CLEI ELECTRONIC JOURNAL, VOLUME 22, NUMBER 1, PAPER 1, MONTH April 2019

Do consumers talk about the software in my product? An Exploratory Study of IoT Products on Amazon

Kamonphop Srisopha, Pooyan Behnamghader, Barry Boehm
University of Southern California, Department of Computer Science,
Los Angeles, USA, 90007
{*srisopha, behnam, boehm*}*@usc.edu*

Abstract

Consumer product reviews are an invaluable source of data because they contain a wide range of information that could help requirement engineers to meet user needs. Recent studies have shown that tweets about software applications and reviews on App Stores contain useful information, which enable a more responsive software requirements elicitation. However, all of these studies' subjects are merely software applications. Information on system software, such as embedded software, operating systems, and firmware, are overlooked, unless reviews of a product using them are investigated. Challenges in investigating these reviews could come from the fact that there is a huge volume of data available, as well as the fact that reviews of such products are diverse in nature, meaning that they may contain information mostly on hardware components or broadly on the product as a whole. Motivated by these observations, we conduct an exploratory study using a dataset of 7198 review sentences from 6 Internet of Things (IoT) products. Our qualitative analysis demonstrates that a sufficient quantity of software related information exists in these reviews. In addition, we investigate the performance of two supervised machine learning techniques (Support Vector Machines and Convolutional Neural Networks) for classification of information contained in the reviews. Our results suggest that, with a certain setup, these two techniques can be used to classify the information automatically with high precision and recall.

Keywords: Passive Crowdsourcing, Internet of Things, User Reviews, Text Classification, Software Evolution

1 Introduction

Online retail stores provide a convenient platform for producers to compete and sell their products. They also provide consumers with the ability to compare a variety of options and purchase the one that best fits their budgets and needs. In addition, consumers can share their experience of using a product through writing a review after purchase. This helps other consumers to have a better understanding of the quality of the products.

Not only do reviews help consumers make more informed decisions, but it also helps manufacturers get feedback on their products so that they can improve them over time. In fact, reviews contain a wealth of information that can be used to create new requirements. The idea of mining user reviews to improve the quality of the product is a form of "Passive Crowdsourcing." According to [1], Crowdsourcing "represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call." Passive Crowdsourcing simply means harnessing the power of the crowd without their knowledge.

Mining app store reviews to improve the quality of software has already gained the attention of researchers [2]. In comparison with user reviews of a software application, user reviews of an IoT product are not exclusively about software. They can be about different product elements, such as software, hardware, customer service, etc. In particular, they contain information such as subjective product evaluations (on hardware, software, or the product in general), personal experiences, problems encountered, product descriptions, and users' visions on how the product should be. As a result, reviews of IoT products are more diverse. If such information in this huge volume of data is detected and classified correctly and efficiently, different stakeholders, such as software engineers, hardware engineers, etc. can better meet user expectations and needs, thus accelerating product or software evolution processes.

Motivated by these observations, we conducted an exploratory study by analyzing 1491 verified purchase reviews (7198 review sentences) of 6 IoT products from Amazon to investigate how IoT product reviews can be categorized, whether enough software related information exists, and how much of this information is useful for software developers and requirement engineers. In addition, we answered what software quality characteristics users mention the most as well as studied if there are any patterns in the reviews with regards to the ratings. Finally, we studied the extent to which two supervised machine learning techniques (SVM and CNN) can be used to classify information in the dataset automatically, and identified configurations that can effectively improve the classification performance.

The rest of the paper is organized as follows. We formally define our research problems and describe data collection in Section 2. In Section 3, we describe the procedure to answer each research question and report the results. Thereafter in Section 4, we present factors that may affect the validity of our study. We discuss the related literature in Section 5. Finally, in Section 6, we conclude the paper and outline plans for future research.

2 Research Design

2.1 Research Questions

To guide our research, we formulate the following research questions:

RQ1: How can IoT product reviews be categorized?

- *Motivation.* As aforementioned, IoT product reviews can contain a wide range of information. This information includes, but is not limited to: subjective product evaluations (about hardware, software, or the product as a whole), personal experiences, problems encountered, product descriptions, and users' visions on how the product should be. In this question, we investigate what types of information are in the reviews.

RQ2: How much information in the studied product reviews is relevant to software engineers?

- *Motivation.* Once the taxonomy is defined in RQ1, the natural next step is to manually examine the reviews and to investigate the distribution of each category, *i.e.*, how often each category occurs. Specifically, we aim to find out how much of this data can help to enable a more responsive software requirements elicitation and software evolution process.

RQ3: How effective is a machine learning approach in classifying the reviews?

- *Motivation.* Sifting through each review manually to find the right information is a laborious task. Since SVM and CNN are often used for classification tasks, we investigate the extent to which they can be used to classify an unseen review based on the taxonomy previously defined in RQ1 automatically. In addition, we aim to find configurations that will improve the performance of the classifiers.

RQ4: How often does each software quality characteristic occur in the software related sentences?

- *Motivation..* In this question, we map one or more software quality characteristics defined by ISO/IEC 25010 [3] to a software related sentence found in RQ2. We aim to investigate how often does each software quality aspect occur in the studied product reviews. We also discuss the nature of our results.

RQ5: Are there any patterns in the studied product reviews with regards to the user ratings?

- *Motivation.* This RQ aims to investigate whether the rank orders of category differ with regards to the rating and whether each category and rating are independent or related to one another. This could help us understand, for example, whether 1-star rating reviews contain more problem discovery sentences than 5-star rating reviews or not.

2.2 Data Collection

To conduct our analysis, we sampled 1491 verified purchase reviews from 6 IoT products from Amazon. The data was collected on September 22, 2017 through a web crawling tool we developed. Table 1 describes the characteristics of the 6 IoT products we considered.

Table 1: Details of the studied products

Product name	ASIN	Domain	Total Reviews
Amazon Echo Dot	B01DFKC2SO	Smart Home	54230
Fitbit Blaze	B019VM3F2M	Smart Watch	7349
GoPro Hero4 Silver	B00NIYJF6U	Action Camera	2327
PlayStation 4	B01LRLJV28	Gaming Console	3135
Pebble Time	B0106IS5XY	Smart Watch	1577
Amazon Echo Show	B01J24C0TI	Smart Home	2565

To mitigate some threats to the external validity, we selected products from various domains: smart home, smart watch, gaming console, and action camera. It is important to note that on Amazon, a user can subjectively give a star rating on a scale of 1 to 5. However, as far as we know, there is no clear cut definition to what each star in a five-star rating system means. Furthermore, there is no limit on how many characters a user can write for a review. Therefore, as to eliminate some bias in the dataset, we sampled 50 reviews per each star rating of each product (total of 250 per product) where each review consists of no more than 20 sentences. We used NLTK[1]'s sentence tokenizer to split a review into a list of sentences. However, at the time of our data collection, 2-star reviews of PlayStation 4 and of GoPro Hero4 Silver contain less than 50 reviews that meet the aforementioned criteria. In this case, we only fetched reviews that meet the criteria from them, 46 and 45 reviews respectively. Table 2 shows the number of sentences for each star rating of each product. For replicability, our dataset is publicly available here[2].

