An Automated Goal Labeling Method Based on User Reviews

Shuaicai Ren* Hiroyuki Nakagawa* Tatsuhiro Tsuchiya*
*Graduate School of Information Science and Technology
Osaka University, Suita, Japan
Email: {s-ren, nakagawa, t-tutiya}@ist.osaka-u.ac.jp

***Abstract*—Requirements analysis is an important step in a software development process. Summarizing user reviews is an efficient way of requirement elicitation. Nevertheless, it is difficult to manually collect user reviews from app stores. In order to solve this problem, we proposed a method that automatically elicits requirements and establishes a goal model for visualization. To improve the previous method, this paper proposes a method of defining labels of goals. Experimental results demonstrate that those labels can help developers understand requirements more easily and precisely.**

***Index Terms*—Requirements elicitation, goal modeling, user reviews, labeling**

I. Introduction

With the expansion of the mobile phone market, various smartphone applications have been developed, such as SNS, payment applications and social games. To ensure their attraction, developers have to collect feedbacks from application users. In application stores, users can publish reviews to rate applications, ask for a new function or report bug information. Developers can investigate these reviews and extract important requirements to improve applications from them. However, the number of reviews is too large to extract requirements from reviews.

In order to help developers understand user requirements, we have reported a preliminary result of review clustering and a goal model construction from user reviews [19]. The goal model is one of requirements models, which describes requirements as goals to be satisfied. In our previous paper, the construction method structurally visualizes requirements extracted from user reviews in a goal model. Nevertheless, the tool has a problem: it is too difficult to understand what these goals represent. To deal with this problem, we propose a new labeling method in this paper.

The contribution of this paper is **goal labels automatically from user reviews**. We use two methods to give weights to user reviews and select useful sentences from reviews as labels. Two experiments demonstrate the difference between new labels and old ones. According to the results of the two experiments, we report the evaluation of our method.

The structure of this paper is as follows. Section II explains the background of our research; Section III describes the previous construction method in detail; Section IV presents the new labeling method; Section V evaluates the labeling method and reports the challenges to improve the method; Section VI covers the conclusion and future work.

DOI reference number: 10.18293/SEKE2020-134

II. Related work

Requirements analysis has always been an important step in the software development life cycle. This step analyzes what kinds of requirements need to be satisfied by the system. After analyzing the requirements, developers can make documents which describe the system capability. For better requirements analysis, many methods have been proposed. Palmieri et al. [17] proposed a tool-supported method that integrates a goal-oriented requirement language and feature modeling to handle regulatory goal model families. Bettenburg et al. [2] reported a study about what kinds of information from users are required by developers. That method helps users to improve the quality of bug reports. Higashi et al. [7] provided a method of improving the accuracy of LDA review classification. Chen et al. [4] developed a tool called AR-miner, which can help developers to filter some useless reviews and classify useful ones. As a result, this tool shows the groups of the most "informative" reviews via an intuitive visualization method. Maalej et al. [11] collected massive reviews from Google Store and App Store, and tried to classify them. They compared several types of technologies for language processing and some machine learning methods. In those machine learning methods, Naive Bayes classifier had the best result. Unlike these papers we focus on how to represent user requirements via a goal model. Our work includes eliciting useful sentences through user reviews and labeling goals from the goal model.

Goal models include KAOS [6], i* [23], NFR [14], AGORA [9] and Tropos [21]. In goal models, requirements are described as goals that demand to be achieved. Goals are linked based on the relationship between them. Abstract goals become the parent goals and detailed goals will become the subgoals. For example, the goal "file operations provided" could be a parent goal of the goal "files edit function provided."

To high quality goal models, the requirements from users are necessary.

