1-s2.0-S1568494616303830-hasrina asoc.pdf

Zarinah Mohd. Kasirun, Assoc. Prof.

Available at: https://works.bepress.com/zarinah_mohdkasirun/7/
Extracting features from online software reviews to aid requirements reuse

Noor Hasrina Bakar\textsuperscript{a}, Zarina M. Kasirun\textsuperscript{a,}\textsuperscript{*}, Norsaremah Salleh\textsuperscript{b}, Hamid A. Jalab\textsuperscript{a}

\textsuperscript{a} Department of Software Engineering, Faculty of Computer Science & IT, University of Malaya, 50603 Kuala Lumpur, Malaysia  
\textsuperscript{b} Department of Computer Science, Kulliyyah of ICT, International Islamic University Malaysia, P.O. Box 10, 50728, Kuala Lumpur, Malaysia

\textbf{A R T I C L E   I N F O}

Article history:
Received 31 December 2015  
Received in revised form 24 June 2016  
Accepted 25 July 2016  
Available online 2 August 2016

Keywords:
Requirements reuse  
Software engineering  
Natural language processing  
Unsupervised learning  
Latent semantic analysis

\textbf{A B S T R A C T}

Sets of common features are essential assets to be reused in fulfilling specific needs in software product line methodology. In Requirements Reuse (RR), the extraction of software features from Software Requirement Specifications (SRS) is viable only to practitioners who have access to these software artefacts. Due to organisational privacy, SRS are always kept confidential and not easily available to the public. As alternatives, researchers opted to use the publicly available software descriptions such as product brochures and online software descriptions to identify potential software features to initiate the RR process. The aim of this paper is to propose a semi-automated approach, known as Feature Extraction for Reuse of Natural Language requirements (FENL), to extract phrases that can represent software features from software reviews in the absence of SRS as a way to initiate the RR process. FENL is composed of four stages, which depend on keyword occurrences from several combinations of nouns, verbs, and/or adjectives. In the experiment conducted, phrases that could reflect software features, which reside within online software reviews were extracted by utilising the techniques from information retrieval (IR) area. As a way to demonstrate the feature groupings phase, a semi-automated approach to group the extracted features were then conducted with the assistance of a modified word overlap algorithm. As for the evaluation, the proposed extraction approach is evaluated through experiments against the truth data set created manually. The performance results obtained from the feature extraction phase indicates that the proposed approach performed comparably with related works in terms of recall, precision, and F-Measure.

\textcopyright 2016 Elsevier B.V. All rights reserved.

1. Introduction

Requirements for any existing system can be extracted and reused for production of a new similar system [8]. However, reuse of software features extracted from Software Requirement Specifications (SRS) is only viable to practitioners who have access to these software artefacts. SRS usually reside within company databases that are kept confidential and therefore makes it hard for external researchers to access and further explore its reuse potential. In related research, software features were extracted from various other forms of Natural Language Requirements (high-level requirements) when SRS are not easily accessible, for example product brochures were used by Ferrari et al. [11], online product listings were used by Davril et al. [9], and the use of multiple web repositories were reported by Yu et al. [29]. Compared to related works, this research is focusing on extracting software features that can represent the functionality of software being reviewed. Few works in the requirements engineering (RE) area used the mobile app reviews to either extract the new feature requests or for the purpose of redesigning the existing functionalities for the mobile apps. However, to the best of current knowledge, none of the works has reported the use of online software reviews purposely to initiate the Requirements Reuse (RR) process.

Various information can be obtained from software reviews that are available on the Internet. These can include user opinions or sentiments towards certain products, user complaints, new feature requests, and also statements about existing software functionalities. Software reviews are accessible artefacts that not only contain beneficial information for new users before making software purchasing decision, but these reviews also contain important information for developers, in which it encompasses the evaluations made by customers including ideas for improvements, new feature requests, and existing functionalities of software being reviewed. Information obtained from the software reviews can be
Table 1: Research problems and proposed solutions.

<table>
<thead>
<tr>
<th>#</th>
<th>Problems</th>
<th>The Proposed Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manual and ad hoc reuse of requirements can be very arduous, time-consuming, and labour-intensive on the results.</td>
<td>Propose a less complex semi-automated process to extract software features that can reduce manual extraction effort.</td>
</tr>
<tr>
<td>2</td>
<td>Existing approaches provide little guidelines for practitioners’ adoptions.</td>
<td>Perform experiment to demonstrate the proposed feature extraction process.</td>
</tr>
<tr>
<td>3</td>
<td>Very limited works in this area that provide empirical evaluations towards their results.</td>
<td>Provide results comparison and empirical evaluations towards the proposed approach as compared to manual approach.</td>
</tr>
</tbody>
</table>

Valuable for developers who have no access to the technical documentation of the software product such as SRS, but have the intention to reuse the information (RR process) for the development of similar products. In manual and ad hoc reuse, domain analysts need to firstly read the whole requirement documents to get an overview on what the product is about. Second, to extract related features, domain analysts need to manually select sentences or phrases that indicate the software feature, in which can be accurate but require more processing time and error-prone when dealing with large data.