Table 2: Number of sentences per star-rating of each product

#	Product	Star rating					Total
		1-star	2-star	3-star	4-star	5-star	
1	Amazon Echo Dot	155	282	280	385	354	1456
2	Fitbit Blaze	234	290	313	361	350	1548
3	GoPro Hero 4 Silver	228	196	165	278	357	1224
4	PlayStation 4	134	183	146	99	158	720
5	Pebble Watch	167	189	189	267	336	1148
6	Amazon Echo Show	235	254	149	204	260	1102
						Total sentences:	7198

3 Methodology and Results

3.1 RQ1

Procedure:

In this section, we discuss the procedure we used to answer **RQ1: How can IoT product reviews be categorized?**

We started by searching past literature for similar efforts on analyzing user reviews on IoT products. However, what we found was that most past literature limits its scope by focusing on analyzing user reviews of mobile applications on App Stores, such as Google Play or the Apple Store [4][5], and tweets on Twitter [6]. As a result, user reviews that may contain information on firmware, operating systems, or embedded software are not explored, unless reviews of a product using them are investigated. In doing so, we cannot assume that the majority of review sentences are software related. Therefore, a taxonomy defined by most past literature would not work.

Top-level taxonomy We conducted three *content analysis* sessions [7] in order to see what information or pattern was contained in the reviews. Upon initial data inspection involving a few review samples, the

[1] http://www.nltk.org/
[2] https://goo.gl/M6BcB1

first author came up with a rudimentary taxonomy which consists of 7 categories: *User Background, Usage Scenario, Software Feature Request, Software Complaint, Software Praise, Hardware Complaint*, and *Hardware Praise*. This resulted in our first draft taxonomy. In the next step, we assigned 52 Master's level CS students, who were taking a graduate level software engineering course at USC, to analyze and categorize IoT product review sentences based on the first draft taxonomy as the course's requirements elicitation activity. Note that students could suggest a category if they felt that a review sentence cannot be classified into any categories in the first draft taxonomy. The responses were gathered and new categories suggested by students were noted. In the third step, a team of 4 Master's student, who were not part of the students in the second step, and one PhD student, the first author, validated categories suggested in the second step by merging similar categories together and redefining them. The taxonomy resulting from this step consisted of 8 top-level categories, shown in Table 3. Additionally and initially, two of the top-level categories (Hardware and Software) had 5 additional sub-categories: *Praise, Complaint, Inquiry, Request*, and *Other.* However, through examining software categories that are relevant to software engineers from past literature discussed in detail in the next paragraph, we adopted its software taxonomy instead of our original 5 sub-categories. We adapted the sub-categories for Hardware category in the same way.

Software-level taxonomy In this level, we can assume that all review sentences are software related. Therefore, we could use the existing taxonomy from past literature. We used the "*Taxonomy for Software Maintenance and Evolution*" purposed by [8] as a starting point because they already evaluated the relevance of each software category for developers performing software evolution and maintenance tasks. However, we modified the definitions to better reflect the nature of our data. In addition, upon inspecting the data, we found that sentences expressing dissatisfaction with the software of a product often contained information describing why the user was dissatisfied. We believe this is relevant to software developers and requirement engineers in order to better meet user needs or expectations. We added these sentences to the Problem Discovery category. On the other hand, sentences expressing satisfaction conveyed little value for software evolution process [8]. In this case, we combined these sentences with the Information Giving category. Table 4 shows the 4 sub-categories for Software category used in our study.

Table 3: 8 Top-level categories and their respective definitions

#	Category	Definition
1	Hardware	Sentences mentioning a hardware component, a physical characteristic, or a physical part of the product. ***Note:*** can be broken down into 4 sub-categories*.
2	Software	Sentences mentioning a product's functionality, a set of capabilities, or GUI. ***Note:*** can be broken down. See Table 4.
3	General	Sentences describing the product as a whole, including persuading to and dissuading against buying or using the product.
4	User Background	Sentences discussing the background of the reviewer or the reviewer's character.
5	Other Product	Sentences referring to another product (*e.g.,* competitive products, accessories, products that should be used with this product, etc.).
6	Usage Scenario	Sentences mentioning a way to use the product, *i.e.,* a use case scenario.
7	Customer Service	Sentences recounting on the reviewer's experience with Amazon services (*e.g.,* package, return, shipment, etc).
8	Miscellaneous	Sentences that are not covered by or do not belong to another category.

*the definition of each sub-category is similar to the definitions provided in Table 4, but with Hardware oriented terms.

Table 4: Categories and their respective definitions for software-related sentences

Category	**Definition**
Information Giving	Sentences expressing satisfaction or informing other users or the sellers about the functionality of the product
Inquiry	Sentences related to attempts to obtain information or help from other users or the seller about the software
Feature Request	Sentences expressing ideas, suggestions, or needs for improving or enhancing the software or functionality of the product
Problem Discovery	Sentences expressing dissatisfaction or describing issues or unexpected behaviors with the software of the product

Summary of Results:

The resulted taxonomy consisting of 8 top-level categories and their respective definitions is shown in Table 3. The second level of the Software category is shown in Table 4. This layer is also applicable to the Hardware category with hardware-oriented definitions.

3.2 RQ2

Procedure:

In this section, we discuss the procedure and applied the taxonomy defined in RQ1 to answer **RQ2: How much information in the studied product reviews is relevant to software engineers?** In particular, this research questions can be broken down into two sub-questions:

- ***RQ2.1*** How much software-related information exists in the studied products reviews?
- ***RQ2.2*** How much of that information is useful to software engineers?

Manual Classification To create the ground truth dataset for our experiment, we manually classified each sentence in the dataset. A team consists of 4 Master's level CS students and one PhD students, the first author, accomplished this task. To mitigate some threats to the internal validity, we performed the following procedures:

First, we created a category guide, which consists of the precise details of the category definitions and examples. Second, we conducted a pilot run by randomly selecting 80 reviews from the dataset and instructing each annotator annotate the selected reviews. We calculated the inter-rater reliability to assess validity and subjectivity of this pilot run using Fleiss's Kappa since we have more than two annotators [9]. The inter-rater reliability value was 0.65, which is in the *"Substantial Agreement"* range according to [10], one level below *"Almost Perfect Agreement"* range. Third, we held several meetings before actual classifying the data to make sure that all annotators had a mutual understanding of the concept regarding the definitions of the categories in the taxonomy and the descriptions of the products. Fourth, we instructed each annotator to annotate at most 100 sentences per day as to avoid errors due to fatigue. Fifth, the first author individually performed a quality check after the dataset was labeled by randomly selecting 50% of the entire dataset. Any inconsistencies detected, *i.e.,* assigned categories that did not adhere to the definitions in the taxonomy for that sentence, were reviewed and fixed. Sixth, categories of software-level categories were classified by only one annotator. This total process spanned 4 weeks. Note that user review sentences are usually short and without a context clue, it sometimes is difficult to determine whether a sentence would belong to hardware or software category. We resolve this type of disagreement by flagging the review sentence and then discussing in order to reach the consensus among annotators. Table 5 shows examples of sentences and their corresponding categories from the manual classification process.

