Linguistic Analysis of Crowd Requirements: An Experimental Study

Javed Ali Khan, Lin Liu, Yidi Jia, Lijie Wen
School of Software
Tsinghua University
Beijing, China
{linliu, wenlj}@tsinghua.edu.cn

***Abstract*—Users of today's online software services are often diversified and distributed, whose needs are hard to elicit using conventional RE approaches. As a consequence, crowd-based, data intensive requirements engineering approaches are considered important. In this paper, we have conducted an experimental study on a dataset of 2,966 requirements statements to evaluate the performance of three text clustering algorithms. The purpose of the study is to aggregate similar requirement statements suggested by the crowd users, and also to identify domain objects and operations, as well as required features from the given requirements statements dataset. The experimental results are then cross-checked with original tags provided by data providers for validation.**

Index Terms—Crowd-based RE, Smart home, Summarization, requirement clustering, experiment.

I. Introduction

Users are playing a central role in requirements engineering (RE) processes. Their involvement in different RE activities such as elicitation, prioritization and negotiation are considered pivotal [1]. With the pervasive use of social media, app stores and user forums, it becomes important for the software vendors to listen to the users and taking their needs and suggestions into consideration in future software design and evolution [4]. Crowd-based RE aims to cater for users' emerging needs through crowdsourcing platforms, such as, Amazon Mechanical Turk [2], TopCoder, or open source forums[8][9] and different online applications marketplaces [13][14]. Information gathered from these sources reflects users' needs and expectations for the software system.

When a large group of users are involved in the requirements elicitation process, they generate many requirements or feedbacks [14]. Many researchers have applied machine learning techniques on user's forum data [5] [6] [7] [10] [11] [3][4] to identify feature requests, bug reports or non-functional requirements. These approaches give useful insights based on crowd data, which can be used by requirement engineers for software evolution planning and bugs fixing.

Inspired from these research efforts, our study focused on the answering of the research question: *Can natural language processing tools together with text clustering algorithms generate good enough requirements summary?* We apply semi-automated approach on a requirements dataset, to identify domain objects, operations and features demanded by users. The proposed approach is experimented on an open dataset of smart home [1] and compared to smart home products online catalogues and the original tagging information provided in the dataset. While we have used the smart home dataset to summarize the large set of requirements and identify software features through semi-automated technique. Requirements collected from crowd users are also used to investigate the creative potentials in a crowd-based requirement acquisition task [2]. After the experiment, a preliminary goal model is constructed from the identified objects and operations as a visualization to requirement engineers and end-users.

II. Research Method

Objective of the study: The ultimate goal of this study is to evaluate how automated approaches can help software analysts identify useful information from a large set of crowd requirements.

Context: This experiment is conducted offline, in the laboratory setting, in the application domain of smart home. The context of requirements summarization is to identify software features from inputs of large groups of diversified users on emerging and unsatisfied requirements by automated and semi-automated data processing procedures. As an example, hundreds of thousand tweets [16], hundreds of open questions in open forums [9] and millions of reviews in online app stores [14] are generated nowadays for specific software. These tweets and reviews contain much useful information regarding software features, which is very expensive and time-consuming if processed manually [4]. This inspired us to conduct an experiment on the information gathered from the crowd to produce useful suggestions for requirements engineers in the form of software features.

Experimental design: The main methodological source that inspired our research is Wohlin's book, "Experimentation in Software Engineering" [41]. In particular, we choose to use an experimental method due to the reason that we run data analysis experiments on the requirements dataset to validate the above-mentioned objective. We conducted the experimental study to examine the performance of natural language processing (NLP) tool together with clustering algorithms in summarizing a large set of requirements on a requirements statement dataset generated by crowd. In this study, we compared three baseline unsupervised machine learning algorithms, such as K-Means, LDA and Biterm topic model, for their performance in the problem setting.

III. Experimental RE dataset

A requirements dataset on smart home was collected by Murukannaiah et al. using Amazon Mechanical Turk (AMT) [1]. The requirements statements are in the form of user stories, the general length of each natural language English sentences is similar to the length of tweets. The dataset is mainly composed of two parts corresponding to the two phases of the data collection. In the first phase, the crowd members were asked to describe their views about smart home. In the second phase, they asked the crowd who were not involved in the first phase to rate the requirements identified, in terms of novelty, clarity and usefulness. They have also gathered the demographics, personality traits and their creative potentials. There are 609 users of AMT involved in the process, 2,966 requirements gathered in total for smart home, and 8,115 entries of rating were gathered for the requirements. Here is an example:

TABLE I: EXAMPLE REQUIREMENTS DATA RECORD

User ID	5
Role	Worker
Feature	my smart home to be able to order delivery food by simple voice command
Benefit	I can prepare dinner easily after a long day at work
Application domain	Health
Tags	food, delivery, dinner, voice
Created at	2015/12/25 2:35
Show other	1
Duplicate of	0

IV. Our study

Our study adopted a semi-automated requirement clustering approach, by which the set of natural language requirements statements are clustered based on their linguistic similarities. After getting the requirements dataset, we applied a preprocessing procedure extracting the relevant data. Then a document clustering is performed using baseline clustering algorithms, such as, K-Means, Latent Dirichlet Allocation (LDA) and Biterm topic model. Then results in the subject topics for each individual requirement are assigned into the clusters. Figure 1 shows the overall experimental process. The following sub-sections will explain each step in more detail.