Mainly, this work contributes to the existing body of knowledge within the feature extraction from NL requirements, which employ the NL processing and data mining techniques in Software Engineering area. Secondly, the application of Soft Computing techniques from the Support Vector Machine area is applied in this real world Software Engineering problem of requirement reuse. The application of Latent Semantic Analysis, LSA (within the scope of Support Vector Machine area) was demonstrated in this paper as a way to identify similar requirements documents, in which researchers interested in exploring natural language artefacts can benefit from the implementation steps outlined in this paper. In addition to LSA, this paper present the use of clustering algorithm such as K-Means, Fuzzy C-Means and Self Organizing Map towards data obtained from LSA. This paper presents the possibilities of clustering algorithm application in software engineering area. Furthermore the quality of documents clustering results produced are then evaluated by measuring its silhouette values, and comparisons between silhouette analysis and clustering accuracy are thoroughly presented. Thirdly, a part from applying the soft computing techniques, this research provides a dataset from software reviews extracted from the internet, which can be later used by other researchers to further explore on improving the feature extraction approach, or for suggesting a new approach for requirements reuse. As to date this paper is written, to the best knowledge of the authors, there is no dataset that is made freely available for requirements reuse research that are based on software reviews.

1.1 Problem statement

Inspired by the related works, this research is aiming to explore the semi-automated process to extract software features from public data, which can assist the process to detect reusable software features from natural language (NL) requirements. The definition of features is referred to as a prominent or distinctive user visible characteristic of a software product [21]. Previous works in the RR area within the Software Product Line Engineering (SPLE) community such as Ferrari et al. [11] and Hariri et al. [15] have looked at the publicly available data as an input to the RR process, but as to date this paper is written, and to the best of the author’s knowledge, none of the related works has reported the use of online software reviews to initiate the RR process. Table 1 lists out the research problems and the proposed solutions that will be further explored in this paper:

Two issues that motivate this research are algorithm complexity and missing guidelines for future practitioners’ adoption. The purpose of this paper is to present a less complex approach for automated feature extraction for RR and to provide insights for future practitioners’ adoption.

In this work, a semi-automated process is presented to extract the phrases from software reviews. Collection of these phrases typically functions as “action” and “object”. To represent a system function, it is believed that a requirement statement must at least encompasses of an “action” (usually represented by verbs), followed by an “object” (represented by verbs, nouns, and adjectives). The representation of “actions” and “objects” that exist in user reviews is believed to expose the software functionalities. This is important so that it can be reused in the production of similar system in the near future. The NL processing approaches for the extraction of the software features from the user reviews is employed in this work. The main aim is to provide lists of early software features which can provide input to domain analysts in the RR process.

The remainder of this paper is organised as follows: Section II describes the related works; Section III presents the proposed approach in detail; Section IV discusses the experiment results and presents threats to validity, and finally, a conclusion and description about future research are provided at the end of the paper.

2. Related works

The related works are divided into three parts: reviews that relate to RE and software product lines, works that are related to feature extraction from NL requirements in software product lines, and lastly the related works in the area of RE that utilise user reviews.

2.1 Related review in the area of RE and SPL

A systematic literature review was conducted by Alves et al. [1] related to RE in SPL. Important findings from their review revealed that there is a need to improve the overall quality of the reviewed studies in terms of empirical validations. Additionally, they have reported that most of the studied do not provide sufficient guidelines for practitioners to adopt the proposed approach and there are very limited commercial or open source tools that are currently accessible, which hinders the practitioners’ adoption of the proposed approach. As for the research trend, a growth in the number of approaches to handle NL requirements in a more automated way is anticipated in the future.

2.2 Related works for feature extraction from NL requirements and SPL

Pertaining to feature extractions from NL requirements process within the SPL context, few authors use online product listings [15], product brochures [11], or web repositories [29] for deriving reusable product lines features. For instance, data mining approaches were used by Hariri et al. [15] to discover common features across products and relationships among features by taking online product listing as their input. Incremental diffuse clustering algorithm and association mining were then used in their study to group common software features. Similar effort appeared in Yu et al. [29], which used project profiles from Sourceforge.net and Softpedia.com as the input to the proposed feature recommendation process. To achieve that, they have introduced an incremental diffuse clustering algorithm for feature discovery and used association mining to augment initial feature profile followed by the
use of KNN approach to make additional feature recommendations. Alves et al. [2] conducted an exploratory study to investigate the suitability of IR technique for identifying common and variable features by comparing Vector Space Model (VSM) and Latent Semantic Analysis (LSA) towards SRS documents as the input to the extraction process. The comparisons were done towards a combination of Hierarchical Agglomerative Clustering (HAC) with LSA, and the combination of HAC with VSM, to observe which one performed better. The findings indicated that with small-size requirements, VSM performed better than LSA. HAC was also used by Chen et al. [6] as an effort to merge requirements into feature trees. Additionally, a systematic literature review for feature extraction from NL requirements in Software Product Lines was reported in [3].