Table 5: Examples of sentences classified by using the taxonomy in RQ1

Review Sentence	**Top-level category**	**Second level**
"I developed a Class IV allergic reaction to the wrist band."	Hardware	Hardware Problem Discovery
"The companion android app is crap though - very slow to start and not very functional."	Software	Software Problem Discovery
"Stay away from this product!"	General	
"Alexa does not answer general questions as Google Home seems to."	Software, Other Product	Software Information Giving
"I got the camera on time but the LCD screen is not working."	Customer Service, Hardware	Hardware Problem Discovery
"The software needs work: i am constantly having to uninstall reinstall the pebble time app for android."	Software	Software Problem Discovery
"If I could set my alarm to wake me up M-F @ 7 am that would be a 5 star moment."	Software	Software Feature Request
"Bought this as a birthday present for my daughter a month ago and gave it to her last night."	User Background	

Summary of Results:

The results of the manual classification process for the top-level categories is shown in Table 6. It shows that most of the users often describe a product as a whole, without specifically mention hardware or software of a product (General - at 31.81%). It also shows that only 26.72% of the review sentences were found to be software related which means that the majority of review sentences are not directly applicable to software engineers - answering ***RQ2.1***. Table 7 shows the manual classification results of only software related sentences by applying the taxonomy in Table 4. It shows that only 8.79% (2.35% of all sentences) contain feedback on how to improve or enhance the software, 45.40% (12.12% of all sentences) contain information about problems users encountered or what made users dissatisfied with the software, and 1.09% (0.29% of all sentences) express an effort to acquire information about the software. This means that only **14.79%** of all sentences are directly applicable to software engineers - answering ***RQ2.2***.

Table 6: Manual Classification Results for Top-level categories

Category	**Count***	**Frequency %**
Hardware	1870	25.98%
Software	1923	26.72%
General	2290	31.81%
User Background	1711	23.77%
Other Product	549	7.63%
Usage Scenario	504	7.00%
Customer Service	199	2.76%
Miscellaneous	291	4.04%

*Each sentence can be classified into one or more top-level categories

Table 7: Manual Classification Results for software related sentences

Category	**Count**	**Frequency % (software only)**	**Frequency % (all sentences)**
Information Giving	860	44.72%	11.94%
Inquiry	21	1.09%	0.29%
Feature Request	169	8.79%	2.35%
Problem Discovery	873	45.40%	12.12%

3.3 RQ3

Procedure:

In this section, we discuss the background and procedure to answer **RQ3: How effective is a machine learning approach in classifying the reviews?** In particular, we can break this research question down into three sub-questions:

- ***RQ3.1*** What is the performance of each supervised machine learning technique (SVM vs CNN)?
- ***RQ3.2*** What combination gives us the highest precision and recall?
- ***RQ3.3*** How generalizable is the model?

In contrast with multi-class classification where each sentence is assigned to only one label, in our top-level categories, each sentence can be classified into one or more categories in the top-level. Such a problem is referred to as multi-label classification.

Text preprocessing We preprocessed the reviews in the following orders: (1) utilizing sentence tokenizer provided by NLTK, a widely used natural language processing toolkit in Python, to breakdown reviews into sentences, (2) lowercasing the resulted sentences, and (3) tokenizing the words on space and punctuation, removing any non-English character.

Vector Space Model: We adopted tf-idf weighting scheme and word2vec to help with classification task.

Term frequency-inverse document frequency scheme (tf-idf) Tf-idf consists of two components: the term frequency and the inverse document frequency. The former considers the number of occurrences of a term in a document, while the latter takes into consideration the information on the frequency of a term in the entire corpus. In other words, the scheme reflects how important a term is to a document in a corpus [11]. This scheme has been widely used in information retrieval and text mining. With this scheme, each term is represented as a 1xD matrix where the columns (D) denote the tf-idf values of the term in each document. In our context, a document refers to a review sentence, while a term refers to each unique word in the corpus.

Word2Vec (W2V). This is based on the distributional hypothesis, which states that words that appear in a similar context tend to share similar meanings [12]. It learns to group similar words together in a vector space in an unsupervised and data-driven manner. For example, if trained on a large corpus of software user reviews, W2V learns that the word "crash" is semantically similar to "freeze", "hang", or "shut_down" as these words or phrases usually appear in a similar context. Similarity, it learns that the word "interface" is semantically similar to "layout" and "ui". Each word is represented by a high-dimensional real-valued dense vector. The algorithms to generate vector representations of words are described in detail in [13]. It has been proven that, with enough data and context, word2vec model can improve the performance of a classifier [14]. To harness the power of this model, we trained our word2vec model on 2.4 million Amazon reviews (over 12 million sentences) based on the dataset made available by [15]. Note that we applied the aforementioned steps for text preprocessing before training a word2vec model. We only used reviews from "Electronics" and "Apps for Androids" category.

Implementation:

Support Vector Machines We used Scikit-learn[3], a widely used machine learning library for Python. We adopted the method based on combining SVM with tf-idf and SVM with W2V to classify the information presented in the reviews. With SVM + W2V, we selected certain features according to the part-of-speech of words, meaning that we considered only nouns, verbs, adjectives, and adverbs in a given sentence. We then combined vectors of these words and averaged them. In addition, we leveraged binary relevance method as a strategy to transform each label into an independent binary classification problem and train one classifier for each label [16].

Convolutional Neural Network We used TensorFlow[4], an open-source library for machine intelligence, to implement our CNN model based on Kim's CNN non-static model [14]. To make our CNN multi-label compatible, we made changes to the architecture as follows: first, we used sigmoid activation function instead of softmax at the output layer; second, we used *sigmoid cross entropy with logits* as our loss function; and third, since the predicted outputs are a set of probabilities, we used a simple rounding function to convert probabilities into one's and zero's in order to evaluate the accuracy against the ground truth. For multi-class classification, we adopted the same CNN architecture without any modifications.