Users submit requirements or bugs by writing reviews in app stores. Some of these reviews are valuable for developers[16]. Nevertheless, there are also meaningless and low-quality reviews. Due to large numbers of reviews, it is hard to find useful reviews manually[8]. We have reported a preliminary result of finding useful reviews and building goal models. This preliminary method still has disadvantages. Normally goals have manual labels that describe requirements. The preliminary method labels each goal with a word that is automatically extracted from reviews. The word that seems to be the most relevant to the goal is chosen as a label. Unfortunately those labels may not clearly describe requirements. To make labels more easy to understand, we propose a new method for labeling goals in this paper.

III. Goal model construction method

In this section, we will briefly introduce the goal model construction method that we have previously proposed in [19]. This modeling method is mainly composed of two steps: clustering and goal labeling.

Step 1: Clustering. Algorithm 1 shows the process of clustering. First, user reviews are broken down into text. Second, we delete stopwords which appear frequently in reviews but do not have meaning, such as "is, are, a, an, the". Deleting those stopwords is a common approach to process natural language. This approach can help us to find words that show users' requirements more easily. In this method, we added some other words like "ur" (your) and "dis" (this) into the stopwords from NLTK[3]. These words come from users' oral habit and have no relationship with users' requirements.

Next, filtered words are added to a dictionary, and then lists representing bag-of-words (BoW) [12] are generated from the dictionary. BoW represents words in a document and the number of their occurrences. Then, the generated lists of BoW are stored in a matrix as vectors. For example, suppose a document contains following two sentences:

- I cannot open any of my company documents now from the app.
- It won't let me open any preexisting documents.

Lists of BoW after lemmatization and filtering stopwords are [“open”: 1, “company”: 1, “document” : 1, “app” : 1] and [“open”: 1, “preexist”: 1, “document”: 1]. Finally, these lists are stored in a matrix. After data preparation, Ward's method [22], a hierarchical clustering method, is applied to the matrix and the result of review clustering is obtained. In Ward's method, the minimum variance criterion is used to couple clusters:

$$d_{ij} = d(\{X_i\}, \{X_j\}) = \|X_i - X_j\|^2 \tag{1}$$

Since two BoWs have the same words "open" and "document" in the above example, the distance between two sentences becomes close by clustering.

Step 2: Labeling. We have reported a preliminary method to label goals. Algorithm 2 shows the process of goal labeling. In this step, we used document frequency to define goal labels. For leaf goals, the goals that do not have subgoals, their labels are clustered words' weight. The weight comes from the document frequency, which means that if a word frequently appears in this cluster, it probably appears in the goal label. For parent goals, their labels come from the words that appear frequently in every subgoal. Through this way we can ensure that parent goals are labeled with words which are from each subgoal.

Algorithm 1 Clustering

```
1: Input: titles and texts of user reviews
2: $array$ = array of $number\ of\ reviews \times size\ of\ vocabulary$
3: for $review$ in $reviews$ do
4:     $wordlist \leftarrow$ lemmatized words not in stopwords
5:     $bows \leftarrow$ bag-of-words (BoW) of $wordlist$ /* generate BoW */
6:     for $\{word\_id, frequency\}$ in $bows$ do
7:         $array[\#\ review][word\_id] \leftarrow frequency$
8: apply Ward method to $array$
9: Output: a clustering result
```

Algorithm 2 Goal labeling

```
1: Input: a coupled cluster
2: $cluster$ = cluster of reviews
3: $wordlist$ = map of $\{review[word], DF\}$ /* $DF$: document frequency */
4: for $review$ in $cluster$ do
5:     for $word$ in $review$ do
6:         /* $word$ does not occur in $review$ yet */
7:         if $word$ is not in stopword and $word$ is not counted then
8:             $wordlist[word]$ += 1
9: exclude words that occur in two sibling goals
10: sort $wordlist$ by $DF$ of $word$
11: select top words from sorted $wordlist$
12: Output: a goal description
```

The previous method applies a clustering method for grouping reviews and constructing a hierarchical structure. However, the goal labels obtained by the previous labeling method are still hard to understand. For example, it is hard to know the meaning of the generated label "never, device, book, version, day". In order to solve this problem, we propose a new labeling method in this paper.

