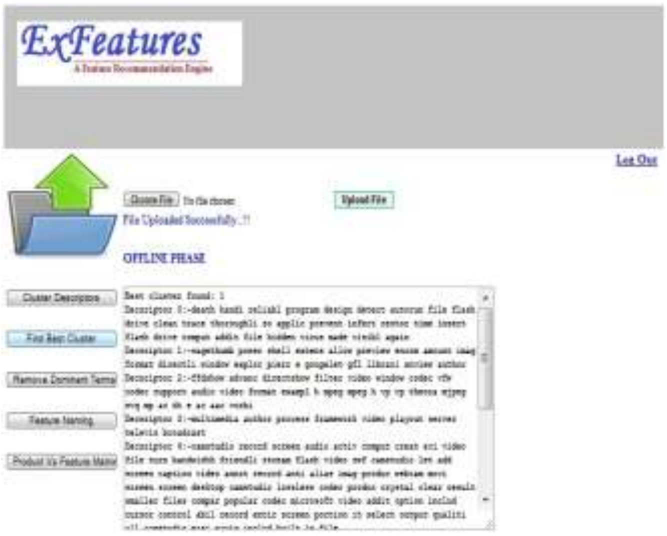

ExFeatures
A Feature Recommendation Engine
Log Out
Choose File No file chosen
Upload File
File Uploaded Successfully..!!
OFFLINE PHASE
Cluster Descriptors
Find Best Cluster
Remove Dominant Terms
Feature Naming
Product Vs Feature Matrix

5.2 Clustering


ExFeatures
A Feature Recommendation Engine
Log Out
Choose File No file chosen
Upload File
File Uploaded Successfully..!!
OFFLINE PHASE
Cluster Descriptors
Find Best Cluster
Remove Dominant Terms
Feature Naming
Product Vs Feature Matrix

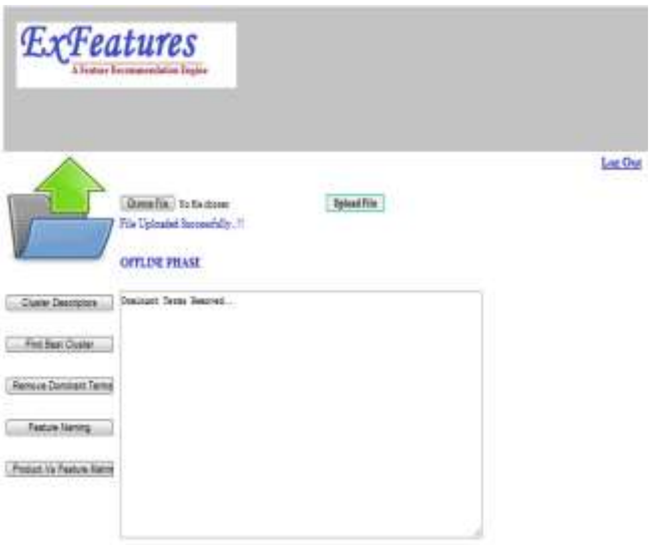
```
Total Clusters Found: 3
Cluster: 0
Descriptor: 0
antivirus live cd minux fork includ clamav scanner design
system admin lightweight live cd antivirus scanner ethernet
includ wi fi dial up includ fast usb modem internet connect
support enabl automat updat viru signatur databas partit
mount automat boot scan clamav the support filesystem as btrf
ext ext ext fat hf hf jf minix ntf reiserf xf antivirus live
cd imag fulli compat unetbootin creat easi to us antivirus
live usb
Descriptor: 1
amiti free antivirus affect easi free antivirus pc protect
virus trojan worm malwar amiti antivirus built in real time
memori shield schedul multipl skin translat support famou
clamav antivirus engin libclaml
Descriptor: 2
gcprotect antivirus small effect antivirus microsoft window due
remov project previou page unknown reason re creat here start
code avail soon full alpha outvit antivirus protek vital
```

5.3 Find Best Cluster


ExFeatures
A Feature Recommendation Engine
Log Out
Choose File No file chosen
Upload File
File Uploaded Successfully..!!
OFFLINE PHASE
Cluster Descriptors
Find Best Cluster
Remove Dominant Terms
Feature Naming
Product Vs Feature Matrix