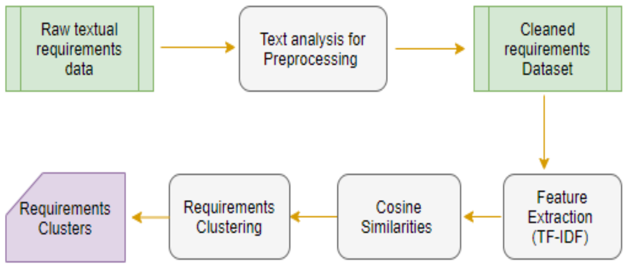
Raw textual requirements data
Text analysis for Preprocessing
Cleaned requirements Dataset
Feature Extraction (TF-IDF)
Cosine Similarities
Requirements Clustering
Requirements Clusters

Figure 1. Our proposed requirements data processing flow

A. Preprocessing

Before processing the textual requirements data, the following preprocessing steps were applied:

Tokenization: First of all, the set of requirements is tokenized based on the principle that textual requirements are represented as unordered collection of words. There are 34,657 tokens generated from the set of requirements.

Stop words removal: Stop words are those terms commonly used in English language as a grammatical component, but without specific meaning, e.g. "the", "we", "my", "or". We remove those words using stop words removal function of existing NLP toolkits (in our case, NLTK 3.3[1]).

Stemming: we use the stemming function of the NLP toolkits to replace derived words to their stem or root form.

Lower Case and Special Characters: All words in the text are converted to lower case, and special characters are also removed.

B. Feature Extraction

1) *Using TF-IDF:* The quality of the requirements statement is usually represented by the importance of words in that requirements statement. TF-IDF, is a straightforward approach which ranks words in a document according to their importance based on frequency. This feature is calculated from the sum of TF-IDF scores for each word in a requirements statement.

Term Frequency (TF) calculates how frequently a word occurred in a requirement. Since requirements have different length, it is possible that some terms occur more frequently in lengthy requirements as in shorter ones. To balance this, the value of TF is divided by the total number of requirements statements.

$$\mathrm{tf}(w_i, r_j) = \Sigma_{i,j=1..n,}\ f(w_i, r_j) / n \qquad (1)$$

Where $f(w_i, r_j) = 1$ if w_i *occurs* in r_i and 0, otherwise, w_i is the word being considered, r_j is the *j-th* requirements statement, n is the total number of requirements statement in the data set.

Inverse Document Frequency (IDF) measures the importance of a word in a requirement statement. Since in TF each word is considered of equal importance, there are certain words which occur more often in each requirement, which dilutes its importance. These words are known as stop words (is, the, are, at). Therefore, to normalize it, the-less-often used words are scaled up while the more frequently used words are weighed down with the following equation.

$$\mathrm{idf}(w_i) = \log n \,/\, |\{w_i \text{ occurs in } r_i,\ r_i \in R\}| \qquad (2)$$

where n is the total number of requirements, while $|\{w_i$ occurs in $r_i, r_i \in R\}|$, means the number of requirements in R that contains the word w_i.

TF-IDF: Multiplying equation (1) and (2) will give the composite weight of each word in each requirement.

$$\mathrm{tfidf}(w_i) = \mathrm{tf}(w_i) * \mathrm{idf}(w_i) \qquad (3)$$

2) *Extraction of Domain Objects and Operations:* In order to identify the domain objects and operations in the smart home dataset, we adopted the part-of-speech tagging. Then based on these objects and operations, we tried to find out the most frequently mentioned features in the dataset.

Nouns as objects: We used NLP toolkit library to identify all the nouns as objects from the dataset. In total 1,898 nouns are

[1] http://www.nltk.org

identified, amongst which top ranked are shown in Table II. To validate our results, we compare our identified objects with smart home vendor products in the online catalogue[1]. The key objects presented in smart homes, are fully covered in our experiment. Also, the identified objects are cross-validated against the tags information given by the requirements provider, together with the requirements, as shown in Table I. Table II shows the top 16 objects mentioned, are compared with the crowd defined tags, which crowd users identified when elaborating the requirements, like as shown in Table I . Some of the objects comes in groups, e.g. "food" comes with "fridge", "room" comes with "thermostat". The "time" object haven't appeared in vendor products, but it ranks fairly high in the experimental smart home requirements dataset. By checking the original dataset, we find that "time" appears in requirements statements in the preposition phrases such as, "at any time", "at the best time", "for some specific time", etc. This can be interpreted as "when" an operation has to be triggered, rather than an object. All the other objects present in the vendor products are also identified by our approach, but we have only displayed the 16 top ranked objects in the Table II.

