Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2021.013026

ARTICLE

Eliciting Requirements from Stakeholders' Responses Using Natural Language Processing

Mohammed Lafi[1,*], Bilal Hawashin[2] and Shadi AlZu' bi[3]

[1]Department of Software Engineering, Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, 11733, Jordan

[2]Department of Computer Information Systems, Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, 11733, Jordan

[3]Department of Computer Science, Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, 11733, Jordan

[*]Corresponding Author: Mohammed Lafi. Email: lafi@zuj.edu.jo

Received: 24 July 2020 Accepted: 24 December 2020

ABSTRACT

Most software systems have different stakeholders with a variety of concerns. The process of collecting requirements from a large number of stakeholders is vital but challenging. We propose an efficient, automatic approach to collecting requirements from different stakeholders' responses to a specific question. We use natural language processing techniques to get the stakeholder response that represents most other stakeholders' responses. This study improves existing practices in three ways: Firstly, it reduces the human effort needed to collect the requirements; secondly, it reduces the time required to carry out this task with a large number of stakeholders; thirdly, it underlines the importance of using of data mining techniques in various software engineering steps. Our approach uses tokenization, stop word removal, and word lemmatization to create a list of frequently accruing words. It then creates a similarity matrix to calculate the score value for each response and selects the answer with the highest score. Our experiments show that using this approach significantly reduces the time and effort needed to collect requirements and does so with a sufficient degree of accuracy.

KEYWORDS

Software requirements; requirements elicitation; natural language processing

1 Introduction

Requirements can be defined as descriptions of the services that a system provides and the constraints on those services [1].

In other words, they are the system's conditions or capabilities [2]. The process of collecting and documenting system functions and constraints is called "requirements engineering" [3] and a crucial part of it is eliciting requirements to find customers' needs [4].

Engaging stakeholders in requirements elicitation is essential for getting accurate, complete, and consistent requirements [5]. However, eliciting agreed requirements from a large number of stakeholders, each of whom has his or her own concern, is challenging.

Automating specific requirement engineering steps is an exciting area, but there are several associated challenges. Requirements engineering involves manual effort, and domain knowledge as well as methodological expertise and soft skills. High quality of requirements engineering is crucial, since it provides information for later stages of software development. Extraction requirements from natural language text is a problem of particular relevance, especially in the context of large-scale open-source development systems.

Requirements, as well as large parts of the software, are described using natural language or text [6,7]. We can use Text Retrieval (TR) and Natural Language Processing (NLP) can be used to process information represented as text, to achieve many software engineering activities, including requirements analysis [6,7].

In the past, researchers have used many techniques to elicit requirements, including interviews and questionnaires. Although, interview is the most common elicitation technique [8], Wagner et al. [8] have pointed out the popularity of questionnaire as an elicitation technique and suggested that researchers should included it for further study.

The questionnaire elicitation technique may contain closed questions, such as multiple choice and true/false questions, or open questions, such as essay questions. Researchers have previously elicited requirements from stakeholders' responses using questionnaires with closed questions [9,10]; the large number of responses received from open question questionnaire poses its own problems since reading and managing a large quantity of responses takes a very long time.

Therefore, determining the most representative stakeholder responses is somewhat tricky. Although, researchers have proposed many solutions, most of the previous work on the topic does not address the issue of selecting a single stakeholder response that represents all stakeholders' responses. Finding out which response is most representative can reduce the time and effort required to read all replies, and make collecting and eliciting requirements more efficient.

Our proposed approach is a new method that combines the interview technique with the questionnaire technique. Other means and techniques of recording stakeholders' responses such as personal interview and workshops can also be used with our approach.

This study's goal is to propose a requirement collection method "agreed upon requirement" using NLP techniques that can help elicit "agreed upon requirement" from multiple stakeholder responses or opinions. NLP can support the requirements elicitation process by using similarity measurements to select the stakeholder response that best represents the stakeholders' majority opinion.

The approach can be broken down as follows: First of all, domain experts suggest open questions; these questions are then distributed to large number of stakeholders, who are scattered over a large area, as is the case in open-source systems.

These answers are then collected and the most representative response is elicited. This response is the initial requirement used to negotiate with stakeholders when creating the final set of requirements.

In this paper, we apply NLP techniques to identify and resolve an agreed upon requirement from multiple stakeholders' responses. We use and compare four popular similarity measures, and adopting term weighting model proposed by Li et al. [11], we use them to select the representative

answer. As far as we know, these similarity measures and weighting model have not been used previously for selecting the most representative answer.

To conduct our experiment, we created different datasets with different data sizes since there is no sufficient dataset for this domain. To evaluate our method, we used performance (execution time) and accuracy methods, which means that the selected stakeholder response is the most representative of all others.

This paper is divided as follows. After a literature review in Section 2, Section 3 presents the proposed approach: Firstly, we propose an efficient automatic approach to select representative requirements from multiple stakeholders' responses to the same question or issue. Secondly, we conduct various experiments that show our approach is effective in terms of accuracy and performance (execution time) (see Section 4). Thirdly, we provide a tool to the proposed approach (see Section 4). Section 4 presents the experiment, Section 5 presents the discussion, and Section 6 presents the conclusion and implications for future work.

2 Related Work

There has been an increase in the use of TR and NLP in software engineering [6,7]. The use of NLP in software requirements phase was discussed early in [12–16].

Arora et al. [17] previously described how to use natural language techniques to process requirements and check that they conform to requirements templates. In another work, Arora et al. [18] describe how to use natural language technique to extract glossary terms.