IV. New labeling method

The new labeling method utilizes sentences as goal labels. There are two approaches for automatically generating sentences. One approach generates sentences directly from documents. However, this approach has too many limitations, which makes this approach difficult to use in practice. So this paper adopts the second approach, which selects the sentences contained in the documents as a representative. To prevent from containing too many requirements in one goal, we select only one sentence for one goal as the label.

The overview of our new construction method is illustrated in Figure 1. Since the clustering method is the same as the previous method (the green part), this section mainly introduces the labeling method (the blue part). The overview of our new labeling method is illustrated in Figure 1. The labeling method is mainly composed of two methods: *selecting by TFIDF* and *selecting by cosine similarity*. Those two have

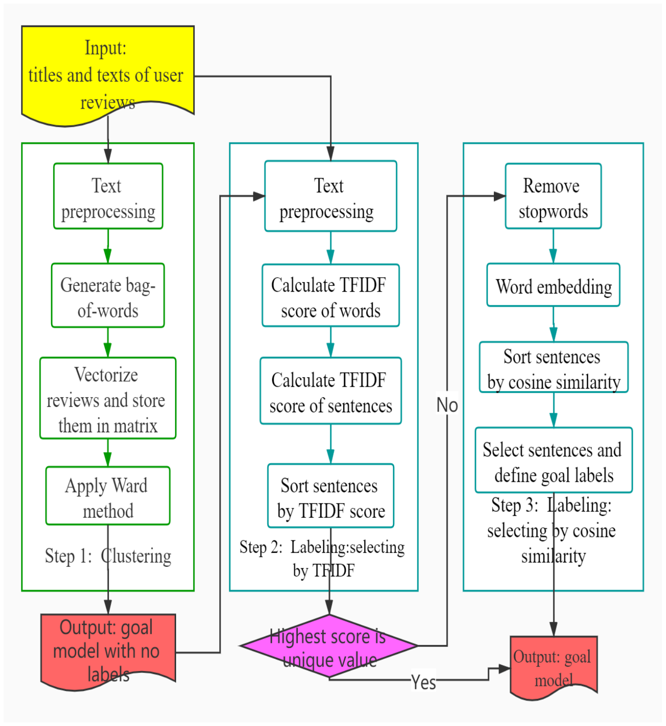
Input: titles and texts of user reviews
Text preprocessing
Generate bag-of-words
Vectorize reviews and store them in matrix
Apply Ward method
Step 1: Clustering
Output: goal model with no labels
Text preprocessing
Calculate TFIDF score of words
Calculate TFIDF score of sentences
Sort sentences by TFIDF score
Step 2: Labeling:selecting by TFIDF
Highest score is unique value
No
Yes
Remove stopwords
Word embedding
Sort sentences by cosine similarity
Select sentences and define goal labels
Step 3: Labeling: selecting by cosine similarity
Output: goal model

Fig. 1. Overview of the construction method. The green part shows the clustering method and the blue part illustrates the new labeling method.

Algorithm 3 Selecting by TFIDF

```
1: Input: titles and texts of user reviews
2: for $review$ in $reviews$ do
3:     $sentencelist$ ← divided sentences in reviews /* segment reviews with periods, exclamation marks, and question marks*/
4: for $sentence$ in $sentencelist$ do
5:     for $words$ in $sentence$ do
6:         $TFIDFwordslist$ ← calculate TFIDF of words
7:     $TFIDFlist$ ← calculate TFIDF of sentence
8: sort $TFIDFlist$ by $TFIDF$ of $sentencelist$
9: Output: sentences list
```

different pretreatment process and effects. In pursuit of better results, we decide to combine these two methods.