TABLE II. OBJECTS AND THEIR FREQUENCIES

Ranking	Objects	Frequencies	Appear in general smart home products[1]?
1	room	181	Y
2	water	159	Y
3	door	159	Y
4	lights	148	Y
5	time	143	-
6	temperature	138	Y
7	system	129	Y
8	phone	116	Y
9	alarm	93	Y
10	food	88	Y
11	TV	86	Y
12	sensors	78	Y
13	windows	72	Y
14	music	67	Y
15	fridge	57	Y
16	thermostat	44	Y

Verbs as operations: By a similar approach, all the verbs are extracted from the dataset using part-of-speech tagging. In total, 350 verbs are identified, amongst which the top 15 operations are shown in Table III. The operations identified by our approach are validated through vendor products. Also, operations identified are cross-validated against user's defined tags. It is observed that 11 operations out of 15 conformed to the crowd defined tags and the four verbs not in the tags do not stand on its own as meaningful operations, which is shown with "-" symbol in Table III. An operation comes together with objects, e.g. "turn on/off light", "play music", "alert me for monthly energy usage", "monitor the sensor". These features are also detected by our approach in section V-C. The top operations listed in Table III, are also presented in vendor products.

TABLE III. OPERATIONS AND THEIR CORRESPONDING FREQUENCIES

Ranking	Operation	Frequency	Appear in tags?
1	Turn	264	Y
2	Alert	222	Y
3	Have	200	-
4	Know	154	-
5	Notify	115	Y
6	Tell	94	-
7	monitor	93	Y
8	Play	93	Y
9	detect	87	Y
10	Open	76	Y
11	Adjust	66	-
12	Control	62	Y
13	Lock	55	Y
14	Sense	44	Y
15	Shut	41	Y

Relations between objects and operations: To find association relationships between the identified objects and the control operations, we used the co-occurrence statistics. As in the smart home dataset, an object does not always appear immediately after an operation, there could be a few words between them. Therefore, in order to find its co-occurrence, we introduced the threshold range parameter λ. The value of λ is defined manually, we set λ to 10, to be inclusive, and by observing its occurrence patterns in the smart home dataset. For example, the requirement statement "notify me when a box …", here we have the operation "notify" while the object is "box", and the words between them are "me", "when", and "a". There are three words between the verb representing the operation and the noun representing the object, which is within the threshold range λ, so they are recorded as a co-occurred pair. In this way, we have got 3,800 unique object and operation pairs. The most frequently co-occurred pairs are shown in Table IV. We note that if operations and objects were synonymous, we manually aggregated them together, e.g. "alert/notify me when", "turn/switch off light". These co-occurred words are then clustered in section V-C, to get useful features using K-Means clustering.

TABLE IV. EXAMPLE VERBS + OBJECTS INSTANCES

Ranking	Verb	Frequency
1	alert/notify me when	31
2	keep track	14
3	alert me via a specific sound	10
4	turn/switch off light	9
5	send me a message	8
6	play (favorite/soothing) music	8
7	open and close doors	7
8	lock window and doors	6
9	turn on heat /cool	8
10	turn on laundry	3
11	preheat oven	2

C. Similar requirements identification

After finding the TF-IDF values for each word in the set of requirements, similar requirements are identified with respect to

[1] http://www.smarthomefargo.com/vendors/

every other requirement. To achieve this, we use cosine similarity function provided by the general-purpose machine learning tool SCIKIT-LEARN (Sklearn[1]), which compares each requirement to every other requirement and returns a value likeliness value between 0 and 1.

D. Requirements Clustering and Summarization

The clustering and summarization of requirements can be described as a multi-document summarization task where each requirement in the dataset are considered as an individual document. In general, multi-document summarization can either be extractive or abstractive [3]. In an extractive approach, specific keywords are selected, which are already presented in the data, to represent the entire data set. While in an abstractive approach, a summary is generated which represents the narrative of the entire dataset. In our case, we adopted the extractive approach. In the literature, multiple techniques have been proposed for text summarization, amongst which some popular algorithms used are LDA [10, 17, 18, 19], TF-IDF [4], K-Means [8, 22] and Biterm topic model [3]. We have applied all the above mention algorithm on the smart home dataset which are described below while TF-IDF are already describe in section IV-B.

1) Latent Dirichlet Allocation (LDA): LDA [20] comes under the umbrella of topic modeling algorithms. LDA is used to identify the associated topics of text documents using a probabilistic distribution algorithm. In our case, the textual documents are the smart home requirements collected from users. LDA algorithm involves a control variable referred as the number of topics, represented as *K*. We can change the value of *K,* which will display *K* topics on the probability distribution of words. We can also control the number of most popular words, to be displayed under each topic *K*. We have used LDA topic extraction function in SkLearn python library.

2) Biterm Topic Model (BTM): Biterm topic model is a topic modeling algorithm propose by Yan et al [21]. This algorithm is specialized for processing short text documents such as twitter, or microblogs. BTM takes preprocessed document as an input and return popular topics that commonly occur in the entire corpus of documents based on a probability matrix.