2.3. Related works in the area of RE that utilise user reviews

Few authors have addressed the use of user comments for mobile app reviews and extracted software features with the purpose: evaluating user requests for new features for an app or user requests on their preferences in redesigning the existing features. Various approaches were used in the related works in this area.

For example, Guzman and Maalej [12] used collocation findings to extract fine-grained features, utilised sentiment analysis to extract sentiments and opinions associated to the features, and applied topic modelling to group-related features. They have extracted 32210 reviews for 7 iOS and Android apps and compared the results with 2800 manually peer-analysed reviews. The results indicate that their proposed approach is effective in extracting the most frequently mentioned features. Groups of features are coherent and relevant to app requirements, and sentiment analysis results positively correlate to the manually assigned scores. In Guzman and Maalej [12], extraction process was done by using the NLTK toolkit in which nouns, verbs, and adjectives were extracted from the mobile app reviews, followed by stop words removal. This was then followed by lemmatization process that grouped different inflected words with the same part of speech tagging together (lemmatization group words that are syntactically different but semantically similar). Then, they applied the collocation algorithm provided by the NLTK toolkit for extracting features from the reviews.

Jacob and Harrison [17] developed a prototype, Mobile App Review Analyser (MARA), that is able to extract feature requests from online mobile app reviews and used it to extract such requests from a sample of 136,998 online reviews. MARA was designed to retrieve all the reviews available for an app, mine the content of the reviews for identifying sentences or fragment of sentences expressing feature requests, summarise such contents, and present the summarisation in a user-friendly manner.

Carreno and Windbladh [5] analysed the user reviews available for third-party mobile applications as a way to extract new or changed requirements for future releases of a particular software. In their work, the authors used topic modelling to extract the main topics from the user feedback and evaluated them on different publicly available data sets.

Additionally, Hu and Liu [16], Popescu and Etzioni [24], and Khan et al. [22] proposed various approaches to mine and summarise customer reviews from the web. Hu and Liu [16] proposed to extract frequent product features from online customer reviews, identify opinion sentences (in terms of positive or negative opinions), and provide a summary for collected reviews. Their aim was to provide a summary of frequent product features that can be useful for future customers and manufacturers. In their approach, simple nouns and verb groups were identified through syntactic chunking. Additionally, fuzzy matching was used to deal with word variant and misspelling. Meanwhile, Popescu and Etzioni in OPINE [24] mined product features through noun phrase extractions for parts and properties of product towards the same data sets used by Hu and Liu [16]. Popescu and Etzioni [24] proposed the use of relaxation labelling for finding semantic orientation of words. Popescu and Etzioni claimed that OPINE obtained a precision result that is 22% better than the one reported by Hu and Liu [16]. Khan et al. [22] proposed the use of hybrid dependency patterns to extract product features from unstructured reviews or free text. Their proposed approach exploited lexical relations and opinion context to identify features, and the experiment result indicated a significant improvement in the average precision and recall. The most recent publication found is the work by Hamza and Walker [13], in which they have reported the use of NLP technique in combination with three heuristics against the SRS documents. The heuristics used lingers around the actors and actions identified in the SRS documents. They have presented a qualitative evaluation and provided snapshots on the sample output. However, the precision and recall measurement were recorded in their work. It is realised that the three heuristics used by Hamza and Walker [13] may not be suitable in this current work because it was designed in such a way to suit the sentence structure in SRS documentation only.

To summarise the contributions from the related works presented in this section, Table 2 presents the research objectives, Input (sources for the feature extraction approaches), the Process (technique used), and the Output (Information Extracted) from the related works studied.

The following research gaps are observed based on analysis of existing related works: First, none of the approach from related works used expert reviews as input source to RR activity. Second, although few works looked at user reviews, their focus are primarily on extracting sentiments, user opinions, or common topics for the purpose of improving the same software for next releases, and not focusing on extracting legacy requirements for reuse in the production of a similar system. Finally, none of the related works provided clear guidelines on how to group similar features after the extraction process.

3. The proposed approach

In this paper, the feature extraction process called Feature Extraction for Reuse of Natural Language requirements (FENL) is demonstrated. The overall process is illustrated in Fig. 1a and the algorithm for the summary of automated feature extraction process is provided by Fig. 1b. FENL is separated into four main phases, with the first three phases to be fully automated. At the time this paper is written, the automation of Phase 4 is left for future work. The FENL offers to extract software features from various forms of requirements, such as online software reviews, legacy requirements or product descriptions by using NL processing, and IR techniques. However, for this research, only the freely available software reviews from the Internet is used.

Before going through the FENL process in detail, a description of the data sets used for this research is provided.