[3]http://scikit-learn.org/stable/modules/svm.html
[4]https://www.tensorflow.org/

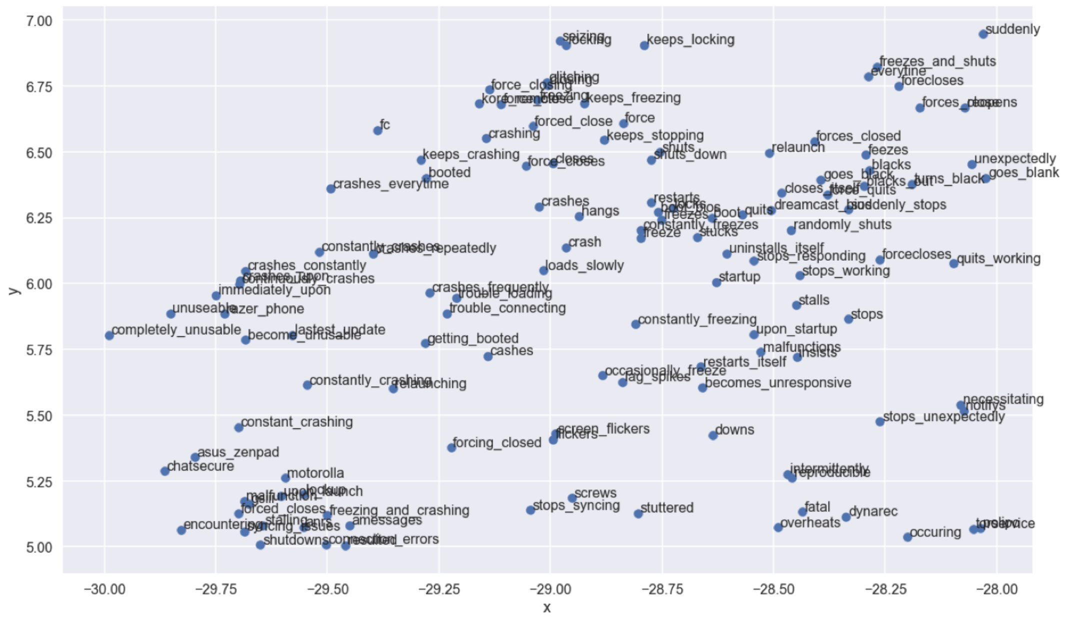

Figure 1: An example of a semantic space generated by a word2vec model that relates to the word "crash"

Metrics Evaluating the performance of multi-label classification classifier is more complicated than that of multi-class classification. In multi-label classification, predictions can be fully correct (exact match), partially correct (partial match), or fully incorrect (none match) [17]. Therefore, several evaluation metrics should be reported. In this paper, we reported the performance of our classifier on 2 label-based metrics: Macro-average Precision and Macro-averaged Recall. However, the complete results for Example-based metrics (Hamming Loss, Jaccard Similarity, etc) can also be found in our project repository.

Label-based measures evaluate each label separately and then averages over all labels [17].

$$Precision_{MacroAvg} = \frac{1}{n} \sum_{k=1}^{n} \frac{TP_k}{TP_k + FP_k} \tag{1}$$

$$Recall_{MacroAvg} = \frac{1}{n} \sum_{k=1}^{n} \frac{TP_k}{TP_k + FN_k} \tag{2}$$

where n is the number of labels, TP_k is the total number of instances that are correctly identified by the approach for the label k, FP_k is the total number of instances incorrectly identified by the approach for the label k, and FN_k is the total number of true instances that are not identified by the approach for the label k.

Results:

We decided to group category (3) User Background, (4) Other Product, (5) Usage Scenario, (6) Customer Service, and (7) Miscellaneous together because we believe that correctly identifying the information about the product (Hardware, Software, and General) itself matters the most. Another reason is that some of these categories occur much less frequently. A model that only returns predictions for categories that occur more frequently and never returns categories that occur less frequently will have high accuracy. However, this does not indicate that the model has a good predictive power as the model does not return any other label, except the most frequent one. This type of model is often used as a simple baseline model. For the sub-categories, we also decided to group category Inquiry with Problem Discovery because Inquiry occur much less frequently than any other category (Table 9). However, we decided not to group Feature Request category with any other category since this category is crucial for software engineers performing software evolution and maintenance tasks.

Furthermore, to prevent over-fitting and better test the generalizability of the model, we adopted two cross-validation measures. The results reported for the above metrics are based on these cross-validation measures.

10-fold cross validation We used the standard 10-fold cross validation to split the dataset in 10 folds and used 9 folds for training and 1 fold for evaluating. This process is repeated 10 times, rotating the training and testing folds.

6-fold product cross-validation Reviews of each product may contain words and jargon only applicable to the product. That means that term-features that work for one product may not work for others. We conducted a cross-product validation, a method previously proposed by [18]. In particular, we divided the dataset into 6 folds where each fold represents reviews of each product. We trained the classifier on 5 products reviews and tried to predict the classification of the reviews in the remaining product. The process is repeated 6 times, rotating the training and testing folds.

Table 8: Classification performance on the top level categories

Product#	**6-fold Product Cross Validation**																							
	SVM + tf-idf*								SVM + w2v‡								CNN + w2v‡							
	HW		SW		GN		OT		HW		SW		GN		OT		HW		SW		GN		OT	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
1	0.61	0.53	0.75	0.52	0.65	0.56	0.65	0.63	0.63	0.60	0.75	0.58	0.70	0.56	0.73	0.64	0.70	0.43	0.75	0.56	0.76	0.51	0.68	0.66
2	0.75	0.59	0.71	0.55	0.62	0.60	0.66	0.51	0.76	0.61	0.74	0.61	0.72	0.62	0.73	0.55	0.79	0.56	0.75	0.62	0.70	0.53	0.70	0.54
3	0.75	0.59	0.53	0.46	0.68	0.47	0.64	0.65	0.76	0.65	0.53	0.59	0.75	0.39	0.67	0.68	0.78	0.62	0.59	0.53	0.77	0.46	0.78	0.52
4	0.70	0.35	0.46	0.43	0.67	0.53	0.68	0.56	0.80	0.44	0.51	0.43	0.72	0.58	0.70	0.73	0.85	0.34	0.53	0.45	0.70	0.54	0.72	0.67
5	0.83	0.66	0.71	0.65	0.72	0.58	0.57	0.53	0.83	0.65	0.74	0.60	0.75	0.61	0.64	0.58	0.85	0.64	0.70	0.71	0.75	0.61	0.64	0.52
6	0.71	0.60	0.75	0.56	0.66	0.48	0.59	0.58	0.77	0.57	0.78	0.66	0.77	0.57	0.66	0.65	0.74	0.62	0.82	0.61	0.72	0.62	0.70	0.57
P(MA)	0.67								0.71								0.73							
R(MA)	0.55								0.59								0.56							
10 fold Cross Validation																								
	0.77	0.63	0.72	0.60	0.69	0.59	0.65	0.60	0.77	0.65	0.74	0.64	0.75	0.59	0.71	0.64	0.80	0.62	0.77	0.61	0.73	0.61	0.72	0.60
P(MA)	0.71								0.74								0.76							
R(MA)	0.60								0.63								0.61							

HW = Hardware; SW = Software; GN = General; OT = Other; P = Precision; R = Recall; P(MA) = Macro Average Precision; R(MA) = Macro Average Recall; * included stop words; ‡ trained on the combination of 2.4 millions Amazon reviews with our dataset

