Available online at www.sciencedirect.com

ScienceDirect

Information and Software Technology 49 (2007) 1128–1140

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

Using classification techniques for informal requirements in the requirements analysis-supporting system

Youngjoong Ko [a,*], Sooyong Park [b], Jungyun Seo [b], Soonhwang Choi [b]

[a] *Department of Computer Engineering, Dong-A University, 840, Hadan 2-dong, Saha-gu, Busan 604-714, Republic of Korea*

[b] *Department of Computer Science/Program of Integrated Biotechnology, Sogang University, Seoul 121-742, Republic of Korea*

Received 15 April 2006; received in revised form 5 November 2006; accepted 20 November 2006
Available online 4 January 2007

Abstract

In order to efficiently develop large-scale and complicated software, it is important for system engineers to correctly understand users' requirements. Most requirements in large-scale projects are collected from various stakeholders located in various regions, and they are generally written in natural language. Therefore, the initial collected requirements must be classified into various topics prior to analysis phases in order to be usable as input in several requirements analysis methods. If this classification process is manually done by analysts, it becomes a time-consuming task. To solve this problem, we propose a new bootstrapping method which can automatically classify requirements sentences into each topic category using only topic words as the representative of the analysts' views. The proposed method is verified through experiments using two requirements data sets: one written in English and the other in Korean. The significant performances were achieved in the experiments: the 84.28 and 87.91 F1 scores for the English and Korean data sets, respectively. As a result, the proposed method can provide an effective function for an Internet-based requirements analysis-supporting system so as to efficiently gather and analyze requirements from various and distributed stakeholders by using the Internet.

Keywords: Requirements engineering; Requirements classification; Internet-based application system; Similarity measure

1. Introduction

As software becomes more complicated and large-scale, the requirements analysis plays an increasingly important role in the software development. The globalization of the software industry has caused the distributed requirements environment, including various stakeholders from various regions [1]. Since modern projects require fast delivery, and the changes of their requirements occur more frequently, requirements activities are not terminated after developing requirements specification; they must be continued throughout the whole process of system development [2].

Since large-scale projects have usually had difficulty handling many various requirements from the distributed environment, interactions among stakeholders can be an important success factor. Specifically, the participation of client groups is very important in multi-sites development [3,4]. As the Internet is widely used, it has become a convenient interface for various stakeholders. If an Internet-based requirements analysis tool is developed, it can be exploited as an efficient method to overcome spatial and temporal barriers when collecting requirements from various clients in requirements elicitation processes [5,6]. Nevertheless, there are still no efficient tools for automated acquisition and analysis of early software requirements [7]. In addition, since early informal requirements are written in natural language, many problems occur in their handling. However, the informal requirements are still regarded as one of the most important communication

* Corresponding author. Tel.: +82 51 200 7782; fax: +82 51 200 7783.
E-mail addresses: yjko@dau.ac.kr (Y. Ko), sypark@sogang.ac.kr (S. Park), seojy@sogang.ac.kr (J. Seo), soonhwang@sogang.ac.kr (S. Choi).

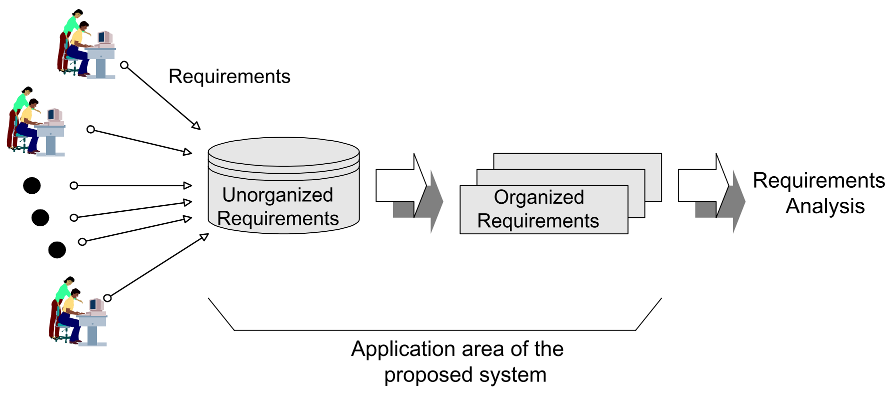


Fig. 1. Requirements analysis process using the proposed system.

mediums between developers and their clients [8]. The lack of formality, organization, and definiteness from informal requirements makes it difficult to efficiently handle them in early software development steps.

Informal requirements collected in the early requirements elicitation phase should be classified into their identification, priority, criticality, feasibility, risk, source, and type before analyzing them or beginning negotiation processes [13]. For example, if ones perform a trade-off analysis based on system quality and cost, requirements must be classified into each quality and cost attribute before the requirements can be used in negotiation with stakeholders. Because new requirements and viewpoints are continuously created and frequently changed, the classification of the requirements is a difficult task in negotiation, in trade-off analysis, and in viewpoints approaches such as WinWin Negotiation model [9,10], ABAS (Attribute Based Architecture Style) [11], and Viewpoints approaches [12].

Since the classification of the requirements from complex and large-scale projects is a difficult and time-consuming task, automatic requirements classification techniques using natural language processing can be used for efficient requirement analysis tools. Many natural language processing techniques have been studied to support automatic requirements analysis [18–22,24]. These techniques have used various methods including, but not limited to: sliding windows, similarity measurement, and semantic networks for requirements analysis. Accordingly, we have attempted to apply natural language processing techniques to requirements analysis [20,24]. In [20], the requirements-analysis supporting system is presented to alleviate some critical problems such as conflict, inconsistency, and ambiguity. Using text classification techniques for requirements organization was proposed in [24]. This paper presents a more advanced requirements classification technique, which can automatically produces keyword lists and classify requirements using only topic word input.

This paper proposes a new automatic requirements classification method which is applied to an Internet-based requirements analysis-supporting system. Requirements are collected from the distributed environment and they are stored in a database for further analysis. If a great number of requirements are generated from complicated projects and the distributed environment, they then require an efficient classification method to reorganize the large number of requirements. Using the proposed analysis system, requirements analysts can automatically classify the collected informal requirements into several views, or topics, by using only the topic word of each view. Since the result of classification can be used as the input of various analysis methods, the proposed method can supply a good basis for efficient requirements analysis as shown in Fig. 1. Moreover, it can also reduce difficulties in analyzing informal and unorganized requirements which are written in natural language. Although there are many promising studies to automatically organize requirements by using natural language processing techniques, our research is original and advanced in that it can automatically classify requirements into several views by using only topic words from analysts.

The rest of this paper is organized as follows: Section 2 describes related work; Section 3 explains the proposed classification technique for an Internet-based requirements analysis system in detail; Section 4 reports experimental results; Section 5 covers the time savings and the development of the proposed system; last, Section 6 concludes with future work and a summary.