3.1. Data sets

As mentioned earlier, SRS are not easily accessible by everyone. In this research, software reviews that are available on the Internet are used as the input to demonstrate the FENL, as an alternative when the SRS are not available. Related works in this area used various forms of requirements that include product descriptions, brochures, use cases, and the most recent ones used the user comments available for the mobile applications. In this case, software reviews compiled by experts that are available on the Internet are chosen.
Table 2
Research Objectives, Input, Process, and Output for the Related Feature Extraction Approaches.

<table>
<thead>
<tr>
<th>#</th>
<th>Author(s)</th>
<th>Research Objectives</th>
<th>Input (Source)</th>
<th>Techniques Used (Process)</th>
<th>Information Extracted (Output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[15]</td>
<td>To discover common features and its relationships</td>
<td>Online product listings</td>
<td>Screen scraper utility, Incremental diffusive clustering, and Can's metrics to determine ideal number of clusters, LSA, VSM, and HAC.</td>
<td>Feature descriptors (sentences) that are clustered together</td>
</tr>
<tr>
<td>2</td>
<td>[2]</td>
<td>To investigate the suitability of IR technique for identifying common and variable features</td>
<td>SRS documents</td>
<td></td>
<td>Feature trees</td>
</tr>
<tr>
<td>3</td>
<td>[12]</td>
<td>To identify app features from reviews and extract user sentiments from the identified features.</td>
<td>Mobile app reviews</td>
<td>Natural Language Processing (NLP) collocation finding, sentiment analysis, and Latent Dirichlet Allocation (LDA) topic modelling.</td>
<td>Nouns, verbs, and adjectives.</td>
</tr>
<tr>
<td>4</td>
<td>[5]</td>
<td>To process user comments to the main topics for requirements engineers to revise the requirements for next releases.</td>
<td>Mobile app reviews</td>
<td>Sentiment analysis and LDA topic modelling.</td>
<td>Main themes or topics</td>
</tr>
<tr>
<td>5</td>
<td>[17]</td>
<td>To identify common topics across feature requests that reside within mobile app reviews.</td>
<td>Mobile app reviews</td>
<td>Predefined linguistic rules to identify words that represent feature requests, LDA, and LDA topic modelling.</td>
<td>Matching feature requests with topic modelled by LDA</td>
</tr>
<tr>
<td>6</td>
<td>[16]</td>
<td>To mine product features and identify opinion sentences.</td>
<td>Online product reviews</td>
<td>Used association mining to find frequent features, syntactic chunking to identify noun and verb groups, and fuzzy matching to deal with word variant and misspelling.</td>
<td>Opinion sentences and summary of reviews.</td>
</tr>
<tr>
<td>7</td>
<td>[24]</td>
<td>To mine reviews in order to build a model of important features</td>
<td>Online reviews</td>
<td>Relaxation labelling and finding semantic orientation of words.</td>
<td>Extraction of noun phrase for parts and properties</td>
</tr>
<tr>
<td>8</td>
<td>[30]</td>
<td>To extract early requirements using NL approach.</td>
<td>User manuals and project reports</td>
<td>Text analysis (pattern matching approach) and topic modelling techniques.</td>
<td>Early requirements such as goals, functions, and constraints.</td>
</tr>
<tr>
<td>9</td>
<td>[22]</td>
<td>To extract product features from unstructured reviews</td>
<td>Online reviews</td>
<td>Regular expressions, Hybrid dependency patterns (combined patterns of noun phrases), and opinion lexicons.</td>
<td>Extraction of noun phrases with evaluative expressions</td>
</tr>
<tr>
<td>10</td>
<td>[13]</td>
<td>To automatically recommend features for Software Product Lines (SPL) and how it can relate to each other.</td>
<td>SRS documents</td>
<td>NLP + Heuristics that are based on actors and actions.</td>
<td>Feature sentences and relationships (e.g., optional or mandatory).</td>
</tr>
</tbody>
</table>

When looking at recent works such as Carreno & Windbladh [5], Jacob and Harrison [17], and Guzman and Maalej [12], the authors have used user comments left by mobile app users. The data used in their works are raw and unprocessed. In this work, in order to obtain more reliable data, the compiled expert reviews are used as the input to feature extraction process. There are some characteristic comparisons in between user feedback or comments and compiled expert reviews, and indicated by Table 3.

Among important aspects filtered out include the data used will most likely to be free from bias, user complaints, moods or sentiments, and the use of unpleasant words. The firsthand reviews from users were not used in this research because this research intends to filter out the user complaints and sentiments. Importantly, this research focuses on information related to the software features available for reuse. Thus, using compiled reviews from experts will be a better choice to suit this purpose. These aspects are deemed important to ensure features extracted are free from noise that might reduce the accuracy of feature extraction results.

To suit to this purpose, two websites that provide a compilation of software reviews by experts have been chosen for online educational software, namely toptenreviews.com and superkids.com. The following subsections are brief descriptions of the websites chosen for extracting the software reviews.