3) K-Means Clustering:
Cosine similarities and TF-IDF part are used in K-means clustering. After calculating the cosine similarities amongst the requirements, we use K-Means unsupervised clustering algorithm to group similar requirements. K-Means algorithm groups requirements on a given value of "*K*". For each cluster, a centroid is defined, around whom requirements distribute closely, the ones within close similarity distance to this centroid are assigned to the group. The similarity distance between requirements in different groups are further than the ones in the same group.

V. Analysis Results

We applied three different unsupervised clustering algorithms on the smart home requirements dataset to find similar requirements. K-Means algorithm gives more useful results as compared to LDA and Biterm topic model, which are explained below.

A. Preprocessing and Feature Extraction

Crowd users express their requirements in free-style English natural language, which need to be preprocessed before sending for the clustering process. We have 2,966 raw user's requirements, by tokenizing, we got 34,657 words. By further removing stop words, we got 18,457 actual words from the smart home dataset. TF-IDF from Sklearn is then applied on the actual words from the dataset to extract the features and got a feature matrix of (2966, 2206). After getting the *tf-idf* matrix, we can now measure its cosine similarity, in order to generate a measure of similarity between each requirement and all the other requirements in the corpus.

TABLE V. RESULTS OF K-MEANS CLUSTER

K	Cluster id.	No. of rqmts.	Keywords in each cluster
K=150	27	139	TV, alerts, automatic, based, cool
	3	27	control, temperature, rooms, purity, air
	6	4	Ability, tell, open, doors, close
K=120	23	253	Automatic, turn, smart, 's', dishwasher
	111	13	Weight, scales, store, bought, track
	64	4	Good, entertainment, kid, activities, understand
K=170	75	179	Automatic, thermometer, health, smart, rooms
	103	8	Help, dinner, need, monitor, know
	168	2	Scooper, pooper, robot, smart, zones

B. Results using K-Means clustering algorithm

As mentioned earlier, K-Means generates K clusters. In the literature [42], the elbow method is used to identify the optimal value of K. Therefore, we applied the elbow method on smart home dataset, where we get the optimum value of K=5. Unfortunately, this did not work well on our dataset because the number of requirements per cluster is still too many to represent useful information for software developers. By manual analysis, on an average, there are 20-25 crowd requirements in the smart home data set that are quite similar in semantics/context and are representing similar features. We experimented with multiple random values for K = {10, 30, 55, 100, 120,150,170}, to evaluate the performance of algorithm. Where we found that when K is set to 150, K-Means cluster generates 150 clusters and we get human interpretable results with this setting. When the average requirements per cluster are 20-25, each cluster obtained is quite manageable and contains similar requirements as in Table V. When we increase the K value to greater than 150, the requirements in the clusters increase, which becomes difficult to handle again. Similarly, when we use a K value smaller than 150, the results are degraded. The results for K=150, K=120 and K=127 are shown in Table V, we have chosen three random examples from the clustering for each value of K. Like when K=150, Cluster 27 contains 139 requirements, which is the

[1] http://scikit-learn.org/stable/index.html

TABLE VI. GROUPED SIMILAR REQUIREMENTS USING K-MEANS CLUSTER

Cluster No	Words per cluster	Requirements Per Cluster with priorities
36	Clean, Floor , Dirty, vacuum and robot.	my smart home to clean the floors regularly, A vacuum that will tell me when the floors are 100 percent clean, to have standard automatic floor cleaning, my dirty clothes clean without water and detergent powder, my vacuum cleaner to clean to floors at a particular time daily or as I Program, a home that knows when the floor is dirty and sends appropriate robots to clean it, the li tter box to be auto cleaned when dirty, a vacuum that automatically cleans the floor when it senses that it is dirty, a small robot that is integrated into the house to sense when the floor is dirty and to clean it and when necessary mop it
63	Solar, Panels, roof, power, smart	My smart house to have solar panels, my smart home to have dynamic solar panels which can be set just opposite to the sun, so lar panels to power the whole house, solar windows, Solar panels, a solar panel, solar system, a roof made of solar panels that f unctions as a roof and an energy source, an automatic solar panel, my smart home to be outfitted with solar panels, ability to au tomatically adjust solar panels throughout the day to get best energy production, solar roof, to control my solar panel with my s martphone, smart solar panels that adjust themselves, solar Sheeted floors/walls, have solar power,
137	Medicine, pills, reminder, cabinet,	My medicine pill holder to notify me about the pill that are going to expire, My medicine pill holder to notify me that new pills need to be purchased, my medicine cabinet to show me which medicines and drugs should not be taken together or may have e xpired, my medicine cabinet to remind me to give children medicine, my smart home to set alarms to remind me when to take medicine, a medicine box that includes an auto-dial options to re-order my birth control, a medicine cabinet, spoken reminders to take medicine at the right times, to have all the required medicines, a place to store medicine and have the dispensing be don e automatically based on the prescription

largest requirements cluster, cluster 3 contains 27 requirements represents the average cluster size, while smallest cluster is 6, which have only 4 requirements. Similarly, the results for K=120 and K=170 are shown in Table V.