Table 9: Classification performance on software related categories

Product#	**6 fold Product Cross Validation**																	
	SVM + tf-idf*						SVM + w2v‡						CNN + w2v‡					
	FR		IG		PD		FR		IG		PD		FR		IG		PD	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
1	0.73	0.41	0.71	0.58	0.64	0.81	0.74	0.49	0.77	0.60	0.68	0.87	0.87	0.22	0.75	0.53	0.62	0.90
2	0.55	0.35	0.73	0.75	0.72	0.72	0.58	0.41	0.78	0.87	0.83	0.74	1.00	0.24	0.73	0.88	0.83	0.70
3	0.62	0.56	0.78	0.65	0.72	0.83	0.75	0.33	0.84	0.75	0.78	0.90	0.00	0.00	0.79	0.62	0.68	0.88
4	0.33	0.33	0.59	0.61	0.79	0.78	0.33	0.33	0.79	0.67	0.81	0.88	0.00	0.00	0.82	0.70	0.82	0.91
5	0.55	0.60	0.82	0.65	0.58	0.78	0.40	0.40	0.87	0.64	0.57	0.83	1.00	0.20	0.88	0.65	0.56	0.86
6	0.81	0.41	0.59	0.79	0.72	0.64	0.84	0.38	0.61	0.85	0.80	0.71	0.90	0.13	0.50	0.84	0.74	0.58
P(MA)	0.67						0.71						0.70					
R(MA)	0.63						0.65						0.55					
10-fold Cross Validation																		
	0.76	0.47	0.74	0.76	0.75	0.78	0.72	0.47	0.75	0.76	0.75	0.80	0.79	0.26	0.71	0.80	0.76	0.75
P(MA)	0.75						0.74						0.75					
R(MA)	0.67						0.68						0.60					

FR = Feature Request; IG = Information Giving; PD = Problem Discovery; P = Precision; R = Recall; P(MA) = Macro Average Precision; R(MA) = Macro Average Recall; * included stop words; ‡ trained on the combination of 2.4 millions Amazon reviews with our dataset

The performance results for classifying top-level categories in Table 8 show that incorporating w2v improved precision and recall of SVM in both 6-fold and 10-fold. However, the performance of CNN + w2v and SVM + w2v are comparable. The 6-fold cross validation results show that the SVM + w2v and CNN + w2v generalized well, *i.e.,* their performances did not drop significantly as SVM + tf-idf. Table 9 shows the performance of different methods in classifying software sentences into different software categories. Since only a small quantity of Feature Request sentences are in the dataset, the performance of CNN dropped significantly. In fact, for product 3 (GoPro Hero4 Silver) and 4 (PlayStation 4), CNN did not classify any sentence in the Feature Request category. SVM + w2v's performance, on the other hand, did not drop as significantly as CNN + w2v from 10-fold to 6-fold product cross validation. This implies that the combination generalized well across different products. Surprisingly, SVM + tf-idf outperformed CNN + w2v on classifying software related sentence (R(MA) for 10-fold). However, with more software related sentences, we believe that CNN would perform as well as or slightly better than SVM.

3.4 RQ4

Procedure:

In this section, we discuss the procedure we used to answer **RQ4: How often does each software quality characteristic occur in the software related sentences?**

To the best of our knowledge, mapping user review sentences to software quality characteristics has rarely been explored in literature. Nevertheless, there exists an automatic tool such as SURF, proposed by Di Sorbo et al. [19], that can classify software related sentences into one of the twelve "Topic Clusters" (e.g., Contents, Pricing, GUI, Security, Download, etc.). However, only three of the twelve clusters (GUI, Function, Security) are to some extent related to software quality aspects. In addition, an automatic tool can produce false positives (i.e., misclassification) which can affect the validity and outcome of the results if used. In this case, we had to manually classify each software related sentence into one or more software quality characteristics.

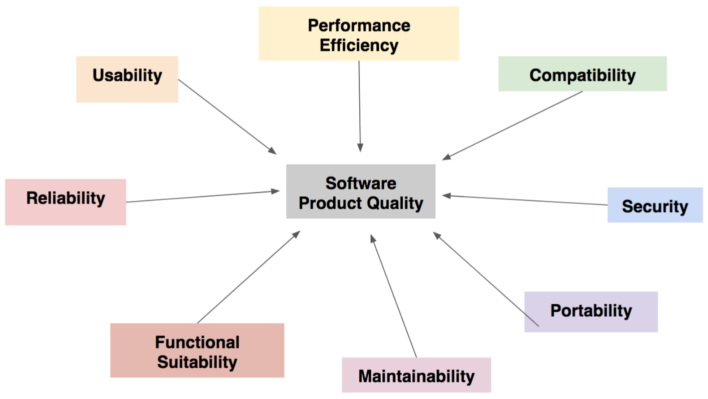

Figure 2: Eight software quality characteristics defined by the ISO/IEC 25010

Manual Classification We randomly sampled 1000 software related sentences from a total of 1923 software related sentences found in **RQ2** for analysis (Table 6). In our case, one thousand (1000) software related sentences were more than enough to statistically represent the entire software related sentences (with 99% confidence level and 3% margin of error, a minimum random sample size required to be a representative of 1943 sentences is 943 sentences). Each sentence was classified with respect to the ISO/IEC 25010 software product quality definitions. To ensure that we eliminated as much bias as we could when annotating, we utilized the same manual classification procedure described in **RQ2**. However, only two annotators were working on this task, the first author and one of the four Master's level CS students that worked on **RQ2**. We resolved disagreements by discussion. ISO/IEC 25010 [3] defines the quality of a system as the extent to which the system satisfies the stated and implied needs of its various stakeholders, which in this case are the end-users. The model comprises of eight high-level software quality characteristics: Functional suitability, Performance efficiency, Usability, Portability, Compatibility, Reliability, Maintainability, and Security. Figure 3 depicts the eight top-level software quality characteristics according to the ISO/IEC 25010. Note that each characteristic in the top-level contains several sub-level categories (please refer to [3] for the full list of definitions for each software quality characteristic and its sub-level categories). The sentence can be classified into one or more software quality aspects, as shown in Table 10. For example, "*Would receive five stars if it played well with other devices and had just a little more customization options.*" implies that the user is not satisfied with the completeness of the functionality that the device offers due to the lack of some customization options (Functional Completeness which is a sub-category of "Functional Suitability") and with the "Compatibility" issue with other devices.

Results and Discussion:

Table 10 shows an example of software related sentences manually classified with respect to the ISO/IEC 25010. We can observe in Figure 3 that, surprisingly, although the studied subjects are an IoT product that can connect to its ecosystem and the Internet, users rarely do discuss issues or details about *Security*, but they focus more on the *Functionality*, *Usability*, and *Compatibility* of software in the product. This is to be expected because these three software quality characteristics can easily and directly be observed by the end-users. To clarify, first users know that an IoT product must be able to communicate or interoperate with other devices in its ecosystem or to the Internet (Compatibility) and second the completeness and correctness of the functionality as well as the ease of use and the aesthetics of the interface can be easily assessed by them. Maintainability is, however, rarely discussed by the users. The reason may be because in these products, users do not have control over updating or reinstalling firmware or an operating system in the product. In other words, they do not experience an issue associated with the new or updated version of the product directly (*Modifiability*) because this procedure is done automatically in the background and they may not be aware of the change in the version of software, firmware, or an operating system. Portability (associated with *Adaptability*, *Installability*, and *Replaceability*) is another interesting software quality characteristic that is hardly ever discussed by the users. The reason may be that software, firmware, or an operating system within an IoT product cannot be ported into other devices and can only be installed in a particular device. Additionally and similarly, users may not experience updating or reinstalling firmware, software, or an operating system of this devices directly and therefore, users are not exposed to the *"Adaptability"* aspect of software in the product.