3.1.1. Toptenreviews.com

Reviews in the toptenreviews.com are for users and developers who wanted to get an overview of the product they wanted to buy. These reviews are also beneficial for developers (and domain analysts) who did not get access to SRS and can use these expert reviews as a source of identifying features for the product they want to build without having to initiate the RE process from scratch (reuse of requirements). In toptenreviews.com, software are reviewed by experts and compiled periodically as compared to the firsthand review data sets used in the related works Guzman and Maalej [12]; Jacob and Harrison [17]; Carreno and Windbladh [5]. Data used in these compiled reviews are more formal and contain more

---

information about the product functionalities, and have fewer user complaints such as design flaws or bugs, which make it more usable for the purpose to extract product requirements.

3.1.2. SuperKids software reviews

SuperKids software reviews are written by teams consisting of parents, teachers, and children towards educational software being used. The reviews are compiled periodically and designed to give unbiased information to potential users about software products' functionality. The reviews compiled on this website are provided by a privately held company, Knowledge Share LLC, a completely

---

Table 3
Characteristic comparisons for user comments and expert reviews for software products that can be extracted from publicly available sources.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>User Commentor-Feedback</th>
<th>Expert Reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features to be added</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Current Features</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Step by step how to use</td>
<td>Yes, Sometimes</td>
<td>Yes, Sometimes</td>
</tr>
<tr>
<td>Bugs report</td>
<td>Yes</td>
<td>Sometimes</td>
</tr>
<tr>
<td>User complaints</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Moods or sentiments</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Size</td>
<td>Very Large, Managed</td>
<td></td>
</tr>
<tr>
<td>Probability to find unpleasant words</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Bias</td>
<td>Very Likely, Unlikely</td>
<td></td>
</tr>
<tr>
<td>Authors</td>
<td>Anyone can write, Experts appointed by organisations</td>
<td></td>
</tr>
</tbody>
</table>

3. Experimental procedure

The FENL process is separated into four phases as illustrated earlier in Fig. 1. Accessing Natural Language Requirements, Terms Extractions, Feature Identification, and Formation of Feature Models. The open-source scraping tool is used to select the text from the online software reviews and converted them to text files. The following subsections describe the automated processes for Phase 1 until Phase 3.

3.2.1. Phase 1: accessing requirements (software reviews)

The purpose for Phase 1 is to find software reviews available on the Internet as an alternative to using SRS documents. To demonstrate this, 32 software reviews pertaining to children learning posted at toptenreviews.com and superkids.com have been scraped by using open-source screen scraper utility. The 32 software reviews came from four subcategories under the domain of online learning software for children:

a) PL1: Preschool Learning (10 compilations)
b) PL2: Algebra Learning (10 compilations)
c) PL3: Language and Reading Software (3 compilations)
d) PL4: Creative Writing (9 compilations)

The reviews were extracted in June 2015. The source of raw review extracted have been made available online.3

3.2.2. Phase 2: terms extraction to identify similar reviews

Documents that have been scraped in Phase 1 are used as the input to the terms extraction process. Fig. 2 lists out the algorithm used for the terms extraction:

First, each document went through text preprocessing to remove stop-words (words such as “a, are, the, to, an, at, is,” and more, which do not provide added meaning if considered). Additionally, the punctuations, numbers, and special characters are filtered out from the review documents. In Step 2, WordNet lemmatization was used. Lemmatization is a process of grouping together the different inflected forms of a word so they can be analysed as a single item. For example, words such as “coloring, colored, and colors” are now being referred to as the basic word “color.” This helps in reducing the number of words extracted. In Step 3, the words were tagged with Parts of Speech Tagger from NLTK, and only selected nouns, verbs, and adjectives. In Step 4, all terms that occurred only once and twice are removed, and in Step 5 the remaining words are now tabulated in the term-document-matrix.

Figs. 3 and 4 show a sample input-output from the term extraction process.

Steps 1 until 4 in Fig. 2 are repeated for all selected reviews. A final spreadsheet contains n-termsby m-documents (terms-document matrix). Based on the terms collected, the term weights were calculated by using the term frequency inverse document frequency, tf-idf. The tf-idf weight is a weight used in IR and text mining to evaluate how important a word is to a document in a collection. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the collection [26]. These tf-idf value will be used to determine the document similarities with LSA – will be described in Section 4.3. The main outputs for Phase 2 are important terms (verbs and nouns) in each document and their occurrences. These terms will later be used in Phase 3 to identify related reviews (documents).