Kurtanović et al. [19] use machine learning to classify requirements as functional and non-functional, but they use NLP to tokenize, remove stop words, and lemmatize requirements. In another work, Kurtanović et al. [20] used NLP to analyze user reviews.

Khan et al. [21] propose a framework for prioritizing requirements. Their work focuses on requirements after requirement discovery phase.

Barbosa et al. [22] propose an approach to cluster requirements using NLP techniques such as tokenization, stop word removal, stemming, and similarity measures. They use Vector Space Model (VSM) to represent the requirements. Barbosa et al. [22] have performed an experiment using 50 user stories proposed by Cohn [23]. However, their goal was to cluster and sequence requirements, not to elicit requirements.

Johann et al. [24] propose an approach to extract features from app stores using both app descriptions and user reviews. They apply NLP techniques, such as tokenization, removing stop words, and filtering, to get the set of related terms. They then extract and match the features from both app descriptions and user reviews.

Berry et al. [25] discuss using of NLP in different software engineering tasks. They argue that getting high recall and high precision at the same time is difficult, so the NLP tool is preferable for get high recall instead of high precision.

Bakar et al. [26] propose using NLP techniques to extract a software functionality features from public data such as user reviews.

The approach of Guzman et al. [27] is to extract features from app reviews and assign either positive or negative sentiments to these reviews. The approach consists of a set of steps: Firstly, they extract reviews from Apple app store; then, they process them by tokenization and stop word removal; next, they use the collocation algorithm provided by Natural Language Tool Kit (NLTK) to create features; they then use the Latent Dirichlet Allocation (LDA) a topic modeling

algorithm to assign topic to the extracted features; they also assign a value ranging from 5 (as a positive) to −5 (as a negative); finally, they perform experiments on five app reviews. Their results seem promising.

Sampaio et al. [28] use NLP techniques to identify aspects of corpus-based represented as an unstructured source of requirements.

App stores such as the Apple App Store, Google Play, and Blackberry World, provide assorted information about apps, including app description from development companies, user comments, feedback ratings, price and the number of downloads [29]. Harman et al. [29] call this app store repository mining. They have proposed an approach to extract features from Blackberry World app store, which is composed of the following phases: They first use web crawling tool to extract information from Blackberry World app store; then, they parse the collected information using pattern template; Next, they extract features using a five-step algorithm, in which the features pattern is identified, noise words are filtered out, word frequency and collocation analysis is performed, and features are clustered using a greedy algorithm.

Falessi et al. [30] have studied and evaluated different NLP techniques and their use for equivalent requirements identification. They argue that their results are applicable to similar applications.

Natt och Dag et al. [31] describe an NLP approach that links customer wishes and product requirements. They use vector space to represent requirements and cosine similarity to calculate similarity between customer wishes and product requirements. Their approach is an improvement on previous methods that rely on a keyword search, but they suggest three ways for further improvement: Using different similarity measurements, using the previous discovered similarity between requirements, and incorporating semantics similarity instead of syntax similarity.

Ferrari et al. [32] describe using NLP in requirements engineering from four dimensions: Discipline, dynamism, domain knowledge, and datasets. By discipline they mean using NLP to check the requirements' compliance to standards and the requirements' ambiguity. By dynamism they mean using NLP for requirements tracing and categorization. By domain knowledge they mean using NLP and data-mining for topic modeling of existing requirements and as a recommender system to elicit requirements. Ferrari et al. emphasize the importance of datasets in NLP. They argue that datasets are scare for two reasons: The confidentiality of requirements, and datasets needing human intervention to create them.

Salton et al. [33] emphasize on the importance of including weights in text similarities comparisons.

Li et al. [11] propose an approach to compute the similarity between two sentences. First, they represent the sentences using VSM, using words that include the two sentences. They set the vector as follows: If the word exists in the sentence, they set the corresponding value to 1; otherwise, they set the value to the similarity score between that word and other words. Second, they calculate the similarity between the two vectors using cosine similarity. They also propose a method to take the order of the words into consideration. However, we do not consider the order of the words because it may not be essential in the response.

Okazaki et al. [34] argue that aggregating similarity values simply obtains the sentence similarity of all word pairs.

Kengphanphanit et al. [35] introduced a technique to extract requirements form social media. Also, they classified requirements into functional and non-functional requirements.

Our proposed approach works in a setting in which there is a large number of distributed stakeholders; this idea is not presented in any of the above works. There is also a lack of work aiming to find the representative response from a large number of responses, even though this is essential, and was the motivation behind this study.

3 The Proposed Approach

Our proposed approach consists of four main phases (see Fig. 1). The first phase (the requirements collection process), is collecting requirements from stakeholders. Stakeholders are provided with an online form that containing a set of pre-prepared questions. The different stakeholders' responses to these questions are then collected (see Fig. 2). In the second phase (the requirements preprocessing phase), the approach preprocesses stakeholders' responses, using NLP techniques, to extract the set of terms/and/or words that form the -stakeholders' responses, as described in Subsection 3.1. In the third phase (the requirements selection phase), we use a selection algorithm to choose the stakeholder's response that best represents the majority of other responses, as described in Subsection 3.2. In the fourth phase (the validation phase), we validate the proposed approach's result by comparing our result with the expected result. The validation is done by human coders since this has been suggested by many other researcher [18–20,22].