2. Related work

2.1. Requirements elicitation and analysis

Requirements elicitation has been performed by various methods such as introspection, interview, questionnaire, protocol analysis, and brain storming [14]. In addition, calling, remote meeting, net meeting, and the other Internet-based techniques are usually used in projects with distributed stakeholders for the requirements elicitation [1]. In order to efficiently analyze the requirements collected by the requirements elicitation, they must first be classified into various types before requirements analysis [13]. Requirements analysis includes many kinds of activities such as state-mode analysis, use-case analysis, trade-off analysis, and negotiation. Various requirements analysis methods such as WinWin, ABAS, and the Viewpoints method, used to support these activities, require classified requirements as input data. Although these analysis methods need to classify initial collected requirements, they do not have efficient tools for automated requirements classification yet.

Boehm [10] proposed the **WinWin** approach, as it provides a negotiation model between development teams and clients for improving requirements consistency and completeness. In the WinWin approach, various stakeholders have their own requirements called 'win conditions'. These 'win conditions' can conflict with each other. The conflicts can be solved by the proposed WinWin Negotiation model wherein distributed stakeholders are able to get 'win conditions' accepted by all stakeholders. However, these 'win conditions' must be classified into suitable topics in advance of negotiation. For example, 'win conditions' about quality attributes are classified into any quality attribute type such as performance, usability, or reliability, but the WinWin approach does not provide automated classification tools or methods [15].

Klein and Kazman [11] presented the **ABAS** (Attribute Based Architectural Style) method for building architectural styles based on quality attributes. Architectural styles can be developed from quality attributes after the requirements analysis phase. The ABAS method provides guidelines for selecting and building any architectural style based on quality attributes. Quality requirements are usually written in natural language, and they should be classified into quality attributes for this method. After system analysts analyze the quality attributes from these classified quality requirements, they can derive architectural styles from the analysis [11].

Sommerville and Sawyer [12] proposed the **Viewpoints** approach, from which requirements are derived not from a single viewpoint but from several viewpoints. Various viewpoints can be created: the customer viewpoint, the security viewpoint, the marketing viewpoint, the personnel viewpoint, etc. Requirements must be reorganized by several viewpoints in the requirements elicitation phase in order to analyze projects using this approach.

2.2. Applying natural language processing techniques to requirements engineering

There have been many studies for linguistic-engineering approaches to build automatic requirements analysis-supporting systems. The role of natural language processing in requirements engineering was discussed in [16] and the overview of linguistic-engineering approaches to large-scale requirements management was presented in [17]. Various attempts have been made to use automated techniques to assist in the analysis of requirements written in natural language:

Parmar [18] proposed the TTC (Two-Tiered Clustering) algorithm which indexes and clusters requirements specifications. First, it identifies the verbs and keywords of each requirement, and then it clusters these requirements by functionality. Next, each cluster is subdivided using cosine similarity among clusters.

Natt och Dag et al. [19] presented an approach based on statistical techniques for the similarity analysis of natural language requirements aimed at identifying duplicate requirement pairs in market-driven software development.

Park et al. [20] described a system that uses a sliding window model and a parser to support the analysis of requirements using a similarity measuring technique.

Cybulski and Reed [7] presented an elicitation method and a supporting management tool that help analysts to analyze and refine requirements by using a parser, semantic networks, a domain-mapping thesaurus, and faceted classification schemes to allow the proper formalization of requirements written in natural language.

Fantechi et al. [21] discussed the application of analysis techniques based on a linguistic approach to detect defects related to such an inherent ambiguity.

Chen et al. [22] presented ideas where concepts in texts from electronic meetings are automatically classified by using automatic indexing, cluster analysis, and Hopfield net classification.

These studies can be roughly classified into two different text-processing approaches: statistic and linguistic approaches. The statistical approaches are used in this paper. There are several reasons that we use these approaches [20,23]:

- Cost: statistical approaches can be built automatically. Linguistic approaches are still regarded as expensive ones to implement, and they are not always more effective than well-executed statistical approaches.
- Transportability: statistical approaches can be easily applied for other application domains while linguistic approaches need to rework for being applied other application domains.
- Scalability: the repository in statistical approaches can be easily updated when new components are inserted.
- Finally, requirements to be analyzed in the proposed system are collected from the Internet, and they are written in natural language. To make matters worse, many requirements are short-worded and poorly written. Therefore, it is hard to use linguistic tools, such as parser and thesaurus, for requirement analysis.

3. A requirements classification technique

3.1. The overview of the proposed requirements classification system

In order to automatically classify collected requirements, a bootstrapping framework is developed and applied to our requirement classification system [24]. The requirements classification system requires only topic words as the representative of an analysts' view in a classification process. The bootstrapping framework consists of the following steps:

1. *Preprocessing*: each sentence is separated from all collected requirements, and each content word is extracted from the separated sentence.
2. *Constructing a set of centroid-sentences as training data for each topic category*:
   - The keywords of each topic are extracted.
   - Each centroid-sentence for each topic category is chosen by the extracted keywords and a similarity measure; each centroid-sentence is regarded as having the core meaning of each topic category, and it is a topic–keyword sentence, which includes a topic word or a keyword, or a requirement sentence which has high similarity to the topic category by the similarity measure technique.
3. *Learning classifier*: the Naive Bayes classifier is learned by using the set of centroid-sentences as training data.

The preprocessing module has two main roles: (i) segmenting requirements into sentences, and (ii) extracting content words. The Sogang Tagger for Korean, Brill's POS tagger for English [25], and a stop-word list are used in order to extract content words. A classification task needs more words (features) than a topic word for each topic category, and the classification task can obtain enough words through the bootstrapping process of the proposed method. Its starting point is topic words provided by analysts and the basic processing unit is a sentence. In step 2, a set of centroid-sentences is composed of requirement sentences that reflect the special features of each topic category. The topic words, keywords, and a similarity measure technique are used for extracting this centroid-sentence. Since a text classifier such as Naive Bayes can be built with the set of centroid-sentences for each topic category, the collected requirements can be classified into analysts' views by using the text classifier finally.

The proposed bootstrapping method will be explained in the following sections.

3.2. Constructing the set of centroid-sentences for training data

The proposed bootstrapping method starts with the fact that most analysts can extract topic words for analysts' views about the software to be developed, and the final goal is to build up the sets of centroid-sentences as training data. The set of centroid-sentences consists of topic–keyword sentences and sentences assigned by the similarity measure technique.

3.2.1. Generating a keyword list for each topic category

Each topic word represents the main meaning to a topic category. Since each topic word cannot have enough information for the appropriate topic category, we first need to extract useful keywords that are semantically related to each topic word. A keyword list consists of words that are highly similar to each topic word. In traditional research, a thesaurus and MRD (Machine-Readable Dictionary) have been used for generating the keyword lists [18,26]. However, the feature of words in requirement sentences is more dependent on an application domain than that of words in general corpus. In that sense, we cannot expect high performance when using the thesaurus and MRD. Also, it is a very difficult and time-consuming task to construct a thesaurus or MRD for each application domain. Therefore, we automatically create a keyword list for each topic category through the following similarity measure technique.

The similarity score between a topic word, T, and a candidate word or keywords, W, is estimated by the cosine metric as follows:

$$sim(T,W) = \frac{\sum_{i=1}^{n} t_i \times w_i}{\sqrt{\sum_{i=1}^{n} t_i^2 \times \sum_{i=1}^{n} w_i^2}} \tag{1}$$

where t_i and w_i represent the occurrence of words, T and W, in i-th requirement, respectively; they are denoted by binary value, 0 or 1, and n is the total number of collected requirements. This formula estimates the similarity score between words according to the degree of their co-occurrence in the same requirement.