Williams et al [2], conducted a pilot study with software developers, where they were asked to summarize tweets into most frequent topics. Later, the experts were asked if they think these words are enough for capturing the users' viewpoints. The experts preferred to see full tweets along with topics, because word-clouds usually lacks in context and structure. Keeping that in view, we also identified associated requirements with each cluster, shown in Table VI. We picked three random clusters from 150 clusters. It can be seen from the Table VI, that K-Means algorithm efficiently summarize the topics and based on these topics, similar requirements are grouped. With this result, developers have a clear understanding of crowd requirements, as topics along with related requirements are displayed.

C. Clustering Co-occurred features

We have found 3,800 co-relations between the objects and operations, in which many of the co-relations are similar from the linguistics aspects. Therefore, we applied K-Means clustering over the 3,800 co-relations in order to find out the similar co-relations and group them. We followed the steps proposed earlier in this paper, text preprocessing, calculating TF-IDF and finally calculating cosine similarities. Same procedure was followed for selecting optimum value of K as explained in section V-B. We have chosen different value of K = {10, 20, 30, 40, 50}. The results of the K-Means co-occurred clustering are shown in Table VII, we have picked randomly three clusters for K=50. It can be seen from the Table VII, that K-Means algorithm grouped similar features efficiently and also improved the results shown in Table III by removing the stop words and getting interesting features, which will help requirement analyst to focus on highlighted requirement.

TABLE VII. SIMILAR FEATURES USING K-MEANS, WHEN K=50

Cluster No	Words per Cluster	Features per cluster
3	Energy, usage, mo nitor, consumption	Save energy, monitor my energy, monitor my energy usage, reduce energy, analyze energy, monitor energy
7	Turn, time, temper ature, television, st ove	Turn off appliances, turn on music, turn o n the TV, turn on at night, turn off or dim, turn off my stove, turn off the television, t urn off the oven, turn of appliances
48	Control, temperatu re, remote, device, regulate	Decrease temperature, regulate temperatur e, control temperature and spray, regulate i deal temperature, control the temperature, control tv, have a remote control for wind ows, control all devices, control tv consum ption

D. Results using LDA

In the literature [10, 17], LDA is widely used for summarization of requirements. We also applied LDA algorithm on the smart home dataset in order to get useful topics. Namely, software features which can be easily understood by software analysts as potential user's requirements, can be generalized as requirements from the crowd. But we did not get useful results from this approach, even using different parameters with LDA algorithms, such as *n-gram*. Requirements in the smart home dataset are gathered on the principle of novelty. LDA helps to find similar topics on a probability distribution, but in the smart home dataset features are unique which leads to un-interpretable results (topics obtained do not express meaningful users requirements/features). Williams et al [2] proposed in their pilot study that developers are interested in full requirements along with identified topics so that user requirements are clearly understood.

E. Results using Biterm Topic Model (BTM)

The Biterm topic model algorithm takes preprocessed text as inputs, therefore we removed stop words from the smart home dataset and then run the smart home dataset with the implementation BTM provided by Yan et al [21]. BTM takes K as input to the displayed topics. We use multiple values of K to get the related topics. Similar to the LDA algorithm, we did not get useful topics that is generalizable for understanding crowd requirements.

VI. Related work

A. Crowd-based RE and Crowd Feedback

In recent years, there is a growing research interest in the crowd-based RE. The disruptive rise in the use of Internet, mobile and social media applications gives rise to this research area. Many researchers and organizations are contributing to crowd-based RE [24, 25]. Groen et al [23] proposed a novel Crowd-based RE technique for gathering user feedbacks.

Kanchev et al performed details analysis on the usage of social media in RE and how requirements engineers can benefit from it [26]. Khalid et al [27] conducted a qualitative study that assessed apps stores used as a source by users to report problems in form of reviews on apps. They identified 12 complaints types by comments reported by users. Pagano and Maalej [28] provide a study on users' feedbacks from app stores, which in turn can be used as a source for eliciting requirements. Hosseini et al [29], Srivastava et al [30], Maalej et al [31] and Almalik et al [32], proposed mechanisms for gathering continuous user feedbacks from a large community of users so that later this feedback can also be used by the RE staff to better identify user requirements. The approach proposed in this paper is based on the principle of Crowd-based RE [23], the dataset we used for the experimentation is collected through crowd-based RE which represents actual user's requirements.

B. Mining User Feedbacks from App Stores

End users are growing in terms of group size for online mass market-oriented software, due to which volume of end users' feedbacks or reviews on app stores increases exponentially. So far, much work has been done in user feedback mining for features extraction. Feedback mining are categorized as classification, grouping and ranking.

Groen et al [13] develop 16 patterns to identify non-function requirements while Johann et al [34] used 18 part of speech patterns and 5 sentence patterns to identify common features by comparing app description and users' reviews. Guzman et al [17] [36] used sentiments analysis to identify user's features. Maalej et al [14], Panichella et al [33] implemented supervised machine learning algorithms to classify users' feedbacks into improvements, bugs and others while Bakiu et al [35] uses machine learning to identify usability requirements. Harman et al [15], Gu et al [37], Andrea et al [38], Villarroel et al [39] and Chen et al [19] used both supervised and un-supervised learning to classify and rank application reviews.