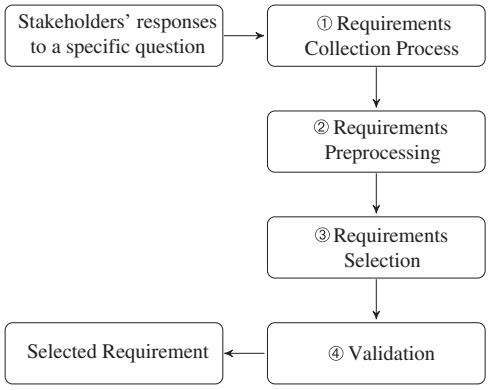

Figure 1: The four main activities of the proposed process: Requirements collection process, requirements preprocessing, requirements selection, and validation

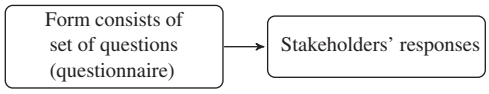

Figure 2: Collection of requirements by getting the stakeholders' responses

Before we go into more detail about the four phases of our approach, it is important to describe the approach's setting. Falessi et al. [30] use a combination of four dimensions in their work: An algebraic representation model, a term weighting model, a term extraction approach, and a similarity measure. Their work shows this combination is effective for finding equivalent requirements in reuse systems, which is relevant to our application in finding the best stakeholder response.

We adopted the approach of Falessi et al. [30]. Furthermore, we use the VSM as representation model, because it has clear semantics, and it has been widely used by other researchers in similar studies. For term extraction, we use a simple term extraction pipeline consisting of the following stages: Tokenization, stop word removal, and lemmatization. For the term weighting model, we use a model proposed by Li et al. [11] and adopted by Guzman et al. [27], Johann et al. [24], and Kurtanović et al. [19]. According to this model, if the term exists, then the approach assigns one as term weight. Otherwise, the model assigns the similarity factor between the term and other terms as term weight. However, our approach does not consider the order of the words; instead, it considers them as a bag of words. Choosing a similarity metric is not prescribed theoretically [36], so we study the performance and accuracy of well-known vector similarity metrics, cosine, dice, Jaccard (the Tanimoto coefficient), and overlap to find the more suitable one in this application domain (see Tab. 1).

Table 1: Similarity metrics and the equations used to calculate them

No.	Similarity metric	Equation used
1	Cosine	$\frac{RV_i \cdot RV_j}{\sqrt{\|RV_i\| \cdot \|RV_j\|}}$
2	Jaccard	$\frac{RV_i \cdot RV_j}{\|RV_i\| \cdot \|RV_j\| - RV_i \cdot RV_j}$
3	Overlap	$\frac{RV_i \cdot RV_j}{min(\|RV_i\|, \|RV_j\|)}$
4	Dice	$\frac{2 \cdot RV_i \cdot RV_j}{\|RV_i\| \cdot \|RV_j\|}$

Our approach is a NLP language model in essence it starts by processing, tokenizing, and lemmatizing the requirements. Then, it represents these requirements as vectors. Also, our approach uses mathematical functions, such as cosine, jaccard, overlap and dicce, to compute the similarity between different vectors that represent requirements. Also, it selects the stakeholder response that represents most other stakeholders' responses using a numerical value.

Falessi et al. [30] adopt a similarity threshold of 0.5, while Johann et al. [24] adopt a similarity threshold of 0.7. The similarity threshold seems to be domain-dependent.

When the similarity threshold increases, more constraints are added to answers to be considered a candidate. Using this threshold would increase the accuracy but would decrease the number of resulting candidate answers.

We performed preliminary sensitive analysis experiments that led us to use 0.7 as a threshold.

3.1 Requirements Preprocessing

The goal of this phase is to create a list of terms appearing in the stakeholders' responses. However, most large systems consist of a large number of stakeholders, so the list will be enormous. Therefore, the approach uses NLP techniques to find the most representative response. To get a list of words/terms, we perform the following steps: first, we identify the words/terms that form each stakeholder's response by using the word tokenization operation that is part of the

NLTK; then, we remove any unnecessary words by employing the stop-words removal operation from the NLTK; next, we normalize each word and/or term by reducing their different forms to a single form, by using the NLTK lemmatization operation[1]; finally, we add words/terms to the words/terms list (see Figs. 3 and 4).

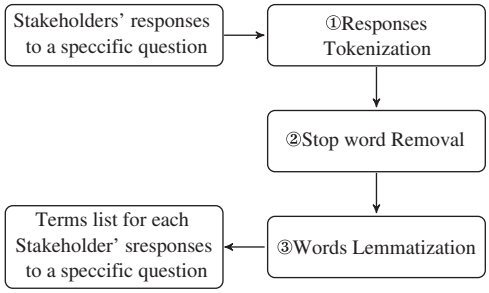

Figure 3: Requirements preprocessing to extract terms from stakeholders' responses

Input: Stakeholders' responses to a specific question
Output: Terms list for each Stakeholder' s response to a specific question
1: Initialize Words/terms list
2: **for all** stakeholders' responses **do**
3: Responses Tokenization
4: Stop word Removal
5: Words Lemmatization
6: add words/terms to Words/terms list
7: **end for**

Figure 4: Detailed steps of how terms are extracted from stakeholders' responses

3.2 Requirements Selection

The goal of this phase is to find the central stakeholder response that can represent most of other stakeholder responses. Similarity between requirements has been used by many researchers to elicit requirements [13], extract glossary terms [18], analyze user reviews [19], classify requirements [20], cluster requirements [22], extract functionality features from app stores [24], and extract functionality features from public data [26]. Therefore, we used similarity between stakeholders' responses to pick the most representative response. To achieve this goal, we performed four steps (see Figs. 5 and 6).