The most important criterion for good keywords of a topic category is a similarity with the topic word of each topic category, and it can be measured by formula (1). Then the ambiguity of words must be considered. That is, if any word has a high similarity with topic words of two or more topic categories, the word must not be regarded as a keyword candidate because it does not have the power to discriminate these topic categories. To apply the former criterion to our method, each word is first assigned to the keyword candidate list of a topic category with the maximum similarity score, and the important score of each assigned word is recalculated by using the following formula for latter criterion:

$$score(W, TC_{\max}) = sim(T_{\max}, W) + (sim(T_{\max}, W) - sim(T_{second\_\max}, W)) \tag{2}$$

where T_{max} is a topic word with the maximum similarity score to a word W, TC_{max} is the topic category of the topic word T_{max}, and T_{second_max} is the other topic word with the second highest similarity score to the word W.

This formula means that a highly ranked word in a topic category has a high similarity score to the topic word of the topic category ($sim(T_{max}, W)$) and a high similarity score difference to other topic words ($sim(T_{max}, W)$-$sim(T_{secondmax}, W)$). We sort out candidate words in descending order, and then we choose the top m words as keywords for the topic category. By using these topic words and keywords, we can extract the sentences that sufficiently contain the features of each topic category. They are called topic–keyword sentences.

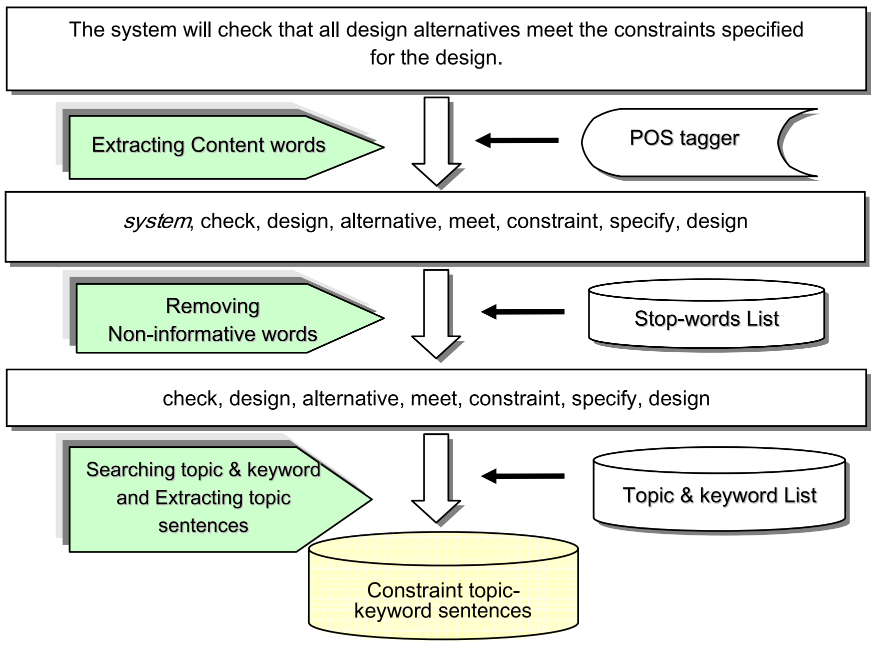

The system will check that all design alternatives meet the constraints specified for the design.
Extracting Content words
POS tagger
system, check, design, alternative, meet, constraint, specify, design
Removing Non-informative words
Stop-words List
check, design, alternative, meet, constraint, specify, design
Searching topic & keyword and Extracting topic sentences
Topic & keyword List
Constraint topic-keyword sentences

Fig. 2. The process of extracting topic–keyword sentences.

3.2.2. Extracting topic–keyword sentences

First, each sentence with a topic word or a keyword is extracted as a topic–keyword sentence from collected requirement sentences. The topic–keyword sentence generally contains many meaningful content words for each topic category. The content words are in the first order co-occurrence of topic words and keywords; they directly occur in the same sentence with a topic word or keywords.

Fig. 2 shows the process of extracting topic–keyword sentences.

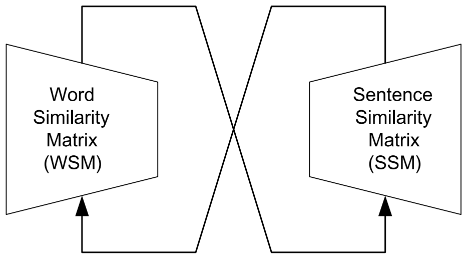

Word Similarity Matrix (WSM)
Sentence Similarity Matrix (SSM)

Fig. 3. Iterative computation of word and sentence similarities.

3.2.3. Measuring word and sentence similarities

Although a set of extracted topic–keyword sentences has many words (features), they do not have sufficient ones for a text classification task. Thus a similarity measure technique is used in order to measure similarities between topic–keyword sentences and unclassified requirement sentences. By using this technique, more words can be obtained for each topic category. Here, each unclassified sentence, which is not assigned as a topic–keywords sentence, is classified into each topic category through measuring similarities between the unclassified sentence and the topic–keyword sentences. As similar words tend to appear in similar contexts, we compute the similarity by using contextual information [20,26,27]. In this paper, words and sentences play complementary roles. That is, a sentence is represented by the set of words it contains, and a word by the set of sentences in which it appears. Sentences are similar to the extent that they contain similar words, and words are similar to the extent that they appear in similar sentences. This definition is circular. Thus, it is applied iteratively using two matrices as shown in Fig. 3.

In Fig. 3, each topic category has a word similarity matrix WSM_n and a sentence similarity matrix SSM_n. In each iteration n, we update WSM_n, whose rows and columns are labeled by all content words that occur in the topic–keyword sentences of each topic category and in unclassified requirement sentences. In that matrix, the cell (i,j) holds a value between 0 and 1, indicating the extent to which the i-th word is contextually similar to the j-th word. Also, we keep and update an SSM_n, which holds similarity values among sentences. Each row of the SSM_n corresponds to an unclassified requirement sentence, and each column corresponds to a topic–keyword sentence.

To compute the similarities, we initialize WSM_n to the identity matrix. That is, each word is fully similar (1) to itself and completely dissimilar (0) to all other words. The following steps are iterated until the changes in the similarity values are small enough.[1]

1. Update the sentence similarity matrix, SSM_n, using the word similarity matrix, WSM_n.

[1] The convergence of this iteration was proved in Karov and Edelman's paper [26].

2. Update the word similarity matrix, WSM_n, using the sentence similarity matrix, SSM_n.

3.2.3.1. Affinity formulae. To simplify the symmetric iterative treatment of similarity between words and sentences, we define an auxiliary relation between words and sentences, which we call affinity. A word W is assumed to have an affinity to every sentence, which is a real number between 0 and 1. The affinity reflects the contextual relationships between W and the words of the sentence. If W belongs to a sentence S, its affinity to S is 1. If W is totally unrelated to S, the affinity is close to 0. If W is contextually similar to the words of S, its affinity to S is between 0 and 1. In a similar manner, a sentence S has some affinity to every word, reflecting the similarity of S to the sentences involving that word.