Due to the scope of this study, we will not be investigating whether or how the distribution differs when compared to user reviews of software in other domain (e.g., mobile app user reviews). However, in our recent preliminary study done on 1004 user reviews (2536 review sentences) of 46 releases of three OSS android applications by the same set of annotators, we found that users do mention software replaceability (i.e., Portability) in a much greater quantity and software interoperability (i.e., Compatibility) in a much lower quantity than what they exhibit in this domain [20]. However, it is important to note that these results are heavily dependent on the domain of the studied software system. Nevertheless, we plan to carry out an in-depth analysis on this idea in the future.

Table 10: An example of software related sentences with its respective software quality characteristic(s)

Software Related Sentence	**SQ Characteristics**
"Does not support sync with apps such as Sleep Cycle or Apple Health."	Functional Suitability, Compatibility
"The camera keeps freezing up even after updating the firmware."	Reliability
"The Fitbit app keeps all his stats and is easy to use!"	Functional Suitability, Usability
"I absolutely loved my Echo dot until I realized there is ZERO security and anyone within range can connect their phone to your device via Bluetooth."	Compatibility, Security
"It's difficult enough to use because you have to memorize all the commands to really benefit from it's full functionality."	Usability
"When we ask Alexa to take an action, there is a good time of silence before Alexa responds."	Performance Efficiency
"This device is not friendly to a person who is not tech smart."	Usability

3.5 RQ5

Procedure:

Aforementioned, in addition to writing a review of a purchased product, a user can subjectively give a star rating on a scale of 1 to 5. In this question, we examine the relationships between each user rating and each type of user review sentence. In particular, we break **RQ5: Are there any patterns in the studied product reviews with regards to the user ratings?** down into four sub-questions:

- ***RQ5.1*** Does the distribution of top-level categories differ across user ratings?
- ***RQ5.2*** Is there an association between the rank order of software-level categories and rating?
- ***RQ5.3*** Is there an association between the rank order of software quality characteristics and rating?
- ***RQ5.4*** Are category and rating independent or related to one another?

To answer ***RQ5.1***, ***RQ5.2***, and ***RQ5.3*** we seperated user review sentences based on the rating that the sentence belongs to. Then, for each rating, we ranked the results according to the percentage each category has. We also employed a number of statistical tests to see whether the ranks differ significantly. To answer ***RQ5.4***, we used the results obtained from the previous three sub-questions and employed a number

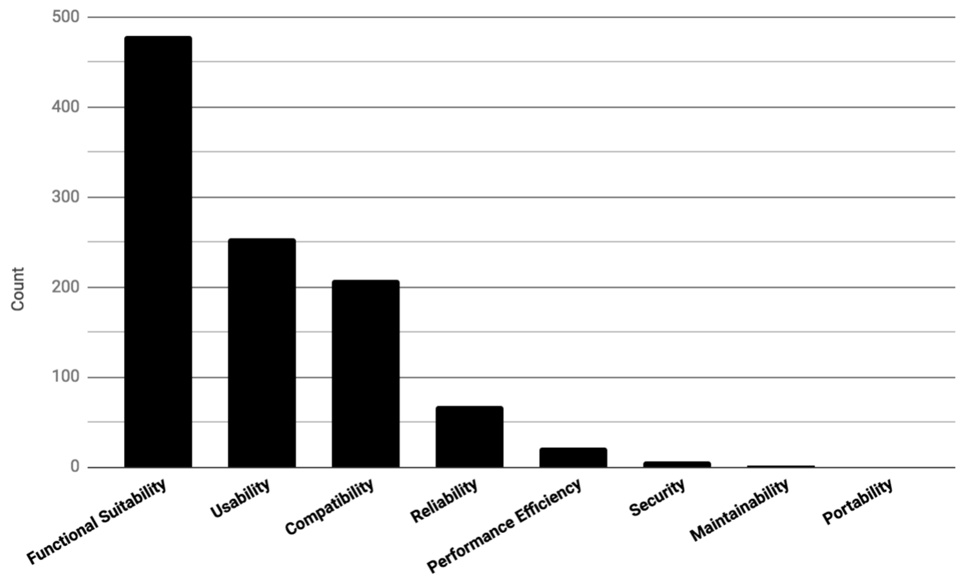

Figure 3: Count of software related sentences in each software quality characteristics

of chi-squared tests of independence ($\chi^2 - tests$) to test if each category and user rating are independent or related to one another.

Results:

Table 11: Percentage of each top-level category per each rating

	Hardware	Software	General	User Background	Other Product	Usage Scenario	Customer Service	Misc	N
1-star	18.00%	19.12%	27.08%	20.80%	3.70%	2.72%	3.98%	4.61%	1433
2-star	21.01%	21.62%	24.59%	17.14%	5.49%	3.75%	2.58%	3.81%	1785
3-star	23.81%	22.28%	20.33%	16.54%	7.63%	4.33%	2.44%	2.63%	1638
4-star	20.94%	22.33%	23.33%	17.25%	6.18%	6.80%	0.96%	2.20%	2087
5-star	17.13%	18.05%	26.86%	19.88%	6.02%	7.73%	1.50%	2.84%	2394

Table 11 shows the distribution of top-level category per rating. The table also shows that regardless of what rating users give, users often give a broad assessment of the product (1st rank for General category) and oftentimes discuss the background of themselves more than giving an assessment of product's software or hardware (2nd ranking in 1-star and 5-star). We initially hypothesized that a one-star rating review would contain more specific details on why the users dislike the product. However, the data shows contradictory results. Upon inspecting the reviews, we found that for one-star reviews, users often give a short and broad critque on the product as a whole, for example, *"The device has been a huge disappointment"*, *"I'm very disappointed in this"*, or *"This camera is garbage"*.

To answer ***RQ5.1***, we first employed a number of Mann-Whitney U tests to test whether the two independent samples have the same distribution. Our null hypothesis (**H**0) is that there is no difference between the ranks of the two samples. We set the significance level to 0.05. We also employed Kendall's tau rank correlation to test the statistical associations based on the ranks of the data. The null hypothesis (**H**0) for this test is that there is no association between the two samples. The tau correlation coefficient returns a value of 0 to 1, where 0 denotes no relationship and 1 denotes a perfect relationship [21]. Similarly, we set the significance level to 0.05.

We found that there is no difference ($p > 0.05$) in terms of the distribution of all X-rating to Y-rating pairs, where $X \neq Y$ (i.e., all ten pairs). However, the tau correlation reveals weak associations (not statistically significance) between the rank orders of the pair of 1-star and 3-star ($\tau = 0.35$, $p = 0.13$) and 1-star and 4-star ($\tau = 0.5$, $p = 0.053$).