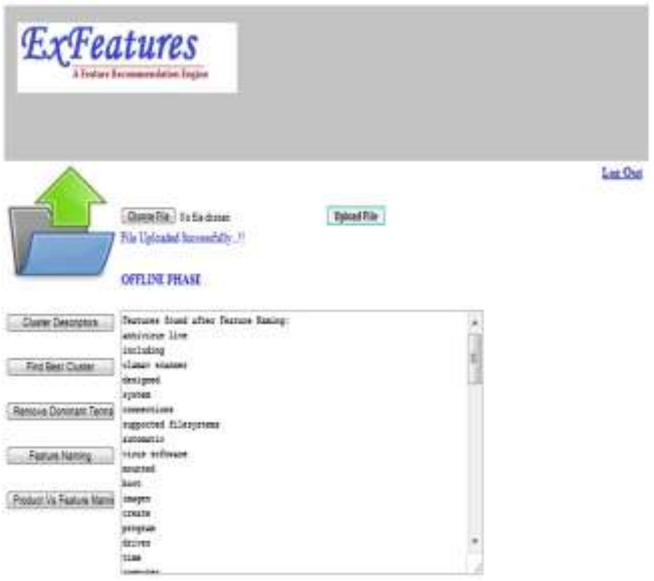
```
Best cluster found: 1
Descriptor 0:-death handi reliabl program design detect autorun file flash
drive clean trace thoroughli so applic prevent infect restor time insert
flash drive comput addit file hidden virus made visibl again
Descriptor 1:-pagethumb power shell extens allow preview excel amount imag
format directli window explor pixel e googeler gfl librari author author
Descriptor 2:-ffdshow advanc directshow filter video window codec vfw
codec support audio video format exampl h mpeg mpeg h vp vp theora mjpeg
mvq mp ac dt v ac aac vorbi
Descriptor 3:-multimedia author process framework video playout server
televis broadcast
Descriptor 4:-camstudio record screen audio activ comput creat avi video
file turn bandwidth friendli stream flash video swf camstudio let add
screen caption video annot record anti alias imag produc webcam movi
screen screen desktop camstudio lossless codec produc crystal clear result
smaller files compar popular codec microsoft video addit option includ
cursor control abil record entir screen portion it select output qualiti
[illegible]
```

5.4 Remove Dominant Terms


ExFeatures
A Feature Recommendation Engine
Log Out
Choose File No file chosen
Upload File
File Uploaded Successfully..!!
OFFLINE PHASE
Cluster Descriptors
Find Best Cluster
Remove Dominant Terms
Feature Naming
Product Vs Feature Matrix

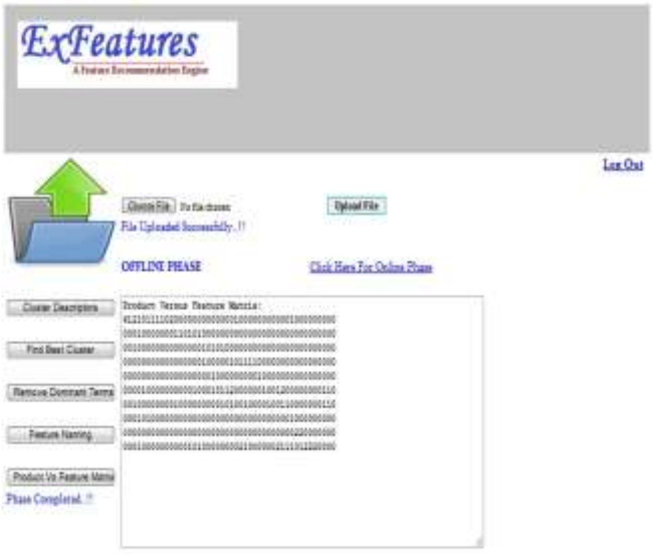
```
Dominant Terms Removed...
```

5.5 Feature Naming



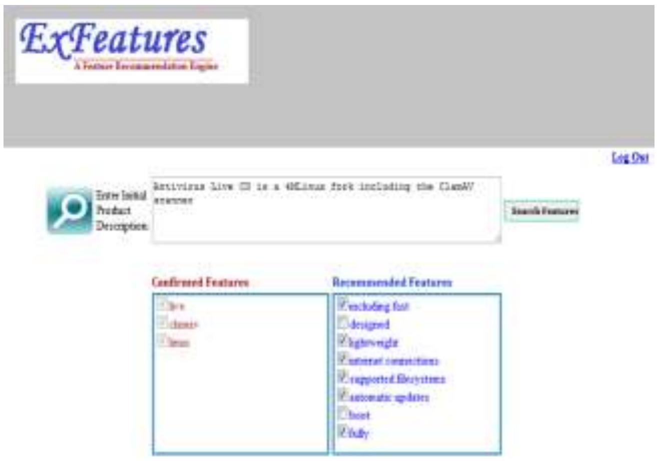
```
Features found after Feature Naming:
antivirus live
including
clamav scanner
designed
system
connections
supported filesystems
automatic
virus software
mounted
boot
images
create
program
drivers
time
```