In these formulae, $W \in S$ means that a word belongs to a sentence. Affinity formulae are defined as follows [26]:

$$aff_n(W,S) = \max \ W_i \in S \ sim_n(W,W_i) \tag{3}$$
$$aff_n(S,W) = \max \ W \in S_j \ sim_n(S,S_j) \tag{4}$$

In the above formulae, n denotes the iteration number, and similarity values are defined by WSM_n and SSM_n. Every word has some affinity to the sentence, and the sentence can be represented by a vector indicating the affinity of each word to it. Note that affinity is asymmetric as follows:

$$aff_n(W,S) \neq aff_n(S,W) \tag{5}$$

3.2.3.2. Similarity formulae. The similarity of W_1 to W_2 is the average affinity of sentences that include W_1 to W_2, and the similarity of a sentence S_1 to S_2 is a weighted average of the affinity of the words in S_1 to S_2. Each similarity formula is defined as follows [27]:

$$sim_{n+1}(S_1,S_2) = \sum_{w \in s_1} weight(W,S_1)\ aff_n(W,S_2) \tag{6}$$

$$\text{if } W_1 = W_2$$
$$sim_{n+1}(W_1,W_2) = 1$$
$$\text{else}$$
$$sim_{n+1}(W_1,W_2) = \sum_{W_1 \in S} weight(S,W_1)\ aff_n(S,W_2) \tag{7}$$

The weights in formula (6) are computed by the methodology in the next section. The weights are the reciprocal numbers of sentences that contain W_1 and the sum of these weights is 1. These values are used to update the corresponding entries of WSM_n and SSM_n.

3.2.3.3. The intuitive explanation of the similarity measure algorithm through an example. Here, we try to describe the above similarity measure algorithm more intuitively through an example shown in [26]. In the initial iteration, only identical words have a similarity value, so that $aff(W, S) = 1$ if $W \in S$ and the affinity is otherwise zero. Then, in the first iteration, the similarity between S_1 and S_2 depends on the number of words that appeared in S_1 and S_2 simultaneously. In the next iteration, each word $W \in S_1$ contributes to the similarity of S_1 to S_2 by a value between 0 and 1, indicating its affinity to S_2, instead of voting either 1 (if $W \in S_2$) or 0 (if $W \notin S_2$). Through the similar process, sentences also contribute values to word similarity.

This similarity measure algorithm can catch the higher-order contextual relationships between sentences. This point makes a contribution to obtain higher performance than other similarity methods. The following simple example shows the process of capturing the higher-order contextual relationships.

The set of simple three sentences
S$_1$: "*eat banana*", S$_2$: "*taste banana*", S$_3$: "*eat apple*"

Iterative Process
Initialization: Every word is similar to itself only.

First iteration:

- *The sentence similarity*: the sentences "*eat banana*" and "*eat apple*" have similarity of 0.5 because of the common word '*eat*'. By the similar reason, the sentences "*eat banana*" and "*taste banana*" have similarity 0.5.
- *The word similarity*: '*banana*' is learned to be similar to '*apple*' because of their common usage with '*eat*'. '*taste*' is similar to '*eat*' because of their common usage with '*banana*'. But '*taste*' and '*apple*' are not similar yet.

Second iteration:

- *The sentence similarity*: the sentence "*taste banana*" has now some similarity to "*eat apple*" because, in the previous iteration, '*taste*' was similar to '*eat*' and '*banana*' was similar to '*apple*'.
- *The word similarity*: the word '*taste*' is now similar to '*apple*' because the '*taste*' sentence ("*taste banana*") is similar to the '*apple*' sentence ("*eat apple*"). Yet, '*banana*' is more similar to '*apple*' than '*taste*', because the similarity value of '*banana*' and '*apple*' further increases in the second iteration.

3.2.3.4. Word weights. In formula (6), the weight of a word is a product of three factors. It is used to exclude the words that are expected to be given unreliable similarity values. The weights do not change in their process of iteration.

1. *Global Frequency*: The frequent words in total requirements are less informative of sentence similarity. For example, a word like 'software' frequently appears in total requirements. The formula is as follows [26]:

$$max\left\{0, 1 - \frac{freq(W)}{\max 5_x freq(x)}\right\} \tag{8}$$

In formula (8), $\max 5_x freq(x)$ is the sum of the five highest frequencies in total requirements.

2. *Log-likelihood factor*: In general, the words, which are indicative of the sense, usually appear in topic–keyword sentences more frequently than in total requirements. The log-likelihood factor captures this tendency. It is computed as follows [26]:

$$\log \frac{\Pr(W_i|W)}{\Pr(W_i)} \tag{9}$$

In formula (9), $Pr(W_i)$ is estimated from the frequency of W_i in the total requirements, and $Pr(W_i|W)$ from the frequency of W_i in topic–keyword sentences. To avoid poor estimation for words with a low count in topic–keyword sentences, we multiply the log-likelihood by formula (10) where $count(W_i)$ is the number of occurrences of W_i in topic–keyword sentences. For the words which do not appear in topic–keyword sentences, we assign the weight (1.0) to them. The other words are assigned the weights that add 1.0 to the computed values.

$$min\left\{1, \frac{count(W_i)}{3}\right\} \tag{10}$$

3. *Part of speech*: Each part of speech is assigned a weight. We assign the weight (1.0) to 'proper noun' and 'common noun,' and we assign the weight (0.6) to 'verb'.

The total weight of a word is the product of the above factors, and each weight is normalized by the sum of factors of words in a sentence as follows [26].

$$weight = \frac{factor(W_i, S)}{\sum_{W_i \in S} factor(W_i, S)} \tag{11}$$

In formula (11), $factor(W_i, S)$ is the weight before normalization.

3.3. Constructing the Naive Bayes classifier from centroid-sentence sets

In the above section, we obtained centroid-sentences of each topic category for training data. Using these centroid-sentences, we can build up a Naive Bayes classifier. Finally, the classifier can automatically classify collected requirements into each topic category.

The Naive Bayes classifier is a probabilistic generative model for the data, and embodies three assumptions about the generative process: (i) the data are produced by a mixture model, (ii) there is a one-to-one correspondence between mixture components and categories, and (iii) the mixture components are multinomial distributions of individual words; the words of a requirement are produced independently of each given the topic category [28]. From these assumptions, the Naive Bayes classifier can be derived by finding the most probable parameters for the model.