The first step is create the Lexical Semantic Vector (LSV) based on the words/terms list obtained from the previous stage. LSV contains all the distinct words in all stakeholders' responses for a specific question. Let the stakeholder responses to a question k be the set R, where $R = \{R_1, R_2, \ldots, R_n\}$. Then, $LSV = \{terms \in R_1 \cup terms \in R_2 \cup \ldots \cup terms \in R_n\}$. Next, we created an LSV response vector (RV) for each stakeholder response as follows:

$$RV_k(i) = \begin{cases} 1 & \text{if } term_i \in R_k \\ similarity(term_i, \forall term \in R_k) & \text{if } term_i \notin R_k \end{cases} \quad (1)$$

[1] We use lemmatization instead of stemming because lemmatization produces terms that are easier to understand than stemming.

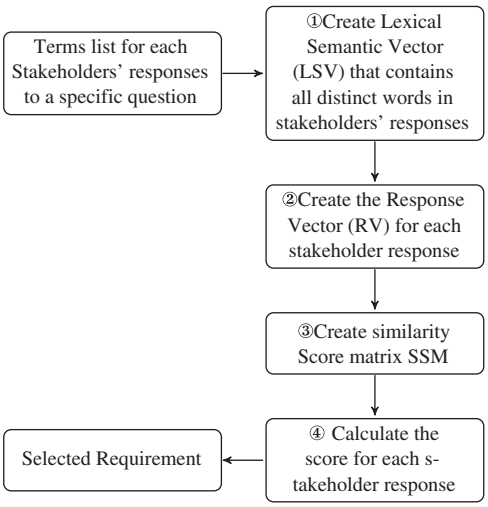

Figure 5: Selection of the most representative stakeholder' response

Input: stakeholders' responses to a certain question (R) and Terms list for each Stakeholder's response to a specific question
Output: score for each stakeholders' response and best stakeholder response with highest score
{Create Lexical Semantic Vector (LSV) that contains all distinct words in stakeholders' responses}
$LSV = \{terms \in R_1 \cup terms \in R_2 \cup \cdots \cup terms \in R_n\}$
{ Create the Response Vector (RV) for each stakeholder response as illustrated in Figure 7}
{Create Similarity Score Matrix SSM as illustrated in Figure 8}
{Calculate the score for each stakeholder response}
for $i = 1$ to N **do**
  $SV[i] = \sum_{j=1}^{n} M[i,j]$
end for
{select stakeholder response with the highest score}
$TheHighestScore = \max_{1 \le j \le n} SV[i]$
select stakeholder response with $TheHighestScore$

Figure 6: Detailed steps on how to select the most representative responses

Fig. 7 illustrates the detailed steps of creating the response vector (RV).

Input: Lexical Semantic Vector (LSV)
Output: Response Vector (RV)
{ Create the Response Vector (RV) for each stakeholder response }
for all stakeholders' responses to a certain question (R) **do**
  for all term in LSV **do**
    if $LSV[i] \in R_i$ **then**
      $RV[i] = 1$
    else
      $RV[i] = similarity(LSV[i], R_i)$
    end if
  end for
end for

Figure 7: Creating the response vector (RV) for each stakeholder response

We used the similarity function provided by the NLTK and implemented in Python to calculate the similarity between two terms $term_i$ and $term_j$. The similarity function in NLTK depends on NLTK wordnet. To compare all stakeholders' responses, we created a similarity matrix (SM), which stores the similarity between all response pairs. The similarity between two stakeholders' responses represented as response vectors as RV_i and RV_j is calculated using one equation from Tab. 1 depending on the similarity metric used as illustrated in Fig. 8. The resulting similarity matrix is illustrated as follows:

$$SM = \begin{bmatrix} m_{11} & m_{12} & m_{13} & \dots & m_{1n} \\ m_{21} & m_{22} & m_{23} & \dots & m_{2n} \\ & \dots & \dots & \dots & \\ m_{n1} & m_{n2} & m_{n3} & \dots & m_{nn} \end{bmatrix} \tag{2}$$

where m_{ij} is the similarity between RV_i and RV_j computed using an equation from Tab. 1 and n is the number of stakeholders' responses.

Input: Response Vector (RV)
Output: Create Similarity Score Matrix SSM
1: {Create Similarity Score Matrix SSM}
2: {N : number of stakeholders' responses}
3: **for** $i = 1$ to N **do**
4: **for** $j = 1$ to N **do**
5: $similarity = equation from Table\ 1.$
6: **if** $similarity > threshold$ **then**
7: $SM[i, j] = similarity$
8: **else**
9: $SM[i, j] = 0$
10: **end if**
11: **end for**
12: **end for**

Figure 8: Creating similarity score matrix SSM

Okazaki et al. [34] state that sentence similarity can be calculated by aggregating the similarity values of all words pairs. Therefore, we used a similar approach calculating the similarity between all stakeholders' responses by adding the similarity values between stakeholders' responses pairs. To do this, we added together the values in each similarity matrix row. We propose a score vector (SV) to hold values of the summation similarity values. We propose the calculation of SV entries as follows:

$$SV[i] = \sum_{j=1}^{n} M[i, j] \tag{3}$$

After calculating the SV, the stakeholder response corresponding to the maximum value in the SV is selected as the response that can best represent most stakeholders' responses.

4 Implementation and Experiment

We conducted an experiment to demonstrate the applicability of the proposed approach to select the most representative stakeholder response. This can be done by ensuring that the

proposed approach selects the requirement that has the highest calculated score depending on the similarity of each stakeholder response to all other responses and compare it with the expected result determined by human coders who determine the result using manual interpretation.