3.3. Phase 3: feature identifications

Phase 3 firstly determine the software reviews (documents) that are related based on the term-document-matrix obtained in Phase 2. Phase 3 was divided into three subprocesses as described below:

3.3.1. Phase 3a: identify similar documents

By using the tf-idf values obtained from the Phase 2, the position of related documents in document space can be obtained by applying the Singular Value Decomposition (SVD) through the LSA implementation. SVD computation is LS is what distinguishes the LSA from the more traditional LSM. SVD computation reduces the dimension of the document so that only relevant vectors are considered while the traditional VSM uses the original dimension of the document, which makes it less effective than LSA. Moreover, VSM uses the keyword matching techniques as compared to concept-based techniques that are applied in LSA. The detailed explanation of LSA implementation with the SVD calculation is available in Deerwester et al. [10]. SVD calculation was applied to reduce the dimension of matrix representation to three different matrices: S, U, and V. In MATLAB R2011a, the following command is used to decompose the term-by-document matrix into matrices S, U, and V:

\[
[SUV] = \text{svd}(A)
\]

A is the tf-idf values obtained in the term-document-matrix. The SVD calculation is performed towards A to obtain three matrices: S, U, and V, where importantly matrix V will hold the positions (coordinates) of the reviews in the document space. Thus, in this current research experiment, rank 2 approximation was implemented, so that the first two columns of matrix V are kept:

\[
V_k = V(:, 1:2)
\]

By keeping the dimension to lower dimensions, the SVD computation should bring together the terms with similar co-occurrences. As a result, rows in V_k contain the coordinates of individual document vectors. These coordinates when projected to an x-y plane will indicate the position of all documents in the problem space. As a result, unrelated documents were discarded: the documents that are clearly far from other documents in the document space.
Step 1: Remove stop-words, punctuations, numbers, and special characters by using text preprocessing.
Step 2: Apply WordNet Lemmatization\footnote{WordNet is a lexical database like Thesaurus that organizes English words into sets of synonyms called synsets.}.
Step 3: Apply the Part of Speech Tagging from NLTK\textsuperscript{2} to the document and select the required terms (verbs and nouns).
Step 4: Remove all terms that occur only once and twice.

Fig. 2. Terms extraction process.

Fig. 3. Sample of raw text from the online reviews (input), scraped and saved as text file.

(as computed by LSA) will not be taken into the next phase of the experiment. The basic K-means algorithm is used to confirm the groupings of the documents. The K-means algorithm is a commonly used clustering algorithm with the aim to optimise an objective function (the distance) that is described by the equation:

\[
E = \sum_{i=1}^{c} \sum_{x \in C_i} d(x, m_i)
\]  \hspace{1cm} (1)

\(m_i\) is the centre of cluster \(C_i\), while \(d(x, m_i)\) is the Euclidean distance between point \(x\) and \(m_i\). Fig. 5 is the algorithm for K-means \cite{14}:

In Matlab, the following command is used to demonstrate the K-Means clustering:

\[
[idx, c, sumd, d] = \text{kmeans(coordinate, 4, 'start', 'sample', 'replicates', 100, 'maxiter', 1000, 'display', 'final')}
\]
The Fuzzy C-Means, FCM clustering is as well applied for grouping similar documents. FCM have been used in related software engineering publications such as Chong et al. [7] and Noppen et al. [23]. In fuzzy clustering, each data point has a degree of membership (or probability) of belonging to each cluster [4]. A data point that resides closer to the centroid will have a stronger membership to the cluster center. Summation of all membership in a cluster should be equal to 1. The algorithm for FCM is divided into the following steps:

1. Randomly select cluster centre
2. Initialize $U = [U_{ij}]$ matrix $U^0$
3. The $U_{ij}$ is calculated using:

$$U_{ij} = \frac{1}{\sum_{k=1}^{c} \left( \frac{|x_i - c_k|}{|x_j - c_k|} \right)^{2/m}} \quad (2)$$

4. At k-step, the center vectors are calculated by using:

$$C_k = \frac{\sum_{i=1}^{N} U_{ij}^k x_i}{\sum_{i=1}^{N} U_{ij}^k} \quad (3)$$

5. Update $U^k$, $U^{k+1}$ by using Eq. (2) from Step 2.
6. If $\|U^{(k+1)} - U^{(k)}\| < \varepsilon$ or the minimum J is achieved, then stop. Otherwise return to step 2.

Before running the FCM, the following parameters must be specified:

- the number of clusters, $c$
- the fuzziness exponent, $m$
- the termination tolerance, $\varepsilon$

Additionally, the Self Organizing Map, SOM is used to cluster similar documents [27]. Having more choices of clustering is better to provide the comparison on which clustering algorithm can work best for the extracted data set. Self-organizing maps (SOM) segment input vectors according to how they are grouped in the input space. The selforgmap tools in Matlab was used to demonstrate this clustering method.

After running all clustering algorithms, the following are graphs indicating each clustering solutions (Fig. 6a–c).

Fig. 6a–c indicates the results from clustering the documents by using three different clustering algorithms. To determine which clustering method produces the closest results to the one made by human, as appeared in toptenreviews.com, we have measured the silhouette of the three clustering results. The silhouette values are measured to check if there is any document that might be clustered wrongly. The silhouette plot in Fig. 7a–c display how close each point in one cluster is to points in the neighboring clusters.