Requirements are generated by a mixture of multinomial models, where each mixture component corresponds to a topic category. Formally, every requirement is generated according to a probability distribution defined by the parameters for the mixture model, denoted θ. Let there be $|C|$ topic categories and a vocabulary of size $|V|$; each requirement r has $|r|$ words in it. The probability distribution consists of a mixture of components $tc_j \in TC = \{tc_1, \ldots tc_{|TC|}\}$. Each component is parameterized by a disjoint subset of θ. A requirement, r_i, is created by first selecting a mixture component according to the mixture weights, $P(tc_j|\theta)$, then having this selected mixture component generate a requirement according to its own parameters, with distribution $P(r_i|tc_j;\theta)$. Thus, we can characterize the likelihood of requirement r_i with a sum of total probability over all mixture components [28]:

$$P(r_i|\theta) = \sum_{j=1}^{|TC|} P(tc_j|\theta) P(r_i|tc_j;\theta) \tag{12}$$

We here assume that there is a one-to-one correspondence between mixture model components and topic categories, and thus use tc_j to indicate the j-th mixture component as well as the j-th topic category. The parameters of an individual mixture component define a multinomial distribution over words; the collection of word probabilities, each written $\theta_{w_t|tc_j}$, such that $\theta_{w_t|tc_j} \equiv P(w_t|tc_j;\theta)$, where w_t denotes t-th word, $t = \{1, \ldots, |V|\}$ and $\sum_t P(w_t|tc_j;\theta) = 1$. Note that, in general, we assume that requirement length is independent of topic category and the words of a requirement are generated independently of context; the standard Naive Bayes assumption. We further assume that the probability of a word is independent of its position within the requirement. Thus the remaining parameters of the model are topic category probabilities, written θ_{tc_j}. Thus the final parameters of the model consist of multimodal and topic category probabilities: $\theta = \{\theta_{w_t|tc_j} : w_t \in V, tc_j \in TC; \theta_{tc_j} : tc_j \in TC\}$.

The parameter estimation formulae that result from maximization with the data and our prior are the familiar ratios of empirical counts. The estimated probability of a word given a topic category, $\hat{\theta}_{w_t|tc_j}$, is simply the number of times word w_t occurs in the training data for topic category tc_j, divided by the total number of word occurrences in the training data for that topic category. Here, the Laplace smoothing technique is used for the estimation of a word given a topic category, $\hat{\theta}_{w_t|tc_j}$. Smoothing is necessary to prevent zero probabilities for infrequently occurring words.

The word probability estimates are as follows:

$$\hat{\theta}_{w_t|tc_j} \equiv P(w_t|tc_j;\hat{\theta}) = \frac{1 + N(w_t, tc_j)}{|V| + \sum_{t=1}^{|V|} N(w_t, tc_j)} \tag{13}$$

where $N(w_t,tc_j)$ is the count of the number of times word w_t occurs in category tc_j. The topic category probabilities, $\hat{\theta}_{tc_j}$, are estimated in the same manner, and also use the smoothing technique:

$$\hat{\theta}_{tc_j} \equiv P(tc_j|\hat{\theta}) = \frac{1+|tc_j|}{|TC|+|R|} \tag{14}$$

where $|tc_j|$ is the number of requirements in topic category ts_j, $|R|$ is the number of all requirements in training data, and $|TC|$ is the number of topic categories.

Using formulae (13) and (14), the parameters are estimated from the training data. It is then possible to turn to the generative model and calculate the probability that a particular mixture component generated a given requirement. We use the Naive Bayes classifier with minor modifications based on Kullback-Leibler Divergence [29–31]. A requirement r_i is classified into a topic category according to the following formula:

$$\begin{aligned} P(tc_j|r_i;\hat{\theta}) &= \frac{P(tc_j|\hat{\theta})P(r_i|tc_j;\hat{\theta})}{P(r_i|\hat{\theta})} \\ &\approx P(tc_j|\hat{\theta})\prod_{t=1}^{|V|}P(w_t|tc_j;\hat{\theta})^{N(w_t,r_i)} \\ &\propto \frac{\log(P(tc_j;\hat{\theta}))}{n} + \sum_{t=1}^{|V|}P(w_t|r_i;\hat{\theta}) \\ &\quad \times \log\frac{P(w_t|tc_j;\hat{\theta})}{P(w_t|r_i;\hat{\theta})} \end{aligned} \tag{15}$$

where n is the number of words in requirement r_i, w_t is the t-th word in the vocabulary, $N(w_t,r_i)$ is the frequency of word w_t in requirement r_i, and tc_j is a topic category.

4. Experimental evaluation

4.1. Experimental data and evaluation measures

To evaluate the effectiveness of the proposed requirements classification method, we collected requirements from real software development fields for Korean and English. The English requirements data set consists of 196 requirements sentences, and the Korean requirements data set consists of 193 requirements sentences. Next, the English data set and the Korean data set consist of four topic categories and six topic categories, respectively. Tables 1 and 2 show topic words and keywords used in each test data set.

These topic words are input by system analysts, and keywords are automatically created by our keyword extraction method. In addition, the proposed system can supply useful information to system analysts which can then be used to more easily choose topic words. Analysts can check out candidate words of each topic word, which are sorted according to the information score calculated, by using the *tfidf* scheme. The *tfidf* scheme has been generally applied to estimating importance weights for words in the Information Retrieval literature [32]. In Tables 1 and 2, numbers in italic parenthesis denote the ranking of topic words in sorted lists, and they show that the topic words in our experiments are generally located in high ranks.

For evaluation measures, we use the standard definition of precision, recall, and F_1 measure to evaluate experimental results. Precision is the probability that a requirement predicted to be in a topic category truly belongs to that topic category. Recall is the probability that a requirement belonging to a topic is classified into this topic category. The F_1 measure combines recall and precision with an equal weight. Formulae for these evaluation measures are as follows [22,33,34]:

$$precision = \frac{topic\ requirements\ found\ and\ correct}{total\ topic\ requirements\ found} \times 100 \tag{16}$$

$$recall = \frac{topic\ requirements\ found\ and\ correct}{total\ topic\ requirements\ correct} \times 100 \tag{17}$$

$$F_1 measure = \frac{2 \times precision \times recall}{precision + recall} \tag{18}$$

Table 1
Topic words and keywords lists used in the English requirements data set

Topic word	Keywords
Simulink (*9*)	model, port, ford, development, tool
Characterization (*14*)	vector, level, attribute, domain, member
Architecture (*1*)	description, template, connector, modification, relationship
Constraint (*5*)	design, log, alternative, assembly, choice

4.2. Primary experimental results

4.2.1. The observation of performance changes according to the number of keywords

First of all, the number of keywords must be determined for the bootstrapping method. The number of keywords in

Table 2
Topic words and keywords lists used in the Korean requirements data set

Topic word	Keywords
Personnel (*17*) (in-sa)	announcement of appointment (bal-ryoung), ledger (won-jang), transference (jeon-bo), register of the staff (myuong-bu)
Employment (*36*) (chae-yong)	selection (jeon-hyoung), mark (jeom-su), examination (si-heum), application for examination (eung-si)
Attendance (*4*) (geun-mu)	hour (si-gan), time (ta-im), the state of work (sang-hwang-bu), workday (pwoung-il)
Appraisal (*2*) (pyeong-ga)	evaluator (pyoung-ga-ja), subdivision (sea-bun), difference (pyen-cha), objectivity (geak-kwan)
Salary (*5*) (geup-yeo)	the accountant's section (kyoung-li-bu), payment (ji-geup), basic salary (ki-bon-geup), cash transaction (i-chea)
Vacation (*1*) (hew-ga)	fixed term (jeong-ki), sick leave (byoung-ka), service (yong-yeok), rest (jan-yeo)

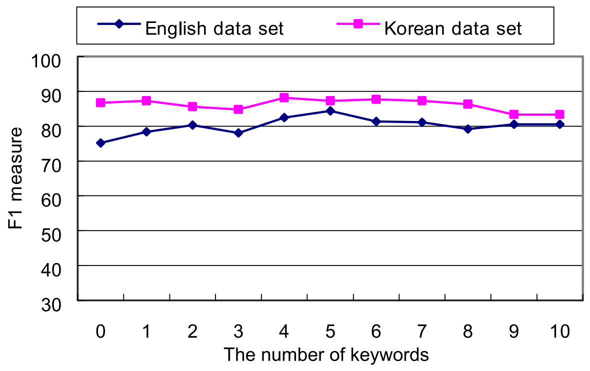


Fig. 4. The performance changes according to the number of keywords.