Step A: Selecting by TFIDF. TFIDF stands for term frequency–inverse document frequency, which is the most frequently applied weighting scheme [1] in text mining. This technique can reflect the importance of a word to a document in a corpus [20]. The importance of a word increases proportionally with the times that this word appears in the document, but it decreases inversely with the frequency that this word appears in the corpus. TFIDF is appropriate for our purpose to select keywords, as it gives a high weight to a word that only appears in one cluster.

Fig. 2 illustrates the overview of step A. Algorithm 3 shows the algorithm for selection by TFIDF. First of all, with preprocessing, these reviews are divided into sentences. The general sentences end with periods, exclamation marks and question marks. In order to prevent sentences from being too

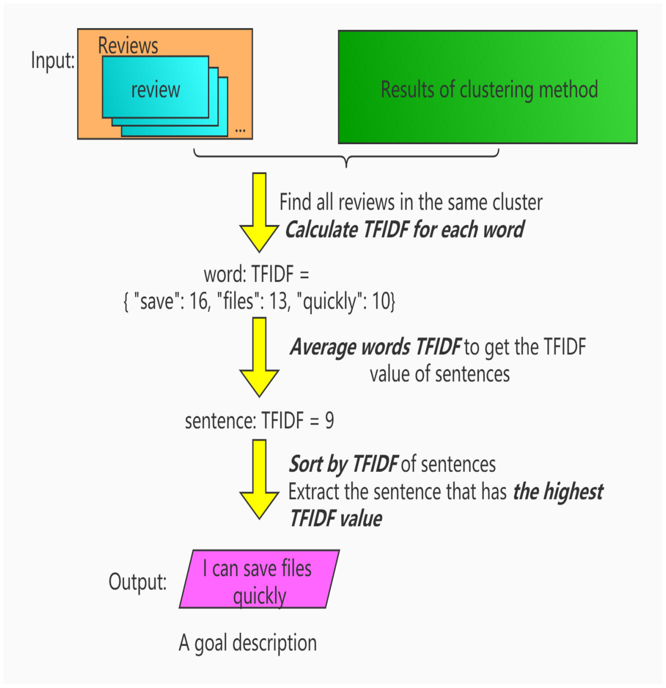
Input:
Reviews
review
...
Results of clustering method
Find all reviews in the same cluster
Calculate TFIDF for each word
word: TFIDF =
{ "save": 16, "files": 13, "quickly": 10}
Average words TFIDF to get the TFIDF value of sentences
sentence: TFIDF = 9
Sort by TFIDF of sentences
Extract the sentence that has the highest TFIDF value
Output:
I can save files quickly
A goal description

Fig. 2. Overview of labeling by TFIDF.

long and containing more than one requirement in one goal label, we segment reviews also with commas that separate sentences in a compound sentence. Next, the TFIDF score for each word is calculated, and then we average them to get the score of the sentence:

$$TFIDFs = \frac{\sum_{i=1}^{n_s} TFIDF_i}{n_s} \tag{2}$$

n_s means the number of words of one sentence. $TFIDF_i$ means TFIDF score of the ith word order. Finally, we sort these sentences and select the sentence with the highest TFIDF score.

Step B: Selecting by cosine similarity. In the previous step, TFIDF score used to sort sentences. However, in some cases, the first two sentences have the same score. We require to find the one that is more similar to the cluster. For this we use word embedding techniques. Word embedding is a general name for a set of technologies to process natural language. One method of word embedding is dimensionality reduction in the word co-occurrence matrix [10]. According to this method, a space with many dimensions per word is embedded into a continuous vector space with a lower dimension, and each word or phrase is mapped to a vector on the real number field. Now there are some tools to achieve this method, for example, the word2vector [13] and doc2vector. After getting the word vector, we apply the cosine similarity, which is a measure of similarity between two non-zero vectors of an inner product space, to calculate the similarity between vectors, words or sentences.

First, reviews were filtered by stopwords. Then, we apply doc2vec to achieve sentence embedding. According to doc2vec, vectors of all sentences synthesize the vector of the cluster so that the cosine similarity between sentences and the cluster can be calculated. The vector with the highest cosine similarity to the cluster will be selected.