The dataset used in the experiment is explained in Subsection 4.1 and the selection of programming language is described in Subsection 4.2. The experiment is described in Subsection 4.3 and results are described in Subsection 4.4.

4.1 Dataset Generation

A dataset is a set of requirements that is annotated with semantic information. In our study, this semantic information indicates whether the requirement represents all other requirements. Finding and using datasets in software requirements engineering is an essential and challenging task [32].

To validate our study, we need a dataset related to the elicited requirements from multiple stakeholders. There was a lack of sufficient datasets in this domain, so we prepared a set of ten questions, each with many possible answers, about a university student registration system. The stakeholders can answer these questions using natural language. All stakeholders (students) participated voluntarily, and it was not an academic assignment. Since the criterion of stakeholder eligibility is that the stakeholder knows the business domain, all students were eligible to participate. The list of questions is shown in Tab. 2.

We collected four 256-answer datasets and combined them to create six 512-answer datasets; we then combined the four 256-answer datasets to create 1024-answer dataset. We also collected another 1024-answer dataset. Tab. 3 shows a sample of the collected answers.

Table 2: Sample of questions in a questionnaire about the university registration system

No.	QText
1	How would you prefer the distribution of the system?
2	What type of network would you prefer to be used in the system?
3	Which authentication method do you prefer?
4	Which way do you prefer to get your grades record?
5	Who will assign and confirm the students grades?
6	What the operating systems would you like the system to operate on?
7	Who is responsible for editing the course info?
8	What is the maximum budget that is devoted?
9	What is the maximum number of months you expect the system development will last?
10	How many messages would like to receive form the system every day?

4.2 Programming Language Selection

We implemented the proposed approach using Python, which has many packages that support NLP. We used Python' NLTK package, which is commonly used in NLP. The code and datasets are available through GetHub repository.

4.3 The Experiment

To demonstrate and validate our study, we carried out the following experiment. First, we prepared a set of questions concerning the system functions based on previous domain knowledge.

These questions were then answered by multiple stakeholders using natural language. For example, if the question was, "Who will assign and confirm the students' grades? " the possible stakeholders' responses could be, "The instructor assigns and confirms the grades," "The registrar assigns and confirms the grades," "The dean assigns and confirms the grades," or "The department chair assigns and confirms the grades." A sample of the stakeholders' responses are shown in the second column in Tab. 3.

Table 3: Four stakeholders' responses and list of words for each of them after pre-processing

No.	Stakeholders' responses	List of words
1	The system should be centralized and the network should be peer-to-peer and the authentication system should be username and password and users prefer to get his/her grades from registrar and student prefer instructor to assign and confirm grades. Also, users prefer to use windows instructor will be responsible for editing course information the budget of this project should be less than 10000 The duration of the development should be last less than six months The user can receive no messages at all.	'duration,' 'instructor,' 'user,' 'budget,' 'user,' 'window,' 'authentication,' 'editing,' '10000,' 'last,' 'prefer,' 'month,' 'get,' 'course,' 'password,' 'six,' 'project,' 'registrar,' 'username,' 'development,' 'also,' 'responsible,' 'message,' 'receive,' 'use,' 'student,' 'information,' 'le,' 'confirm,' 'network,' 'grade,' 'assign,' 'peer,' 'centralized,' 'system'
308	The system should be decentralized and the network should be peer-to-peer and the authentication system should be username and password and users prefer to get his/her grades from online system and student prefer instructor to assign and confirm grades. Also, users prefer to use Mac OS registrar will be responsible for editing course information the budget of this project should be between 10000 and 20000 The duration of the development should be last less than twelve months. The user can receive less than three messages.	'duration,' 'instructor,' 'user,' 'twelve,' 'budget,' 'user,' 'decentralized,' 'authentication,' 'editing,' '10000,' 'last,' 'prefer,' 'month,' 'get,' 'mac,' 'password,' 'course,' 'three,' 'project,' 'registrar,' 'username,' 'development,' 'also,' 'message,' 'responsible,' 'receive,' 'online,' 'use,' 'student,' 'information,' 'le,' 'confirm,' 'network,' 'o', 'grade,' 'assign,' '20000,' 'peer,' 'system'
555	The system should be decentralized and the network should be client server and the authentication system should be one time SMS message and users prefer to get his/her grades from online system and student prefer registrar to assign and confirm grades. Also, users prefer to use Mac OS registrar will be responsible for editing course information the budget of this project should be between 10000 and 20000 The duration of the development should be last less than eighteenth months. The user can receive less than three messages.	'duration,' 'user,' 'one,' 'budget,' 'user,' 'time,' 'decentralized,' 'authentication,' 'editing,' '10000,' 'last,' 'prefer,' 'month,' 'get,' 'mac,' 'course,' 'message,' 'three,' 'project,' 'registrar,' 'development,' 'also,' 'message,' 'responsible,' 'receive,' 'online,' 'use,' 'student,' 'information,' 'le,' 'eighteenth,' 'confirm,' 'network,' 'o', 'grade,' 'assign,' '20000,' 'server,' 'sm,' 'system,' 'client'
788	The system should be decentralized and the network should be client server and the authentication system should be iris recognition and users prefer to get his/her grades from by email and student prefer department chair to assign and confirm grades. Also, users prefer to use linux. Dean will be responsible for editing course information the budget of this project should be between 20001 and 50000. The duration of the development should be last less than twenty four months. The user can receive less than five messages.	'duration,' 'user,' 'budget,' 'dean,' 'user,' 'linux,' 'lastless,' 'twenty,' 'decentralized,' 'authentication,' 'editing,' 'four,' 'five,' 'prefer,' 'month,' 'get,' 'course,' 'email,' 'iris,' 'project,' 'development,' 'also,' 'message,' 'responsible,' 'receive,' 'chair,' 'use,' 'student,' 'information,' 'le,' '20001,' 'confirm,' 'network,' 'grade,' 'assign,' 'recognition,' 'server,' 'department,' 'system,' '50000,' 'client'

We then decided which answer was the best representation of all the others by selecting the stakeholder response with the highest percentage of stakeholder agreement for each question' answer that combines each stakeholder' response.