C. Mining User Feedbacks from Twitter

Twitter is one of the most popular social networks nowadays. Twitter and user reviews are similar in terms of their short length, and free natural language text. End users usually use twitter as a media for expressing their experiences with software. Thus, recently researchers have analyzed twitter data to get that users feedback and used it as a source for identifying new requirements or bugs. Williams et al [4] used machine learning to classify tweets into technical informative tweets and then group them using TF-IDF. Guzman et al [3] proposed a method to classify twitter feeds, then summarize it using Biterm topic modeling approach and finally rank them using weighted function. Earlier Guzman et al [40] conducted an exploratory study on the importance of tweeter as a source for requirement engineers and used SVM and decision tree algorithm to classify different stakeholders. In contrast to all of the above approaches, we identify possible objects, operations and their relations which are main components of software features. Using these objects and operations we constructed goal model semi-automatically to give useful insights to software experts and end-users.

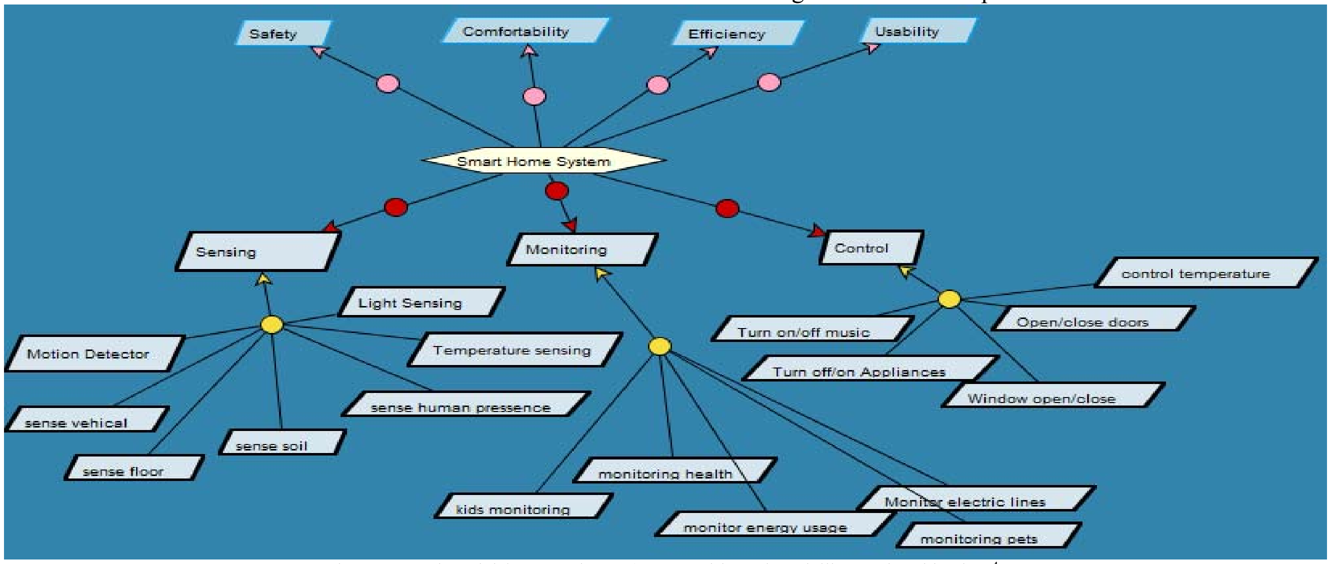

Figure 2. Goal model for smart home (Drawn with goal modelling tool - Objectiver[4])

[4] http://www.objectiver.com/index.php?id=4

VII. Goals and Features Modeling

After analyzing the features extracted from objects and operations using co-occurrence statistics, identified features can be modeled using goal modeling, as shown in the Figure II. Some of the key features of smart homes extracted are modelled using goal modeling technique in order to have a clear overview. The four softgoals of the smart home system are extracted from the benefit column (representing goals of the requirements) in the smart home dataset using POS tagging, which are safety, comfortability, efficiency and usability. These softgoals can be achieved by three main functionalities sensing, monitoring and control, which are derived from the extracted features manually as these are the most dominant functionalities in the smart home dataset. Between the goals and main functionalities is the main agent of smart home system. Sensing functionality is further decomposed into light sensing, motion detector, temperature sensing, sense vehicle, sense human presence, sense soil and sense floor. Monitoring functionalities are composed of kids, health, energy usage, electric lines and pets, while control functionality of smart home is further divided into turn on/off music, control temperature, open/close doors, turn off/on appliances and window open/close. These are the main features containing both objects and operation of smart home, identified by both data analysis approach, partial list of which are shown in Table V, which are confirmed by features offered by the vendor products.

VIII. Limitations and Threats to validity

Despite the encouraging results, a list [41] of threats to validity is discussed below.