V. Experiment and Evaluation

A. Purpose of Experiment

We focus on answering the following two research questions:

- *RQ1:* Do the labels correctly reflect the intent of the goals?
- *RQ2:* Can the labels be properly understood by the developers?

To answer the research questions, we made the goal models first. Reviews for making models were taken from Google Docs with the App Store. Due to the numerous reviews, goal models contain a large number of goals. In order to facilitate the experiment, we extracted a part of the goal model. In this experiment, we used the part corresponding to reviews about the cross-platform function of Google Docs. Three goal models' labels came from manual, the previous labeling method and the new labeling method. As we mentioned in Section IV, we segmented reviews with commas that separate sentences in a compound sentence. In this experiment, sentences that have less than five words were not treated as independent sentences in a compound sentence. The result of modeling and manual labeling is illustrated in Figure 3. Those labels are correct labels. To answer our research questions, two experiments were conducted. We explained the detail of each experiment below.

B. Experiment 1

Design of experiment 1. In the experiment 1, we enumerated the ideal labels made by hand, the labels of the new method and the labels of the previous method. We comprehend the gap between the two kinds of automatically generated labels and the ideal labels and whether automatically generated labels can be understood. This experiment used the correctness rate with ideal labels and understanding rate to evaluate both methods. The evaluation of the correctness rate and understanding rate in the table was completed by the authors.

Results of experiment 1. Table I demonstrates the difference between the previous labels and the current labels. Experiment 1 illustrates that the correctness rate of the previous method is 54%, while the understanding rate is 30%. As for the new method, the correctness rate is 77% and the understanding rate is 100%.

C. Experiment 2

Design of experiment 2. Experiment 2 aims at people other than authors. To comprehend the effect of labels on other people, we produced questionnaires for the examinees. In each questionnaire, examinees were asked about what kind of requirements were reflected. These examinees included four professionals and four non-professionals. The answering order of the questionnaires was random; some examinees were asked to complete the previous labels questionnaires first, while others were the opposite. As for the correctness rate of labels, we also use correct labels to compare the labels obtained by two methods.

Results of experiment 2.

The results of experiment 2 are illustrated in Table II. This figure indicates that new labels have a higher correctness rate in total.

D. Discussion

First, we answer *RQ1*: "Do the labels correctly reflect the intent of the goals? ". The experiment and questionnaire results demonstrate that labels generated by using this method have a higher accuracy rate. But there is one thing to be noted. For abstract goals at the top layers of the goal models, the correctness rate of new labels is lower. Normally, the close to the root goal the label is, the more unreliable the label is. We believed the reason is that we directly select the sentences in the reviews as labels. Users prefer to give feedbacks in detail, such as BUG reports and requirements, so our method could accurately capture the sentences that express these requirements as labels. In terms of sentences that can be used as abstract goals' labels, it is difficult to find them in user reviews. To build more precise labels, we could use other requirements mining methods. Conneau et al. [5] proposed several probing tasks designed to capture simple linguistic features of sentences. We can utilize different methods to deal with top goals and bottom goals. When it comes to bottom goals, we still extract sentences from user reviews as labels. In terms of top goals, we could extract keywords and logical relationships from sentences, and use word embedding and vector synthesis to construct labels.

Here, we answer *RQ2*: "Can the labels be properly understood by the developers?". Compared to previously generated labels, Table II demonstrates that examinees can better understand the labels, but it is still difficult for non-experts. The following improvements should be considered:

- **Construction method needs improvement**:
  To classify reviews in more detail, we require to improve our construction method. Our method uses the cluster distance for determining the goal refinement level. When the developers set the threshold to a small value, the size of each coupled cluster becomes small, and then the total number of goals increases in a model generated. In other words, the proportion of the number of detail goals becomes larger.
- **Logical relationships need to be refined**:
  In this method, the relationship between the two sibling goals is limited to *AND-refinement*. In other words, all sibling goals demand to be achieved. But actually, there are many logical relationships and they may be mentioned in the reviews. Extracting these words from reviews will help us improve the goal model construction method.