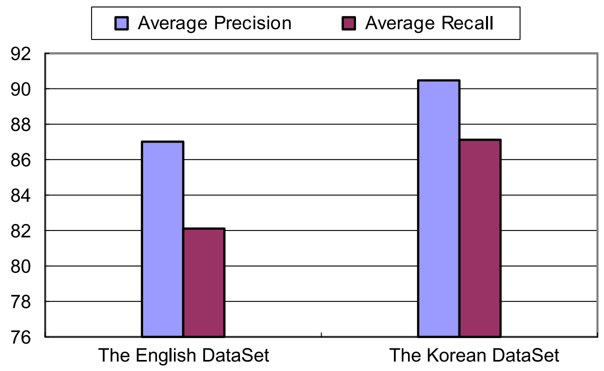


Fig. 5. The performance differences in each data set.

our experiment is limited by the top *n*-th keyword from the ordered list of each topic category. Fig. 4 displays the performance at different numbers of keywords (from 0 to 10) in both English and Korean requirements data sets. Using zero keyword means that only the topic word is used.

As shown in Fig. 4, we obtained the best performance at 5 keywords in the English data set and 4 keywords in the Korean data set. Therefore, we set the number of keywords to 5 for the English data set and 4 for the Korean data set in the following experiments.

4.2.2. *The performance of English and Korean requirement data sets*

Tables 3 and 4 report performance results for each topic category from each data set. As shown in the following Tables, we obtained an 84.28% F_1 score in the English data set and an 87.91% F_1 score in the Korean data set. Fig. 5 shows differences between average precision and average recall in each requirement data set.

Here, we can find the performance difference between these data sets. In fact, the topic categories of the English data set are somewhat more confusable than the Korean data set, and the content words of each topic categories frequently occur in other topic categories. We think that these factors contribute to the comparatively poor performance of the English data set. This indistinctness problem is not caused by language characteristics. As shown in Tables 1 and 2, the English data set and the Korean data set are built up from different domains, and their topic categories show the different aspects in each domain. Because the English data set has more ambiguous topic categories than the Korean data set, it showed a lower performance than the Korean data set.

As shown in Fig. 5, the average precision scores are superior to the average recall scores in both data sets. The proposed system can be more useful for requirements analysts when it is applied to any software development domain of which analysts do not know the distribution and tendency of requirements well. If requirements analysts cannot understand the software development domain, they cannot easily remove requirements which are assigned into fault topic category. From this point of view, a system with high precision can be more effective in that system analysts can observe the classified requirements more accurately. Note that the proposed system also achieved good performances in recall performance measure.

5. Discussions

5.1. *The time savings*

Here, we try to show the difference in human efforts between a manual classification task and the proposed one. The processing times of the manual approach and the proposed approach can be calculated as follows:

Table 3
The performance score of each topic category for English data set

	Simulink	Characterization	Architecture	Constraint	Average
Precision	89.47	85.71	92.85	80.00	***87.01***
Recall	94.44	70.58	76.47	86.95	***82.11***
F_1 measure	**91.89**	**77.41**	**83.87**	**83.95**	***84.28***

Table 4
The performance score of each topic category for Korean data set

	Personnel	Employment	Attendance	Appraisal	Salary	Vacation	Average
Precision	79.31	100	66.66	100	96.87	100	***90.47***
Recall	92.00	100	72.72	61.11	96.87	100	***87.12***
F_1 measure	**85.18**	**100**	**69.56**	**75.86**	**96.87**	**100**	***87.91***

$$\begin{aligned} \mathit{Manual\ Approach} = {} & (\#\ \mathit{of\ Requirements}) \\ & \times (\mathit{Decision\ Time}) \\ & \times (\mathit{Times\ of\ Changed\ Topics} + 1) \\ & + (\#\mathit{of\ Changed\ Requirments}) \\ & \times (\mathit{Decision\ Time}) \end{aligned} \quad (19)$$

$$\begin{aligned} \mathit{Proposed\ Approach} = {} & (\#\mathit{of\ Existing\ Topics} \\ & + \#\ \mathit{of\ Changed\ Topics}) \\ & \times (\mathit{Topic\ Words\ Creating\ Time}) \end{aligned} \quad (20)$$

where (i) '*Decision Time*' denotes the time required in order to classify one requirement sentence, and (ii) '*Topic Words Creating Time*' describes the time required in order to create a new topic word; we suppose that both of them take 5 s.

As we assume two cases of the best case and the worst case which occur in real application area, we can estimate the processing time of a manual approach and the proposed approach, respectively. The best case for the manual approach does not change any requirement and any topics, while the worst case for that changes 50% requirements and topics once; we suppose some bad situation, not the worst case, because it is very hard to assume the worst case. The best case for the proposed approach does not change topics, while the worst case changes all the topics.

In our experiments, the Korean data set consists of 6 topic categories with 193 requirement sentences and the English data set consists of 4 topic categories with 196 requirement sentences. We estimate the processing time of both the manual and proposed approaches on our two different data sets by using formulae (19) and (20) as follows: (i) in the worst case, $(193 \times 2 \times 5 + 97 \times 5) = 2415$ s for Korean data set and $(196 \times 2 \times 5 + 98 \times 5) = 2450$ s for English data set. (ii) in the best case, $193 \times 5 = 965$ s for the Korean data set and $196 \times 5 = 980$ s. In the other hand, the processing time of the proposed approach is estimated as follows: (i) in the worst case, $(6 + 6) \times 5 = 60$ s for Korean data set and $(4 + 4) \times 5 = 40$ s for English data set, (ii) in the best case, $6 \times 5 = 30$ s for Korean data set and $4 \times 5 = 20$ s for English data set.