The silhouette measure ranges from +1, indicating points that are very far from neighboring clusters, thus indicating the compactness of the cluster elements. Value 0 indicates the points that are not distinctly in one cluster or another. The points on the negative side of the graphs indicate that points that probably assigned to the wrong cluster. Fig. 7a, b and c indicates the results, in which documents clustered by the Self Organizing Map to obtain a better clustering result when compared to the document clustering results by K-Means and Fuzzy C-Means.

Unfortunately, not all the documents clustered by SOM resemble the categorisation made by human. The SOM placed all documents within 3 main clusters only. Additionally, it is observed that the third cluster produced by SOM consists only 1 document, leaving 2 major clusters. This is very far from the original categorization (as appear in toptenreviews.com and superkids.com). The original documents if manually grouped can be separated into four product lines. After running these three clustering algorithm, it is observed that K-Means and FCM to produce higher F-Measure values (see Fig. 8). Anyhow, when observing the Silhouette measure, K-Means indicated some negative values depicting some of the items in this clustering solution to be wrongly assigned. Although SOM produces no negative values in the silhouette measure, the F-
Measure values obtained is not satisfactory when compared to the K-Means and FCM. Based on this experiment data, FCM is the best solution for clustering the related document (Phase 3a). Thus, this research is suggesting that future experiments with FENL to adopt the FCM algorithm to be used in the Phase 3a.

3.3.2. Phase 3b: extraction of phrases that represent features

In the related works such as Alves et al. [2] and Weston et al. [28], similar structure of requirement statements are compared because their research used standard requirement documents (i.e., use case specifications and SRS documents). However, when dealing with unstructured documents such as product reviews, measuring sentence similarities is not easily achieved. This is because reviews were written freely and did not follow sentence structure such as sentences that exist in SRS. For example, in SRS, sentences are constructed in the form of Verb + Direct Object, namely “The user shall click on the Exit Button to terminate the application.” With SRS, linguistic pattern in the extraction algorithm can specifically be targeted on sentences consisting “shall, should, must, etc.”, followed by verb and objects (Exit Button), thus, the sentences can directly reflect the functional requirements of a system. When comparing to sentences in software reviews (free written text), there is no standard linguistic pattern that resembles the functional requirements. This makes it hard to perform comparison towards sentences, in

![Graph](image)

**Fig. 6.** (a) Position of reviews in document space that can be grouped into four categories (K-Means). (b) Position of reviews in document space that can be grouped into four categories (Fuzzy C-Means). (c) Position of reviews in document space that can be grouped into four categories (Self Organizing Map clustering).
which there is no guarantee that sentences in review documents contain the “shall statement” that can represent functional requirements.

The description of features is referred to as “a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems” as described by Kang et al. [20]. This research is focusing on extracting the phrases (bigrams or trigrams words, i.e., combination of nouns, verbs, and adjectives) which may bring out the user-visible characteristics of a system. In this context, for example, terms such as “number recognition”, “learning colors”, “interactive tutorial”, “multiplechoice quiz” are considered as terms that represent product features.

To cater to this purpose, in this phase the Parts of Speech (POS) tagger provided in NLTK4 is used to extract the combination of phrases that occur in form of «adjective, noun» or «noun, adjective» AND «verb, adjective» or «adjective, verb» AND «verb, nouns, adjectives». All possible sequences or arrangements of verbs (past or present tenses) are considered as well in this linguistic tagging selection. Table 4 describes the acronyms for the linguistic tags used in this work to represent various forms of verbs, adjectives, and nouns.

Based on the understanding of acronyms in Table 4, the following configuration is sufficient to extract the phrases that represent the features (especially functional requirements) of the software.

From Fig. 9, the tag NNP is used to label the combination of parts of speech category as a Normal Noun Phrase that may consist either Noun + Verb only, or Noun + Adjective + Verbs. The NNI is used to label a longer combination of parts of speech: NNP + NN means Normal Noun Phrases tagged earlier and additional nouns that occur afterwards. Another example is JJ + NN = NNI is used to tagged adjectives and Nouns that forms a longer Noun Phrase, labelled as NNI.

This linguistic pattern is believed to be relevant to the current experiment context, in which it can possibly expose the user visible characteristics of the software being reviewed. Fig. 10 presents a sample of phrases extracted following this procedure.

To determine the accuracy of the extraction algorithm, the extracted phrases are listed out and compared with the one extracted manually (see Section 4 for Manual Extraction procedure). Next, similar features can be grouped based on their semantic similarity. To accomplish this, two versions of word overlap algorithms are used. The results are compared in order to measure efficiency among the methods chosen.