Table 12 shows the distribution and rank order of software-level category per rating. Note that we could not employ Mann-Whitney U test for RQ5.2 because we had less than five categories per rating. To answer

Table 12: Percentage of each software level category per each rating

1-star (N = 274)	2-star (N=368)	3-star (N=364)	4-star (N=466)	5-star (N=432)
PD 75.91%	PD 68.75%	PD 58.24%	IG 51.07%	IG 87.73%
IG 18.61%	IG 21.74%	IG 29.67%	PD 32.83%	PD 7.41%
FR 4.74%	FR 7.88%	FR 10.99%	FR 14.38%	FR 4.63%
IQ 0.73%	IQ 1.63%	IQ 1.10%	IQ 1.72%	IQ 0.23%

FR = Feature Request; IG = Information Giving; PD = Problem Discovery; IQ = Inquiry

RQ5.2, we employed only Kendall's tau rank correlation to test the statistical associations based on the ranks of the data. We found a significantly strong association (at $p < 0.05$) between the rank order of software-related categories in 1-star and 2-star ($\tau = 0.99$), 1-star and 3-star ($\tau = 0.99$), 2-star and 3-star ($\tau = 0.99$), and 4-star and 5-star ($\tau = 0.99$). As you can see from Table 12, 4-star and 5-star have the same rank order but do not have the same rank order as 1, 2, or 3-star. This conforms with the results we obtained from applying Kendall's tau rank correlation.

To answer ***RQ5.3***, we first applied Mann-Whitney U test to see whether each pair (pair of X-rating and Y-rating, where $X \neq Y$) have the same distribution. We found that there is no difference ($p > 0.05$) in terms of the distribution of all X-rating to Y-rating pairs, where $X \neq Y$ (i.e., all ten pairs). Next, we examined if there is an association between the rank order of each pair by using Kendall's tau rank correlation. The tau coefficient shows a strong association but not significant between the pair of 1-star and 5-star ($\tau = 0.8$, $p = 0.065$) and 2-star and 5-star ($\tau = 0.8$, $p = 0.065$). Table 13 shows the distribution and rank order of software quality characteristics per rating.

Table 13: Top-5 software quality characteristics per each rating

1-star (N = 157)	2-star (N=192)	3-star (N=190)	4-star (N=252)	5-star (N=248)
Functional Suitability (38.22%)	Functional Suitability (47.40%)	Functional Suitability (45.26%)	Functional Suitability (44.84%)	Functional Suitability (51.61%)
Compatibility (21.02%)	Compatibility (20.83%)	Usability (25.79%)	Usability (28.17%)	Usability (27.02%)
Usability (19.75%)	Usability (19.27%)	Compatibility (18.42%)	Compatibility (19.84%)	Compatibility (20.16%)
Reliability (18.47%)	Reliability (7.81%)	Reliability (6.84%)	Reliability (4.37%)	Performance Efficiency (0.81%)
Performance Efficiency (2.55%)	Performance Efficiency (2.60%)	Performance Efficiency (2.63%)	Performance Efficiency (1.98%)	Maintainability (0.40%)

To answer ***RQ5.4***, we employed a number of chi-squared tests of independence ($\chi^2 - tests$) to see if each category and rating are independent or related to one another. The null hypothesis (**H**0) for this test is that the two categorical variables are independent. We set the significance level to 0.05.

Top-level Category:

We found that Hardware and rating ($\chi^2 = 33$, $p < 0.05$), Software and rating ($\chi^2 = 19.3$, $p < 0.05$), General and rating ($\chi^2 = 29.26$, $p < 0.05$), User Background and rating ($\chi^2 = 16.475$, $p < 0.05$), Other Product and rating ($\chi^2 = 22.31$, $p < 0.05$), Usage Scenario and rating ($\chi^2 = 66.7$, $p < 0.05$), Customer Service and rating ($\chi^2 = 44.17$, $p < 0.05$), and Miscellaneous and rating ($\chi^2 = 21.03$, $p < 0.05$) are significantly dependent.

Software-level Category:

We found that Problem Discovery and rating ($\chi^2 = 489.86$, $p < 0.05$), Feature Request and rating ($\chi^2 = 35.31$, $p < 0.05$), and Information Giving and rating ($\chi^2 = 517.84$, $p < 0.05$) are statistically dependent. However, only Inquiry and rating are statistically independent ($p > 0.05$)

Software Quality Characteristics

We found that Functional Suitability and rating ($\chi^2 = 7.3$, $p > 0.05$), Compatibility and rating ($\chi^2 = 0.49$, $p > 0.05$), Usability and rating ($\chi^2 = 7.6$, $p > 0.05$), and Performance Efficiency and rating ($\chi^2 = 4.34$, $p > 0.05$) are statistically independent. However, only Reliability and rating are significantly dependent ($\chi^2 = 53.31$, $p < 0.05$).

Summary of findings for *RQ5*

First, we found that the distribution of top-level categories does not differ across user ratings. In other words, there exists no pattern in the top-level category with regards to user ratings. Second, our statistical analysis reveals a pattern in how users talk about the software in the product: in 1-star, 2-star, and 3-star, users mainly focus on describing the issues or unexpected behaviors they are facing with the software in the product, whereas in 4-star and 5-star, users focus more on informing other users about the functionality of the software of the product. Our results agree with one of the results from the study done by Pagano and Maalej [22] on mobile apps reviews on the Google Store. They found that shortcomings and bug reports have high influence on negative ratings (1-star, 2-star, and 3-star), while feature information on positive ratings (4-star and 5-star). This suggests that, similar to mobile app developers, software developers who perform software maintenance and evolution tasks on an IoT product should benefit from analyzing a 3-star review and below to speedily get the crucial information on what problem or issue users may encounter with the software in the product. Third, we found that the distribution and the rank order of the software characteristics do not differ across user ratings. Fourth, our results reveal that Reliability aspect of the software and user rating are statistically dependent. From Table 13, we can observe that 1-star contains extremely high number of sentences mentioned about the Reliability aspect of the software compared to 2-star, 3-star, 4-star and 5-star. This suggests that analyzing a 1-star review can reveal information on the Reliability problem or issue that user may encounter in a greater quantity. Our results also reveal that all but one software level category and user rating are statistically dependent. Table 12 suggests that 1-star contains the highest number of Problem Discovery sentences, 4-star contains the highest number of Feature Request sentences, 5-star contains the highest number of Informative Giving sentences. This suggests that developers can target a particular star rating in order to retrieve specific information (what feature users want, what issue or problem they encounter, or what functionality users like) with regards to the software in their product.

4 Threats to Validity

The study presented in this paper has several factors that may affect the validity of the results. In the following, we discuss possible threats to the validity in our study.

Taxonomy This threat concerns the validity of our taxonomy. To mitigate this threat, we conducted both internal content analysis, involving a team of 5, and external content analysis, involving 52 master's level CS students, to analyze what information or pattern contained in the reviews. In addition, we also adapted taxonomy proposed by past literature which has been evaluated the relevance to developers performing software evolution and maintenance tasks in our work (See Section 3.1). However, other studies that use taxonomy with a different set of categories and definitions might lead to different results.