5.6 Product × Feature Matrix



```
Product Versus Feature Matrix:
[illegible]
```

5.7 Confirmed Features and Recommended Features.



Antivirus Live CD is a 4MLinux fork including the ClamAV scanner

Confirmed Features:
- ☐ live
- ☐ clamav
- ☐ linux

Recommended Features:
- ✓ including fast
- ☐ designed
- ✓ lightweight
- ✓ internet connections
- ✓ supported filesystems
- ✓ automatic updates
- ☐ boot
- ✓ fully

VI. CONCLUSIONS

In proposed method presented a new feature recommender system to support the domain analysis process. This is a critical early phase part of the software development lifecycle and is essential in both application development and product line development. This system mines feature descriptors for hundreds of products from publicly available software repositories of product descriptions and uses this data to discover features and their associations. For feature discovery, this method proposed a new incremental diffusive clustering algorithm. In this method we use the Bi-Parity Distribution approach to make additional recommendations. This has the advantage of expanding an initially light product description before making a more wide set of recommendations.

ACKNOWLEDGEMENT

I avail this opportunity to express my deep sense of gratitude and whole hearted thanks to my guide Prof. D.B. Hanchate sir for giving his valuable guidance, inspiration and encouragement to embark this paper. Without his Coordination, guidance and reviewing, this task could not be completed alone. Prof. D.B. Hanchate sir gave me all the freedom I needed for this paper.

REFERENCES

[1] Negar Hariri, Carlos Castro-Herrera,Mehdi Mirakhorli,Jane Cleland-Huang,Bamshad Mobasher,"Supporting Domain Analysis through Mining and Recommending Features from Online Product Listings,"*IEEE Transactions on Software Engineering,* vol. 39,no. 12,december 2013

[2] Kanifnath S. Hirave, Prof. Dinesh Bhagwan Hanchate "Survey on ExFeature: A Feature Modeling and Recommending Technique For Domain Oriented Product Listing "*International Journal of Engineering Research and General Science* Volume 2, Issue 6, October-November, 2014 ,ISSN 2091-2730

[3] G. Arango and R. Prieto-Diaz, Domain Analysis: Acquisition of Reusable Information for Software Construction. *IEEE CS Press*, May 1989.

[4] 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.

[5] 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.

[6] W. Frakes, R. Prieto-Diaz, and C. Fox, "Dare: Domain Analysis and Reuse Environment," Annals of Software Eng., vol. 5, pp. 125-141, 1998.

[7] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. 20th Int'l Conf. Very Large Data Bases*,1994.

[8] R. Agrwal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. 20th Int'l Conf. Very Large Data Bases (VLDB '94)*, pp. 487-499, Sept. 1994.

[9] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative Filtering Recommender Systems," *The Adaptive Web*, pp. 291-324, Springer, 2007.

[10] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems*, vol. 22, pp. 5-23, 2004.

[11] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher. A recommender system for requirements elicitation in large-scale software projects. *Proc. of the 2009 ACM Symp. on Applied Computing*, pages 1419–1426, 2009.

[12] K. Chen, W. Zhang, H. Zhao, and H. Mei. An approach to constructing feature models based on requirements clustering. Requirements Engineering, *IEEE International Conference* on, 0:31–40, 2005.

[13] C.D. Manning, P. Raghavan, and H. Schutze, Introduction to Information Retrieval. *Cambridge Univ. Press*, 2008.

[14] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. *ACM.* pages 678–684, Chicago, Illinois, USA, 2005.

[15] 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 Engg.,* p. 10, May 2011

[16] C. Duan, J. Cleland-Huang, and B. Mobasher, "A Consensus Based Approach to Constrained Clustering of Software Requirements," *Proc. 17th ACM Conf. Information and Knowledge Management*, pp. 1073-1082, 2008.

[17] F. Can and E. Ozkarahan, "Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Data-bases," *ACM Trans. Database Systems*, vol. 15, pp. 483-517, 1990.