Two groups of human coders, each composed of three people, used this method to validate its result. To avoid bias, the two groups performed their inspections separately, before completing the work. We used majority voting in cases of disagreement.

Next our approach decided which response was the most representative of all other answers.

Finally, the two human-coder groups compared their expected result with the approach's result and verified that the approach performed correctly.

The approach worked as follows. After recording the stakeholders' responses, our approach starts the reprocessing stage, in which the stakeholders' responses are processed through three steps: Tokenization, removal of stop words, and lemmatization. The result of this stage is a list of words for each stakeholder response. The third column in Tab. 3 shows a sample of lists of words.

Depending on the final word list, the approach creates an RV or each stakeholder response. The length of the RV is the number of words, and the dimensions of the RV-matrix are the number of words by the number of responses, depending on the dataset used. The similarity matrix is then created. The similarity matrix dimensions are the number of responses multiplied by the number of responses and values ranging between 0 and 1. (Because of their large sizes, both the RV and the similarity matrix cannot be shown here.) Next, the approach creates the SV using Eq. (2).

4.4 Results

Tabs. 5–8 show the proposed approach's performance and accuracy using cosine, Jaccard, Overlap and Dice similarity metrics, respectively.

Tab. 4 shows a comparison of the approach's performance and accuracy using different similarity techniques on different datasets.

As we can see from Tab. 4 and Figs. 9 and 10,

Table 4: Performance (execution time) and accuracy of different similarity metrics with an increasing dataset size

#	Size	Similarity metric	Time	Accuracy (%)
1	256	Cosine	32.08	75
2	256	Jaccard	34.45	75
3	256	Overlap	34.33	35
4	256	Dice	32.26	75
5	512	Cosine	69.36	75
6	512	Jaccard	76.71	75
7	512	Overlap	80.92	40
8	512	Dice	70.9	75
9	1024	Cosine	161	80
10	1024	Jaccard	165.95	80
11	1024	Overlap	164	40
12	1024	Dice	164	80

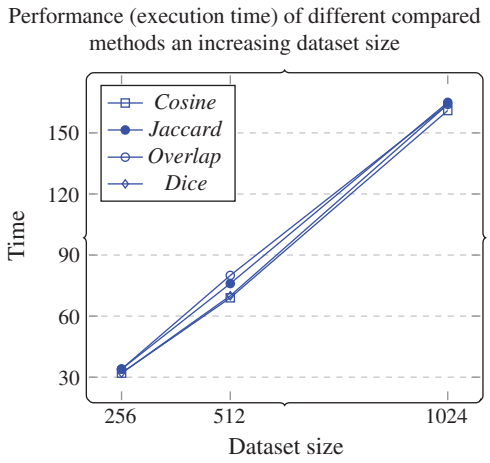

Figure 9: Performance (execution time) of different compared methods with an increasing dataset size

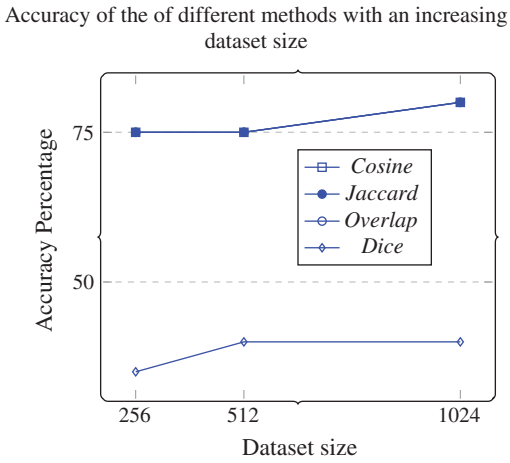

Figure 10: Accuracy of different compared methods with an increasing dataset size

- The approach selects the responses that best represent the stakeholders' responses in up to 80% of cases as the dataset size increases. However, the performance (execution time) increases as dataset size increases.
- The best performance achieved by the approach was the one using cosine similarity metric, which is compatible with results achieved in other studies [30].
- This approach's accuracy was similar using either cosine, Jaccard, or dice, and the worst accuracy achieved by the approach used an overlapping similarity metric.

The two human-coder groups verified that the response selected by the approach is the correct response and that the approach performed correctly.