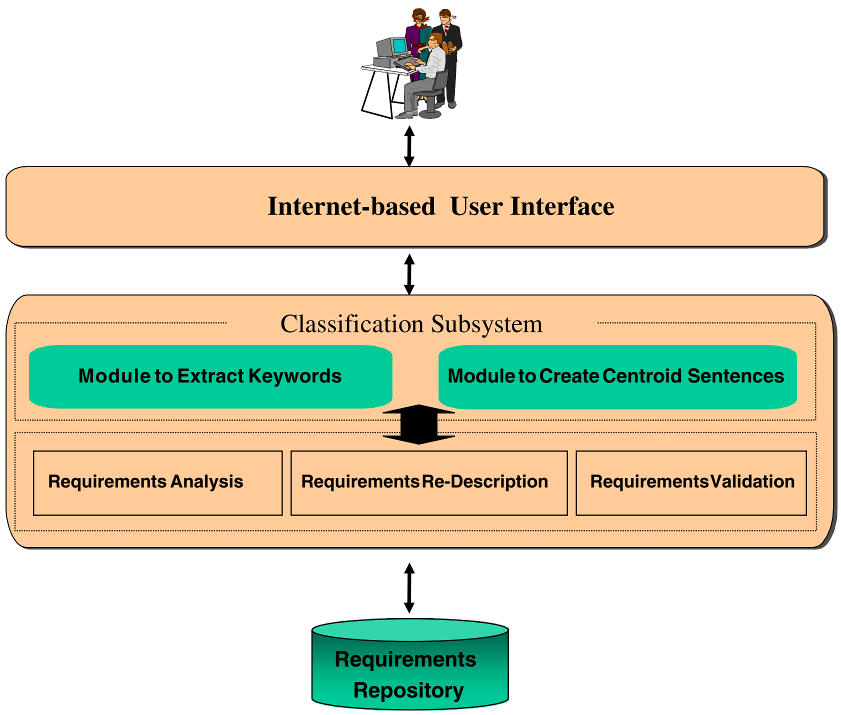
The estimated processing times are summarized in Table 5. As a result, the proposed approach requires only 2–3% processing time in comparison with manual approaches.

Table 5
The estimated results of processing time in both data sets

	The Manual Approach		The Proposed Approach	
	The Worst Case	The Best Case	The Worst Case	The Best Case
The Korean data set	2415 s	965 s	60 s	30 s
The English data set	2450 s	980 s	30 s	20 s

5.2. The internet-based requirements analysis-supporting system

We implemented the Internet-based requirements analysis-supporting system on Linux using C and JAVA. Fig. 6 shows the overall concept of the proposed system.



Internet-based User Interface
Classification Subsystem
Module to Extract Keywords
Module to Create Centroid Sentences
Requirements Analysis
Requirements Re-Description
Requirements Validation
Requirements Repository

Fig. 6. The overall concept of the proposed Internet-based requirements analysis-supporting system.

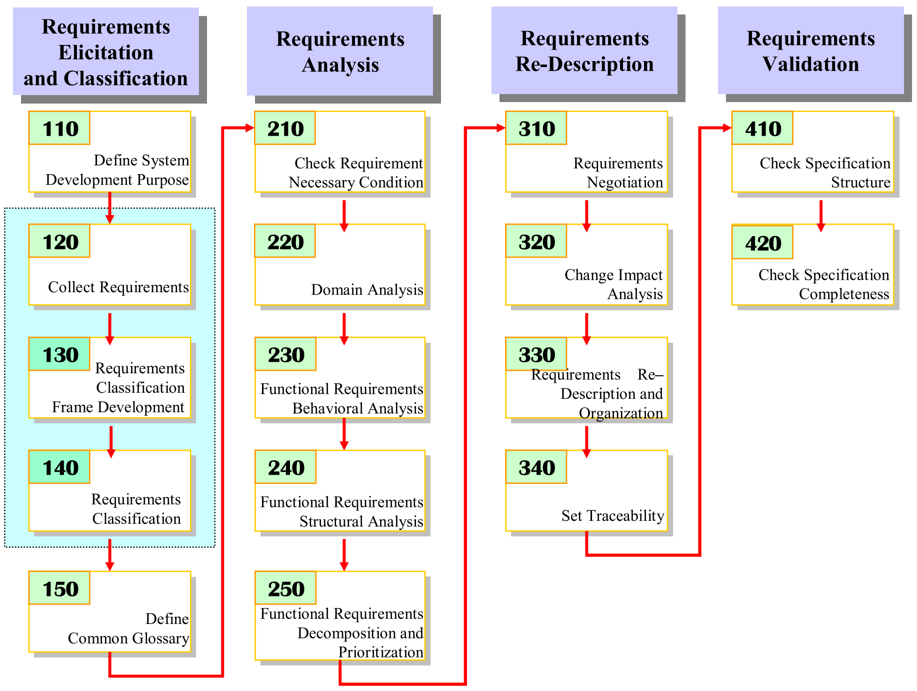

Requirements Elicitation and Classification
Requirements Analysis
Requirements Re-Description
Requirements Validation
110 Define System Development Purpose
120 Collect Requirements
130 Requirements Classification Frame Development
140 Requirements Classification
150 Define Common Glossary
210 Check Requirement Necessary Condition
220 Domain Analysis
230 Functional Requirements Behavioral Analysis
240 Functional Requirements Structural Analysis
250 Functional Requirements Decomposition and Prioritization
310 Requirements Negotiation
320 Change Impact Analysis
330 Requirements Re–Description and Organization
340 Set Traceability
410 Check Specification Structure
420 Check Specification Completeness

Fig. 7. Requirements engineering process.

The requirements engineering process for requirements analysis is displayed in Fig. 7. As shown in Fig. 7, 'process 120', 'process 130', and 'process 140' of the Requirements Elicitation and Classification stage are supported by the proposed system.

For 'process 120', main functions to store and analyze requirements are executed in the server. The users of the proposed system consist of a project manager, system engineers, and general users. They can log into the system and execute functions based on their pre-defined restrictions. The project manager can generate a new project, and he can register the users of the project with restrictions. The functions which are provided to system engineers focus on analyzing requirements, while the user's functions focus on elicitation of requirements.

For 'process 130', a system engineer selects 'Input Topic' in menu lists, and then must determine topic categories by inputting topic words for requirements classification tasks. Based on 'process 130', the requirements classification, 'process 140', is processed. Therefore, the requirements analysis tasks can be made effective and easy by using the requirements classified from our system instead of initial informal requirements.

5.3. The lesson learned in practical application

This paper has shown how to automatically organize requirements in initial requirements analysis. In practical applications, we need a pre-processing step to convert input informal requirements into formal requirements. A preprocessor is used for formatting, elimination of pronouns, and resolving how conjunctions are used [35]. Formatting is mainly used to tag each requirement with a requirement ID so that each requirements statement can be identified by the ID. The elimination is done to substitute pronouns with proper nouns. The problem of conjunctions is to resolve the ambiguities of 'and' or 'or' and separate compound sentences into simple sentences. These actions are performed by a predefined algorithm using Natural Language Processing (NLP) techniques [35]. We recognized that NLP techniques cannot be a complete solution for informal requirements. However, the advantage of using NLP techniques is that the large number of requirements can be processed automatically. Fig. 8 shows the process of a pre-processing step.

In addition, by using the proposed system, a system engineer can obtain an initial automatic rough classification result that he/she can consider as being important concepts, objects or functions. That is, the proposed system can support initial analysis with different viewpoints. This is another advantage of using the proposed system, as the system engineer can classify collected requirements with different concepts in multiple times at low cost.

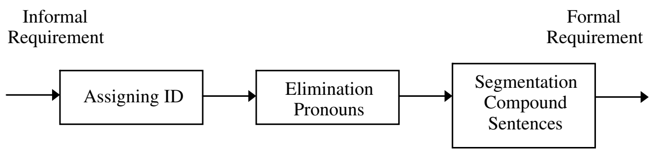


Fig. 8. The process of a pre-processing step.