VI. Conclusion

In this paper, we reported the experience of an automated labeling method on the basis of user reviews. Our method

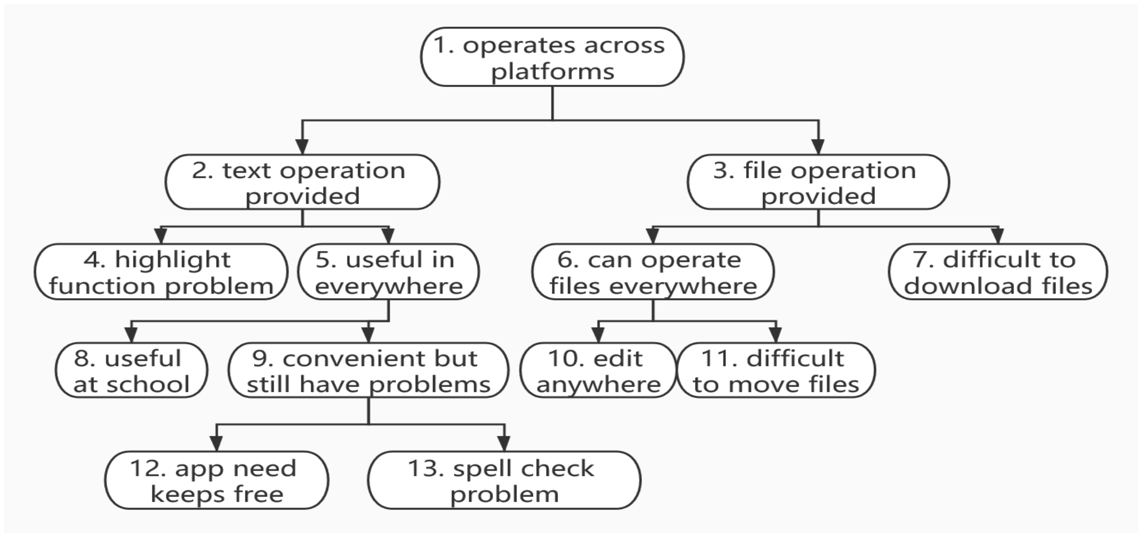
1. operates across platforms
2. text operation provided
3. file operation provided
4. highlight function problem
5. useful in everywhere
6. can operate files everywhere
7. difficult to download files
8. useful at school
9. convenient but still have problems
10. edit anywhere
11. difficult to move files
12. app need keeps free
13. spell check problem

Fig. 3. Labels generated by manual.

TABLE I
LABELS EVALUATION

No.	Correct label	Label type	Label content	Evaluation
1	operates across platforms	Previous	never, device, book, version, day	-/-
		New	Also I never lose anything cause I can access it on any device	C/U
2	text operation provided	Previous	apps, note, time, user, text	-/-
		New	I just hope Google keeps these apps free	-/U
3	file operation provided	Previous	download, platform, live, fact, weird	-/-
		New	Download I hate the fact that you have to buy this	-/U
4	highlight function problem	Previous	text, highlight, browser, copy/cut, superior	-/-
		New	you can't highlight your text	C/U
5	useful in everywhere	Previous	work, school, docs, keep, save	-/-
		New	School I use this for school and it works good	-/U
6	can operate files everywhere	Previous	anywhere, allow, open, document, share	C/U
		New	Easy to access your files anywhere and share with others	C/U
7	difficult to download files	Previous	download, right, template, weird, number	C/-
		New	Templates are annoying I have to download a template	C/U
8	useful at school	Previous	school, ms, finish, right, box	C/-
		New	Use it for school I love it honestly	C/U
9	convenient but still have problems	Previous	note, time, life, feature, helpful	C/U
		New	convenient and a great time saver	C/U
10	edit anywhere	Previous	allow, open, document, share, platform	C/-
		New	edit my documents everywhere that I go	C/U
11	difficult to move files	Previous	live, updates, yesterday, move, adobe	-/-
		New	can't move over files adobe	C/U
12	app need keeps free	Previous	keep, free, save, note, time	C/U
		New	I just hope Google keeps these apps free	C/U
13	spell check problem	Previous	spell, check, belittle, -i, become	C/U
		New	Spell check I can't right-click to fix a word underlined red (spell check) using my chrome book	C/U
C: correct U: understandable -: incorrect or incomprehensible				