Threats to conclusion validity: the analysis to the experimental results may be influenced by the researchers as they are looking for positive evidence for the hypothesis, and hence unable to reject the hypothesis unbiasedly.

Threats to internal validity: the dataset used in our experiment is collected from the crowd users using AMT, where specified end-users took part in curating this dataset. This may lead to instrumentation threats. This may be overcome by looking into more data sources, such as: Twitter, App Stores and User Forums, to check the performance of our proposed approach.

Threats to construction validity: the design of our experiment is missing an important aspect, i.e. lacking an evaluation of our semi-automated approach against human-generated summaries. We might have missed some other features of machines learning or algorithms due to lack of knowledge, e.g., we can use semantic similarities instead of cosine similarities to check the performance of the proposed approach, etc.

IX. Conclusion and future work

This paper presents the results from an experimental study on an available requirements dataset of smart home. The aim of this paper is to explore how well a semi-automated data analysis approach can help analyzing users generated textual requirements data. End users of mass market driven software generate large volume of user data, which is difficult to be analyzed manually. In this paper, we identified linguistically similar requirements by applying multiple clustering algorithms, and we have got some usable results with K-Means algorithm. We also extracted domain objects and operations from the dataset in order to find out frequent association relationships between them, and we use co-occurrence statistics, to identify similar features. Most prominent features are then modeled through goal modeling technique for better understanding of end-users' requirements. The identified features are then cross checked with additional information from the data set and confirmed that the majority of featured being identified from the user requirement data set are easily mapped to the vendor product features. There are a few objects and operations not confirmed, by checking closely, we find that they belong to one of the following cases: (1) as the current approach only considers the most simplified statement patterns of "operation + object", where more complex and refined requirements statements with conditions are not yet handled, which shall be considered in future. (2) There are mixing levels of abstractions involved in the user statements, for example, conditions related to time dimension, statements tackling different system implementation preferences, such as, turn on the light at night, versus turn on the light when it is getting dark; notify me, versus send me a message, or call the police, or alert the fire department, these are similar functions to be implemented with various design alternatives. In summary, the experiment tells us that based on the current results, the semi-automated analysis approach can help us identify domain objects, operations, and list of potential features. Needless to mention, manual refinement and analysis shall be conducted to throughout the process for precision.

In the future, the proposed approach can be extended in three directions: (1) Combined use of the data-driven approach with top-down goal-oriented requirements modelling for cross-validation of requirements from different sources, where data-driven approaches are used for crowd-based end user requirements elicitation, and goal-modelling approach is used for domain experts and developers with extensive knowledge about the domain and system under consideration. (2) Construction of a domain requirements repository and knowledge base where text corpus, entities, relations, constraints and model fragments are managed and updated with active learning ability, such as generating questions to users to clarify confusing requirements. (3) Further improve the design of the empirical study and yield more system evidence to the above-mentioned conclusions.

References

[1] P.K. Murukannaiah, N. Ajmeri and M. P. Singh, "Towards automating Crowd RE", IEEE 25th RE Workshops, 2017, pp.512-515. https://github.com/crowdre/murukannaiah-smarthome-requirements-dataset

[2] P. K. Murukannaiah, Nirav Ajmeri, Munindar P. Singh: Acquiring Creative Requirements from the Crowd: Understanding the Influences of Personality and Creative Potential in Crowd RE. RE 2016: 176-185.

[3] E. Guzman, Mohamed Ibrahim, Martin Glinz: A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution. RE 2017: 11-20

[4] G. Williams, A. Mahmoud: Mining Twitter Feeds for Software User Requirements. RE 2017: 1-10.

[5] G. M. Kanchev, P. K. Murukannaiah, A. K. Chopra, P. Sawyer: Canary: An Interactive and Query-Based Approach to Extract Requirements from Online Forums. RE2017: 470-471.

[6] C. Li, Liguo Huang, Jidong Ge, Bin Luo, Vincent Ng, Automatically classifying user requests in crowdsourcing requirements engineering, JSS, 138, 2018, 108-123.

[7] L. Shi, Celia Chen, Qing Wang, Barry W. Boehm: Is It a New Feature or Simply "Don't Know Yet"?, On Automated Redundant OSS Feature Requests Identification. RE 2016: 377-382

[8] J.C Huang, H. Dumitru, C. Duan and C.C. Herrera, "Support for managing feature request in open forums", Communications of the ACM, Vol. 52 No. 10, pp. 68-74, 2009.

[9] G. M. Kanchev, Amit K. Chopra: Social media through the requirements lens: A case study of Google maps. CrowdRE@RE 2015: 7-12

[10] L.V. Carreno and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution", 35th ICSE'13: 582-591.

[11] D. Pagano and W. Maalej," User Feedback in the AppStore: An Empirical Study", RE 2013: 125-134.

[12] E. Guzman, Rana Alkadhi, Norbert Seyff: A Needle in a Haystack: What Do Twitter Users Say about Software? RE 2016: 96-105

[13] E. C. Groen, Sylwia Kopczynska, Marc P. Hauer, Tobias D. Krafft, Jörg Dörr: Users - The Hidden Software Product Quality Experts?: A Study on How App Users Report Quality Aspects in Online Reviews. RE 2017: 80-89.