Table 5: Cosine similarity metric performance (execution time) and accuracy with an increasing dataset size

No.	Size	Time	Accuracy (%)	Words
1	256	34	70	76
2	256	32.7	80	74
3	256	31.3	90	73
4	256	30.3	60	72
5	512	71.4	80	78
6	512	72.1	70	79
7	512	69.3	70	78
8	512	68.3	80	76
9	512	67.9	70	75
10	512	67.2	80	75
11	1024	175	70	79
12	1024	147	90	68

Table 6: Dice similarity metric performance (execution time) and accuracy with an increasing dataset size

No.	Size	Time	Accuracy (%)	Words
1	256	34.34	70	76
2	256	32.8	80	74
3	256	31.6	90	73
4	256	30.3	60	72
5	512	72	80	78
6	512	72.6	70	79
7	512	69.7	70	78
8	512	68.6	80	76
9	512	68.1	70	75
10	512	69	80	75
11	1024	176	70	79
12	1024	152	90	68

Table 7: Overlap similarity metric performance (execution time) and accuracy with an increasing dataset size

No.	Size	Time	Accuracy (%)	Words
1	256	36.5	40	76
2	256	36.8	30	74
3	256	33	30	73
4	256	31	40	72
5	512	76.6	40	78
6	512	84.3	40	79
7	512	81	40	78
8	512	81.55	40	76
9	512	79.14	40	75
10	512	82.9	80	75
11	1024	176	40	79
12	1024	152	40	68

Table 8: Jaccard similarity metric performance (execution time) and accuracy

No.	Size	Time	Accuracy (%)	Words
1	256	35	70	76
2	256	36	80	74
3	256	34	90	73
4	256	32.8	60	72
5	512	77.5	70	76
6	512	78.8	80	74
7	512	77	90	73
8	512	77	60	72
9	512	73	70	76
10	512	77	80	74
11	1024	177.5	90	73
12	1024	154.4	60	72

5 Discussion and Future Work

Our approach intended to find the most representative stakeholders' response and followed a systematic method to achieve this goal. Achieving this goal will enhance the process of eliciting and collecting requirements in large and diverse stakeholders' environments. Achieving this goal will also help software engineers and system analysts to find out the opinion of the majority of stakeholders. Knowing this will help enormously help in developing applications suitable for stakeholders.

Traditional elicitation techniques such as interview and focus groups and more advanced technique such as storyboard and prototyping are effective techniques to collect requirements from stakeholders. However, these traditional techniques suffer from the need of face-to-face interaction with stakeholders. In large scale systems and distributed system the direct interaction is impossible. Therefore, employing our proposed approach can address this shortcoming enabling developers to collect requirements from large number of stakeholders in distributed systems.

We choose using NLP approaches to process requirements instead of traditional approaches for the following reasons. The accuracy of our approach is better than the traditional approach in finding the most representative stakeholders' response. Also, our approach is more scalable than the traditional approach in terms that the number of stakeholders and number of requirements processed can be increased. In addition, our proposed approach is more robustness than traditional approach in terms our proposed approach performance is stable despite if the requirements have some noise or some of stakeholders provide wrong answers.

The size of the data used ranged from 256 to 1024 elements. We understand that the larger the dataset the better; nevertheless, we believe that the datasets used are adequate to validate the proposed approach' s accuracy. While we believe that there is a wide variety of representations of requirements, and that each system has its own peculiarities, we argue that our approach can be applied to other datasets. We encourage other researcher to use our method on different datasets of different sizes.

The final words list, explained in Subsection 4.3, shows a number of strange words "o", "le," and "sm." These words are the result of using the NLTK's stemmer. Enhancing this stemmer is beyond the scope of this paper.

The work we have carried out here could be extended in various directions. For example, we could consider more similarity measures and compare them with the used similarity measures. We could also use text summarization to further improve our method's accuracy Our work in this study aims to encourage researchers to perform their own studies on this topic, in which more datasets are collected; this is an essential element in the experiment. In the future, we plan to consider sentiment analysis of stakeholder responses to see if stakeholders agree or disagree with the suggested requirement. Further evaluation measurements could be used in further study.

6 Conclusion

In this paper, we have demonstrated how our approach collected user requirements from stakeholders' responses represented as text. The approach converts user responses, represented as text, into terms using NLP techniques before applying similarity measures to find the most representative answer. We carried out an experiment to show our approach's accuracy and effectiveness. Using this approach enhanced the process and reduced the time needed to collect the agreed requirements from multiple stakeholders.

Funding Statement: The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Sommerville, I. (2015). *Software engineering*, 10th edition. London, UK: Pearson.
2. IEEE Standards Coordinating Committee and Others (1990). *IEEE standard glossary of software engineering terminology (IEEE std 610.12-1990)*. Los Alamitos, CA: IEEE Computer Society. pp. 169.
3. Zave, P. (1997). Classification of research efforts in requirements engineering. *ACM Computing Surveys, 29(4),* 315–321. DOI 10.1145/267580.267581.
4. Laplante, P. A. (2013). *Requirements engineering for software and systems*, 2nd edition. Boca Raton, Florida, USA: CRC Press.
5. El Emam, K., Koru, A. G. (2008). A replicated survey of it software project failures. *IEEE Software, 25(5),* 84–90. DOI 10.1109/MS.2008.107.
6. Haiduc, S., Arnaoudova, V., Marcus, A., Antoniol, G. (2016). The use of text retrieval and natural language processing in software engineering. *2016 IEEE/ACM 38th International Conference on Software Engineering Companion*, Austin, Texas, USA.
7. Arnaoudova, V., Haiduc, S., Marcus, A., Antoniol, G. (2015). The use of text retrieval and natural language processing in software engineering. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. Florence, Italy.
8. Wagner, S., Fernández, D. M., Felderer, M., Vetrò, A., Kalinowski, M. et al. (2019). Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology, 28(2),* 1–48. DOI 10.1145/3306607.
9. Lafi, M., Abdel Qader, A. (2017). A novel dynamic integrated model for automated requirements engineering process. *International Journal of Computer Applications in Technology, 56(4),* 292–300. DOI 10.1504/IJCAT.2017.089084.
10. Lafi, M., Abdel Qader, A. (2016). A novel automated requirements prioritization and selection model based on requirements weights and stakeholder importance. *International Journal on Information Technology, 4(4),* 105–109.
11. Li, Y., McLean, D., Bandar, Z. A., O'shea, J. D., Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering, 18(8),* 1138–1150. DOI 10.1109/TKDE.2006.130.