TABLE II
LABELS CORRECTNESS RATE

Method & examinee type	Labels correctness rate
Pervious method & non-professionals	26%
Pervious method & professionals	40%
New method & non-professionals	66%
New method & professionals	80%

weights each sentence and selects one of the sentences as a label for a goal. In the evaluation part, we conducted on experiment to evaluate the correctness of the previous method and this method. Then, we used questionnaires to investigate the developer's understanding of the goal model. As for the result of labeling, it is easier and more precise to understand than the previous method generated labels. The correctness rate of labels decreased from the bottom to the top, but the general labels can correctly reflect the intent of the goal model.

For future work, we identify the following improvement points:

- **Abstract label readability**:
  To improve the goal description readability, first, we should refine stopwords. If the number of stopwords is too large or too small, we might miss some requirements. Next, labels require to contain more useful information.

The experiment shows that this method works well with the bottom goals, but as for some top goals, it is hard to find one sentence to represent the goal. After all, it is hard to find abstract sentences from user reviews, and users prefer to report practical things. Furthermore, the AND refinements are not clear from reading the goal labels. For making more effective labels, we can make a sentence instead of selecting a sentence from reviews. Rolland et al. [18] provided a method in order to help developers understand long documents. In that method, some keywords and logical relationships are extracted and used to generate sentences with some rules. We believe that this is a feasible method to make labels. Keywords and logical relationships also can be found in user reviews. Keywords can help us to make labels while logical relationships can help us to clear refinements.

- **Other elements in the goal model need to be considered**:
  In the actual goal models, goals are not the only type of goal model elements. Functional requirements have been described as goals, while non-functional requirements can be described as soft goals. A functional requirement defines a system or its component whereas a non-functional requirement defines the performance attribute of a software system. The goal models generated by this method only have goals, which means that we cannot classify goals, soft goals and bug reports. The final objective of our research is to improve the automated goal modeling method and to visualize not only goals but also bugs and soft goals. To accomplish this objective, we should introduce a mechanism for visualizing the goal type, such as requirements or bug reports. Now we are trying to find a method to judge goal type based on word combinations. Maalej et al. [11] proposed the method to classify reviews into four types, i.e., bug report, user request, user experience, and rating. In users' views, some words combinations can help us to conduct this work. For example, the words combining "can" and "not" often in bug reports. If we can give enough weight to these word combinations, perhaps we can allow types of labels to be recognized more easily. To embed entities into the goal model and to classify goals into hard goals and soft goals, we plan to consider a goal model refinement process, such as one described in [15].

Acknowledgments

This work was supported by JSPS Grants-in-Aid for Scientific Research (No. 17KT0043, No. 20H04167).