[14] W. Maalej and H. Nabil, "Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews," RE 2015, 116–125

[15] M. Harman, Y. Jia, Y. Zhang, "App store mining and analysis: MSR for app stores", MSR 2012.

[16] E. Guzman, M. Ibrahim, M. Glinz, "Prioritizing user feedback from Twitter: A survey report", IEEE/ACM 4th international workshop on crowdsourcing in software engineering (CSI-SE), 2017: 21-24.

[17] E. Guzman, W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews", RE 2014, pp 153-162.

[18] E. Khabiri, J. Caverlee, and C. Hsu, "Summarizing user-contributed comments," in *AAAI Conference on Weblogs and Social Media*, 2011, pp. 534–537.

[19] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao and B. Zhang, "AR-Miner: Mining informative reviews for developers from mobile app marketplace", ICSE 2014.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. The Journal of Machine Learning Research, 3:993–1022, Mar. 2003.

[21] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in Proc. of the International Conference on World Wide Web, 2013

[22] K. Shetty, J. S. Kallimani, "automatic extractive text summarization using K-Means clustering", international conference on electrical, electronics, communication, computer and optimization techniques (ICEECCOT), 2017, pp 1-9.

[23] E. C. Groen, B. Seyff, R. Ali, F. Dalpaiz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini, M. Stade, "The Crowd in Requirement Engineering- The landscape and Challenges", IEEE Software, 34(2), 2017, pp. 44–52.

[24] M. Stade, M. Oriol, O. Cabrera, F. Fotrousi, R. Schaniel, N. Seyff, O. Schmidt, "Providing a User Forum is not enough: First Experiences of a Software Company with CrowdRE", IEEE 2nd international Workshop on Crowd-Based Requirement Engineering (CrowdRE), 2017, pp.164-169.

[25] E.C. Groen, J. Doerr, and S. Adam, "Towards Crowd-Based Requirements Engineering: A Research Preview," REFSQ 2015, pp. 247–253.

[26] G. M. Kanchev, Amit K. Chopra: Social media through the requirements lens: A case study of Google maps. CrowdRE@RE 2015: 7-12.

[27] H. Khalid, E. Shihab, M. Nagappan and A. E. Hassan, "What do mobile users complain about?", IEEE Software, 2015, pp 70-77.

[28] D. Pagano and W. Maalej, "User Feedback in The Appstore: An Empirical Study," in Proceedings 2013 21st IEEE RE 2013, pp. 125 – 134

[29] M. Hosseini, E.C. Groen, A. Shahri, R. Ali, "CRAFT: A CRowed-Annotated Feedback Technique", IEEE 25th RE Workshops, 2017, pp. 170-175.

[30] P. K. Srivastava and R. Sharma, "Crowdsourcing to Elicit Requirements for MyERP Application," IEEE 1st international Workshop on Crowd-Based Requirement Engineering (CrowdRE),2015, pp. 31-35.

[31] W. Maalej, H.-J. Happel, and A. Rashid, "When Users Become Collaborators: Towards Continuous and Context-Aware User Input," 24th OOPSLA 2009, pp. 981–990.

[32] M. Almaliki, C. Ncube, and R. Ali, "Adaptive Software-Based Feedback Acquisition: A Persona-Based Design," RCIS 2015, pp. 100–111.

[33] S. Panichella, A. Di. Sorbo, E. Guzman, C.A. Visaggio, G. Canfora and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution", SME 2015,pp-1-10

[34] T. Johann, Christoph Stanik, Alireza M. Alizadeh B., Walid Maalej: SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews. RE 2017: 21-30

[35] E. Bakiu, Emitza Guzman: Which Feature is Unusable? Detecting Usability and User Experience Issues from User Reviews. RE Workshops 2017: 182-187

[36] E. Guzman, O. Aly and B. Bruegge, "Retrieving diverse opinions from app reviews", ESEM 2015,pp-21-30.

[37] X. Gu and S. Kim, "what parts of your apps are love by users?", IEEE/ACM ASE 2015, 760-770

[38] A. Di. Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora and H. Gall, "What would users change in my app? Summarizing app reviews for recommending software changes", ACM SIGSOFT FSE 2016, 499-510.

[39] L. Villarroel, G. Bavota, B. Russo, R. Oliveto and M. Di. Penta, "Release planning of mobile apps based on users reviews", ICSE 2016, 14-24.

[40] E. Guzman, Rana Alkadhi, Norbert Seyff: A Needle in a Haystack: What Do Twitter Users Say about Software? RE 2016: 96-105.

[41] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer 2012.

[42] M.A. Syakur, B.K. Khotimah, E.M.S Rochman and B.D. Satoto, "Integration K-Means Clustering Method and Elbow Method For Identification of The Best Customer Profile Cluster", IOP Conf. Series: Materials Science and Engineering, 2018.