12. Ryan, K. (1993). The role of natural language in requirements engineering. *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, CA, USA.
13. Goldin, L., Berry, D. M. (1997). Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation. *Automated Software Engineering, 4(4),* 375–412. DOI 10.1023/A:1008617922496.
14. Kweon, W., Kang, S., Hwang, J., Yu, H. (2020). Deep rating elicitation for new users in collaborative filtering. *WWW '20,* Taipei, Taiwan: Association for Computing Machinery.
15. Omri, S., Sinz, C. (2020). Deep learning for software defect prediction: A survey. *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops. ICSEW'20*, Seoul, Korea: Association for Computing Machinery.
16. Lewellen, S. (2020). Identifying key stakeholders as part of requirements elicitation in software ecosystems. *Proceedings of the 24th ACM International Systems and Software Product Line Conference,* Vol. B. Montreal, QC, Canada: Association for Computing Machinery.
17. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F. (2015). Automated checking of conformance to requirements templates using natural language processing. *IEEE Transactions on Software Engineering, 41(10),* 944–968. DOI 10.1109/TSE.2015.2428709.
18. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F. (2017). Automated extraction and clustering of requirements glossary terms. *IEEE Transactions on Software Engineering, 43(10),* 918–945. DOI 10.1109/TSE.2016.2635134.
19. Kurtanovic, Z., Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. *2017 IEEE 25th International Requirements Engineering Conference*, Lisbon, Portugal.
20. Kurtanovic, Z., Maalej, W. (2017). Mining user rationale from software reviews. *2017 IEEE 25th International Requirements Engineering Conference*, Lisbon, Portugal.
21. Khan, S. U. R., Lee, S. P., Dabbagh, M., Tahir, M., Khan, M. et al. (2016). Repizer: A framework for prioritization of software requirements. *Frontiers of Information Technology & Electronic Engineering, 17(8),* 750–765. DOI 10.1631/FITEE.1500162.
22. Barbosa, R., Januario, D., Silva, A. E., Moraes, R., Martins, P. (2015). An approach to clustering and sequencing of textual requirements. *2015 IEEE International Conference on Dependable Systems and Networks Workshops*, Rio de Janeiro, Brazil.
23. Cohn, M. (2004). *User stories applied: For agile software development*. 1st edition. USA: Addison-Wesley.
24. Johann, T., Stanik, C., Alizadeh, B., A., M., Maalej, W. (2017). Safe: A simple approach for feature extraction from app descriptions and app reviews. *2017 IEEE 25th International Requirements Engineering Conference*, Lisbon, Portugal.
25. Berry, D., Gacitua, R., Sawyer, P., Tjong, S. (2012). The case for dumb requirements engineering tools. *International Working Conference on Requirements Engineering: Foundation for Software Quality,* pp. 211–217. Essen, Germany: Springer.
26. Bakar, N. H., Kasirun, Z. M., Salleh, N. (2015). Terms extractions: An approach for requirements reuse. *2015 2nd International Conference on Information Science and Security*, Seoul, Korea.
27. Guzman, E., Maalej, W. (2014). How do users like this feature? A fine grained sentiment analysis of app reviews. *2014 IEEE 22nd International Requirements Engineering Conference*, Karlskrona, Sweden.
28. Sampaio, A., Loughran, N., Rashid, A., Rayson, P. (2005). Mining aspects in requirements. *Early Aspects 2005: Aspect-Oriented Requirements Engineering and Architecture Design Workshop*, Chicago, Illinois, USA.
29. Harman, M., Jia, Y., Zhang, Y. (2012). App store mining and analysis: MSR for app stores. *2012 9th IEEE Working Conference on Mining Software Repositories*, Zurich, Switzerland.
30. Falessi, D., Cantone, G., Canfora, G. (2013). Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. *IEEE Transactions on Software Engineering, 39(1),* 18–44. DOI 10.1109/TSE.2011.122.
31. Natt och Dag, J. N., Regnell, B., Gervasi, V., Brinkkemper, S. (2005). A linguistic-engineering approach to large-scale requirements management. *IEEE Software, 22(1),* 32–39. DOI 10.1109/MS.2005.1.

32. Ferrari, A. F., Esuli, A., Gervasi, V., Gnesi, S. (2017). Natural language requirements processing: A 4D vision. *IEEE Software, 34(6),* 28–35. DOI 10.1109/MS.2017.4121207.
33. Salton, G., Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management, 24(5),* 513–523. DOI 10.1016/0306-4573(88)90021-0.
34. Okazaki, N., Matsuo, Y., Matsumura, N., Ishizuka, M. (2003). Sentence extraction by spreading activation through sentence similarity. *IEICE Transactions on Information and Systems, 86(9),* 1686–1694.
35. Kengphanphanit, N., Muenchaisri, P. (2020). Automatic requirements elicitation from social media (ARESM). *CCCIS 2020,* Ho Chi Minh City, Vietnam: Association for Computing Machinery.
36. Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. USA: Addison-Wesley Longman Publishing Co., Inc.