6. Conclusions and future work

In this paper, we proposed an automated requirements classification technique for early requirements analysis steps. The proposed technique was applied to the Internet-based requirements analysis-supporting system which produces a basis to efficiently analyze the requirements collected from distributed environment. The developed system reduces the amount of work done by hand at the initial stage of requirements analysis, because it automatically classifies the collected requirements into several views. As a result, the proposed system enables rapid and correct requirements analysis from messy requirements. To evaluate the effectiveness of the proposed technique, we performed experiments using two real requirements data sets: the Korean data set and the English data set. We achieved the 84.28% average F_1 score for the English data set and the 87.91% average F_1 score for the Korean data set. Generally, it is difficult to detect the structure of requirements. However, system engineers, using the proposed system, can easily detect the structure of the collected requirements and analyze them by inputting only topic words.

As future research, we will pursue the study of other methods to easily extract the structure of development domain by reusing previously developed structures. We also plan to develop more detailed requirements process based on our classification technique and find other activities where our technique can be applied. Moreover, we need to apply and test the proposed method to more various application areas in order to verify its utility.

Acknowledgement

This research was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

References

[1] D.E. Damian, D. Zowghi, The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization, in: Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02), 2002, pp. 319–328.

[2] I. Sommerville, Integrated requirements engineering: A tutorial, IEEE Software 22 (1) (2005) 16–23.

[3] K. Wiegers, Software requirements, second ed., Microsoft Press, 2003.

[4] D. Leffingwell, D. Widrig, Managing Software Requirements, second ed., Addison-Wesley, 2005.

[5] T. Leonard, V. Berzins, M.J. Holden, Gathering Requirements from Remote Users, in: Proceedings of 9th IEEE International Conference on Tools with Artificial Intelligence, November 1997, pp. 462–471.

[6] B. Berenbach, Impact of Organizational Structure on Distributed Requirements Engineering Processes: Lessons Learned, in: Proceedings of the 2006 international workshop on Global software development for the practitioner (GSD '06), 2006, pp. 15–19.

[7] J.L. Cybulski, K. Reed, Computer-Assisted Analysis and Refinement of Informal Software Requirements Documents, in: Proceedings of the fifth Asia-Pacific Software Engineering Conference (APSEC' 98), Taipei, Taiwan, December, 1998, pp. 128–135.

[8] A.M. Davis, Predictions and farewells, IEEE Software 15 (4) (1998) 6–9.

[9] B. Boehm, M. Abi-Antoun, D. Port, J. Kwan, A. Lunch, Requirements Engineering, Expectations Management and the Two Cultures, in: Proceedings of 4th International Symposium on Requirement Engineering, 1999, pp. 14–22.

[10] B. Boehm, A. Egyed, WinWin Negotiation Process: A Multi-Project Analysis, in: Proceedings of the 5th International Conference on Software Process, 1998, pp. 125–136.

[11] M.H. Klein, R. Kazman, L. Bass, J. Carriere, M. Barbacci, H. Lipson, Attribute-Based Architectural Style, in: Proceedings of the First Working IFIP Conference on Software Architecture(WICSA1), San Antonio, TX, 1999, pp. 225–243.

[12] I. Sommerville, P. Sawyer, Viewpoints: principles, problems and a practical approach to requirements engineering, Annals of Software Engineering 3 (1997) 101–130.

[13] IEEE Std IEEE-Std-1233-1998, IEEE Guide for Developing System Requirements Specifications, IEEE Computer Society Press, 1998.

[14] J.A. Goguen, C. Linde, Techniques for Requirements Elicitation, in: Proceedings of the International Symposium on Requirements Engineering, 1993, pp. 152–164.

[15] S. Park, H. Peter In, S. Choi, Automated Support to Quality Requirements Classification for Supporting WinWin, in: The 2nd International Conference on Computer and Information Science, 2002, pp. 43–58.

[16] K. Ryan, The Role of Natural Language in Requirements Engineering, in: Proceedings of the first IEEE international symposium on requirements engineering (RE'93), San Diego, USA, 1993.

[17] J. Natt och Dag, B. Regnell, V. Gervasi, S. Brinkkemper, A linguistic-engineering approach to large-scale requirements management, IEEE Software 22 (1) (2005) 32–39.

[18] J. Palmer, Y. Liang, Indexing and clustering of software requirements specifications, Information and decision Technologies 18 (1992) 283–299.

[19] J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson, J. Karlsson, A feasibility study of automated natural language requirements analysis in market-driven development, Requirements Engineering 7 (2002) 20–33.

[20] S. Park, H. Kim, Y. Ko, J. Seo, Implementation of an efficient requirements analysis supporting system using similarity measure techniques, Information and Software Technology 42 (2000) 429–438.

[21] A. Fantechi, S. Gnesi, G. Lami, A. Maccari, Applications of linguistic techniques for use case analysis, Requirements Engineering 8 (2003) 161–170.

[22] H. Chen, P. Hsu, R. Orwig, L. Hoopes, J.F. Nunamaker, Automatic concept classification of text from electronic meetings, Communications of the ACM 37 (10) (1994) 56–73.

[23] M. Mitra, C. Buckley, A. Singhal, C. Cardie, An Analysis of Statistical and Syntactic Phrases, in: Proceedings of the fifth International Conference on Computer-Assisted Information Searching on the Internet (RIAO'97), Montreal, Canada, June, 1997.

[24] Y. Ko, S. Park, J. Seo, Web-based Requirements Elicitation Supporting System Using Requirements Categorization, in: Proceedings of Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE 2000), Chicago, USA, 2000, pp. 344–351.

[25] E. Brill, Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging, Computational Linguistics 21 (4) (1995).

[26] Y. Karov, S. Edelman, Similarity-based word sense disambiguation, Computational Linguistics 24 (1) (1998) 41–60.

[27] Y. Ko, J. Seo, Learning with Unlabeled Data for Text Categorization Using Bootstrapping and Feature Projection Techniques, in: Proceedings of the 42ndAnnual Meeting of the Association for Computational Linguistics (ACL 2004), July 2004, pp. 255–262.

[28] K.P. Nigam, Using Unlabeled Data to Improve Text Classification, Doctoral dissertation, Computer Science Department, Carnegie Mellon University, 2001.

[29] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery, Learning to construct knowledge bases from the world wide web, Artificial Intelligence 118 (1–2) (2000) 69–113.

[30] Y. Ko, J. Seo, Using the feature projection technique based on a normalized voting method for text classification, Information Processing & Management 40 (2) (2004) 191–208.

[31] Y. Ko, J. Park, J. Seo, Improving text categorization using the importance of sentences, Information Processing & Management 40 (1) (2004) 65–79.

[32] C.J. RujsbergenInformation Retrieval, 2, Butterworths, Stoneham, MA, 1979.

[33] Y. Yang, A evaluation of statistical approaches to text categorization, Information Retrieval Journal 1 (1/2) (1999) 67–88.

[34] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in: Proceedings of European Conference on Machine Learning (ECML 1998), Springer, 1998, pp. 137–142.

[35] S. Park, J. Palmer, Automated Support to System Modeling from Informal Software Requirements, in: Proceedings of the 6th International conference on Software Engineering and knowledge Engineering, June 1994.