References

[1] Beel, J., Gipp, B., Langer, S., Breitinger, C.: Research-paper recommender systems : a literature survey. International Journal on Digital Libraries **17**(4), 305–338 (2016). https://doi.org/10.1007/s00799-015-0156-0

[2] Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., Zimmermann, T.: What makes a good bug report? In: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. pp. 308–318. ACM (2008)

[3] Bird, S., Loper, E., Klein, E.: Natural Language Processing with Python. O'Reilly Media Inc (2009)

[4] Chen, N., Lin, J., Hoi, S.C., Xiao, X., Zhang, B.: Ar-miner: mining informative reviews for developers from mobile app marketplace. In: Proceedings of the 36th International Conference on Software Engineering. pp. 767–778. ACM (2014)

[5] Conneau, A., Kruszewski, G., Lample, G., Barrault, L., Baroni, M.: What you can cram into a single vector: Probing sentence embeddings for linguistic properties. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2018). https://doi.org/10.18653/v1/p18-1198

[6] Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming **20**(1-2), 3–50 (Apr 1993)

[7] Higashi, K., Nakagawa, H., Tsuchiya, T.: Improvement of user review classification using keyword expansion. In: The 30th International Conference on Software Engineering and Knowledge Engineering, Hotel Pullman, Redwood City, San Francisco Bay, California, USA, July 1-3, 2018. pp. 125–130 (2018)

[8] Hoon, L., Vasa, R., Schneider, J.G., Grundy, J., 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)

[9] Kaiya, H., Horai, H., Saeki, M.: Agora: attributed goal-oriented requirements analysis method. In: Proceedings IEEE Joint International Conference on Requirements Engineering. pp. 13–22 (2002)

[10] Lebret, R., Collobert, R.: Word embeddings through hellinger pca. Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (2014). https://doi.org/10.3115/v1/e14-1051

[11] Maalej, W., Nabil, H.: Bug report, feature request, or simply praise? on automatically classifying app reviews. In: Proc. of the 23rd IEEE International Requirements Engineering Conference (RE). pp. 116–125 (Aug 2015). https://doi.org/10.1109/RE.2015.7320414

[12] Maalej, W., Kurtanović, Z., Nabil, H., Stanik, C.: On the automatic classification of app reviews. Requirements Engineering **21**(3), 311–331 (Sep 2016). https://doi.org/10.1007/s00766-016-0251-9

[13] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. pp. 3111–3119. NIPS'13, Curran Associates Inc., USA (2013)

[14] Mylopoulos, J., Chung, L., Nixon, B.: Representing and using nonfunctional requirements: a process-oriented approach. IEEE Transactions on Software Engineering **18**(6), 483–497 (June 1992)

[15] Nakagawa, H., Ohsuga, A., Honiden, S.: A goal model elaboration for localizing changes in software evolution. In: Proc. of 21st IEEE International Requirements Engineering Conference (RE'13). pp. 155 – 164. IEEE CS (2013)

[16] Pagano, D., Maalej, W.: User feedback in the appstore: An empirical study. In: 2013 21st IEEE international requirements engineering conference (RE). pp. 125–134. IEEE (2013)

[17] Palmieri, A., Collet, P., Amyot, D.: Handling regulatory goal model families as software product lines. In: International Conference on Advanced Information Systems Engineering. pp. 181–196. Springer (2015)

[18] Rolland, C., Achour, C.B.: Guiding the construction of textual use case specifications. Data & Knowledge Engineering **25**(1), 125 – 160 (1998). https://doi.org/https://doi.org/10.1016/S0169-023X(97)86223-4

[19] Shimada, H., Nakagawa, H., Tsuchiya, T.: Goal model construction based on user review classification. In: Joint Proceedings of REFSQ-2019 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track co-located with the 25th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2019), Essen, Germany, March 18th, 2019 (2019)

[20] Tripathy, A., Agrawal, A., Rath, S.K.: Classification of sentimental reviews using machine learning techniques. Procedia Computer Science **57**, 821–829 (2015)

[21] van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: Proceedings Fifth IEEE International Symposium on Requirements Engineering. pp. 249–262 (Aug 2001). https://doi.org/10.1109/ISRE.2001.948567

[22] Ward Jr., J.H.: Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association **58**(301), 236–244 (1963)

[23] Yu, E.S.K.: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997. pp. 226–235 (January 1997)