Subjectivity in manual classification This threat stems from the fact that our ground truth dataset is based on human judgment. However, it is not uncommon to involve humans to manually classify data in a text classification problem. To mitigate such a threat, we employ multiple measures as aforementioned in Section 3.2. Nevertheless, we cannot claim that our dataset is error-free as some bias may remain.

External Validity This threat concern how generalizable our results are. To mitigate such a threat, we selected products from different application domains: smart home, smart watch, action camera, and gaming console. Furthermore, to eliminate some bias in our dataset, we selected at most 50 reviews per star rating for each product, and each review contains less than 20 sentences. We also performed a project 6-fold cross validation (see Section 3.3) to test the generalizability of our machine learning models. Additionally, since all IoT products have the capability to transfer data or connect to its ecosystem or the Internet, this shared similarity should allow our results to generalize. Nevertheless, there are other IoT domains that we have not covered as well as multiple products in the same domain as the product we selected (i.e., our selected products may not be representative). We encourage further research to investigate whether our results hold in other IoT domains.

5 Related Work

Analyzing user reviews for useful information has gained a lot of attention from researchers in recent years. We highlighted past literature that share similarities with our work.

Pagano and Maalej [22], Khalid et al. [4], Hoon et al. [23], and Harman et al. [24] conducted exploratory studies by investigating multiple aspects of user reviews from mobile application distribution platforms such as the Google Store, the Apple Store, or the BlackBerry Store. They found that user reviews contain information that is useful to the developers and the companies. Motivated by their findings, we investigated user reviews on IoT products to see whether enough software related information exists and how much of those information is directly applicable to software engineers.

Guzman et al. [6] conducted an exploratory study by analyzing the content in Twitter's tweets to find useful information for software engineers. They manually classified 1000 tweets and found that tweets contain relevant information for different stakeholders. They also investigated the automation potential by using several supervised machine learning techniques. Our work is similar to their work, however, our studies' subjects are not software applications.

Maalej and Nabil [5] compared multiple methods that can help with classifying user reviews on App stores automatically. Similarly, Panichella et al. [8] applied several machine learning techniques to classify information in user reviews from App stores. They applied Natural Language Processing (NLP), sentiment analysis, and text analysis to help with classification tasks. They found that combining NLP with sentiment analysis improves both precision and recall significantly. In contrast, we included a neural network approach (CNN) and investigated if vector space models such as Word2Vec and TF-IDF improve the performance of the classifiers.

6 Conclusions and Future Work

In this paper, we conducted an exploratory study by analyzing 1491 verified purchase reviews (7198 review sentences) of 6 IoT products obtained from Amazon. Our results demonstrate that only 26.72% of all sentences are software related, based on our taxonomy defined through external and internal content analysis sessions. We investigated how much information in those software related sentences is useful for software engineers performing software evolution and maintenance tasks. The results show that only 55.28% of software related sentences (14.79% of all sentences) are directly applicable to software engineers. Given that only a small quantity of sentences can help to accelerate software requirements elicitation or evolution process, we studied the extent to which two supervised machine learning techniques (SVM and CNN) can be used to differentiate information contained in the reviews automatically. The results show that utilizing Word2Vec improved the performance of SVM. In addition, Word2Vec also helped the model to generalize better, when classifying unseen reviews of a different product, than tf-idf. Moreover, our results reveal that some software quality characteristics can easier be observed by the users than some software quality characteristics. Surprisingly, even in smart home domain, we found that users rarely discuss the security aspect of the software in these products. In addition, our results reveal patterns in the type of review sentences with regards to the rating. These patterns could help developers performing software evolution and maintenance tasks on an IoT product to easily and speedily identify issues users may encounter with the software in the product.

This work can be extended to several directions. For instance, we plan to incorporate more product from several other IoT domains, to include feedback from the manufacturers on how useful the findings are, to incorporate sentiment analysis and other types of preprocessing approaches from past literature to further improve the performance of the classifier, to officially compare the performance of our classifier with that of past literature for software-level sentences, and finally to investigate if we can capture and construct a formal requirement (*e.g.,* a user story) from these review sentences automatically.

Acknowledgment

This material is based upon work supported in part by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. It is also supported by the National Science Foundation grant CMMI-1408909, Developing a Constructive Logic-Based Theory of Value-Based Systems Engineering.

References

[1] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[2] N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews," *Journal of Systems and Software*, vol. 125, pp. 207–219, 2017.

[3] "ISO/IEC 25010:2011: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," ISO, Standard, 2011.

[4] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?" *IEEE Software*, vol. 32, no. 3, pp. 70–77, 2015.

[5] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *Requirements Engineering Conference (RE), 2015 IEEE 23rd International.* IEEE, 2015, pp. 116–125.

[6] E. Guzman, R. Alkadhi, and N. Seyff, "A needle in a haystack: What do twitter users say about software?" in *Requirements Engineering Conference (RE), 2016 IEEE 24th International.* IEEE, 2016, pp. 96–105.

[7] D. L. Morgan, "Qualitative content analysis: a guide to paths not taken," *Qualitative health research*, vol. 3, no. 1, pp. 112–121, 1993.

[8] 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," in *Software maintenance and evolution (ICSME), 2015 IEEE international conference on.* IEEE, 2015, pp. 281–290.

[9] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[10] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[11] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986.

[12] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[14] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[15] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems.* ACM, 2013, pp. 165–172.

[16] O. Luaces, J. Díez, J. Barranquero, J. J. del Coz, and A. Bahamonde, "Binary relevance efficacy for multilabel classification," *Progress in Artificial Intelligence*, vol. 1, no. 4, pp. 303–313, 2012.

[17] M. S. Sorower, "A literature survey on algorithms for multi-label learning," *Oregon State University, Corvallis*, vol. 18, 2010.

[18] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza, "Content classification of development emails," in *Software Engineering (ICSE), 2012 34th International Conference on.* IEEE, 2012, pp. 375–385.

[19] A. Di Sorbo, S. Panichella, C. V. Alexandru, C. A. Visaggio, and G. Canfora, "Surf: Summarizer of user reviews feedback," in *Software Engineering Companion (ICSE-C), 2017 IEEE/ACM 39th International Conference on.* IEEE, 2017, pp. 55–58.

[20] K. Srisopha and R. Alfrayez, "Software quality through the eyes of the end-user and static analysis tools: A study on android oss applications," in *2018 ACM/IEEE 1st International Workshop on Software Qualities and their Dependencies (SQUADE'18), located at the International Conference of Software Engineering (ICSE'18).* IEEE, 2018.

[21] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[22] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *Requirements Engineering Conference (RE), 2013 21st IEEE International*. IEEE, 2013, pp. 125–134.

[23] L. Hoon, R. Vasa, J.-G. Schneider, J. Grundy *et al.*, "An analysis of the mobile app review landscape: trends and implications," *Faculty of Information and Communication Technologies, Swinburne University of Technology, Tech. Rep*, 2013.

[24] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: Msr for app stores," in *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*. IEEE, 2012, pp. 108–111.