### 3.3.3. Phase 3c: grouping similar features

This section demonstrates the Phase 3c of FENL: Grouping Similar Features. Grouping similar (common) features and segregating the variant features is the essential aspect of the SPL development that realises software reuse. In order to group common features extracted, the surface similarity of each phrase is calculated based on the modified version the Word Overlap Metrics. The comparison between the original word overlap metrics [25] and its modified version is presented in this section.

Grouping similar features are done by selecting the first feature to be a member of the first group. At each iteration, the algorithm decide on whether a new input feature should be inserted into an existing cluster or to create a new cluster. This decision is made based on the similarity threshold obtained by the Word Overlap Metrics. In Rossi et al. [25], the Word Overlap Metrics take the number of common words between the first phrase and the second phrase, and normalise it by the total number of words in both phrases. The original word overlap metrics proposed by Rossi et al. [25], calculate the number of common words between a sentence S and a sentence C, normalised by the total of words in sentence S.

<table>
<thead>
<tr>
<th>#</th>
<th>Acronyms Used</th>
<th>Representing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NNP</td>
<td>Nouns</td>
</tr>
<tr>
<td>2</td>
<td>JJ</td>
<td>Adjectives</td>
</tr>
<tr>
<td>3</td>
<td>VB</td>
<td>Verb Base Form</td>
</tr>
</tbody>
</table>

---

4 NLTK is Natural Language Toolkit for use with Python programming available at nltk.org.
added to total number of words in cluster C, as given by the formula:

\[
W_\text{ol}(S, C) = \frac{\#\text{CommonWords}(S, C)}{|S| + |C|}
\]

The implementation was done in Python 2.7. The data obtained were analysed and sorted in Microsoft Excel. The code snippet for finding word that are related to “choose character” is shown in Fig. 11.

As for demonstrating the modified word overlap metric, Fig. 12 presents the algorithm for its implementation. Here, it is aimed to observe the groupings of words that share similar head nouns. First, the leftmost words in the first cell are compared to the leftmost words in the following cell (left overlap). The same process is done with the first word after the space (right overlap). The complexity of this task is not that high because most of the extracted phrases in this current work are bigrams and trigrams only.

**Fig. 7.** (a) Silhouette measure for K-Means. (b) Silhouette measure for FCM. (c) Silhouette measure for SOM.
Total phrases with common words (word overlap) will be computed and sorted together. This process is repeated to compare all extracted features for each category. The surface similarity, namely the grouping of phrases that share the same head words, for example “access lesson” and “access tutorial” is grouped together based on the same head words. It is understood that “tutorial” might not mean the same as “lesson”. The word tutorial in this contact may be regarded as a sub-feature derived from the word “lesson”, in which this capability not covered by the enhanced word overlap metrics algorithm proposed. However, the groupings formed by the word overlap metrics may provide early input to domain analysts for further analysis in the next process of RR.

The grouping results obtained from the word overlap and its modified version are then passed to Phase 4. In Phase 4, feature models that are based on FODA model by Kang et al. [20] will be manually constructed. However, the automation for constructing feature model is currently beyond the current research scope.

4. Evaluation and discussion

In order to evaluate the quality of the proposed approach, the evaluation procedure has been divided into two main phases. First, the manual feature extraction was conducted to form truth data set for calculating the accuracy (recall, precision and F-Measure). Sec-
Fig. 10. Sample of phrases extracted from software reviews.

```python
i=0
c1=[]
for line in lines:
    eachlines = ''.join(line)
    # assign the features found in the first cluster
    if eachlines.count("choose ")==1 or eachlines.count("characters ")==1:
        print 1
c1[i]=eachlines
    i+=1
else:
    print 0
```

Fig. 11. Source code snapshot in Python for grouping similar words by using the original word overlap algorithm.

MODIFIED WORD OVERLAP ALGORITHM

**Input:** Phrases (bigrams and trigrams), # of occurrences, p

**Output:** Sum of occurrences for each word overlap, sum

For each phrase,

Find the location of the first space, n

**LEFT OVERLAP:**

1. Find the word before n (left) in the current cell, m
2. If m matches with the word before space in the next cell,
   return the number of its occurrences, p

Sum up of all p, sum=sum+p

Fig. 12. Modified Word overlap algorithm.

...and, the proposed approach was evaluated in terms of the accuracy of the features extracted. This section describes both procedures in detail.

4.1. Manual feature extraction

Since no truth data set exist for comparing the extraction result, it is essential to manually construct one. The participants for manual extractions were school teachers and system analysts. Seven school teachers were invited to participate in this manual extraction. Invitation letter with a complete set of instruction and sample expected output were sent out through emails. The teachers came from various levels of educational institutions (ranging from kindergarten to secondary schools) and nine system analysts from local software development companies to conduct the manual feature extractions. The purpose of having system analysts to participate is because it is necessary ensure that the output extracted by teachers (as domain experts) are in accordance with the sys-

---

5 Sample invitation letter to teachers and the instruction is available at: https://www.dropbox.com/s/393u4p56ljdcoh/LetterTeachers.docx?dl=0.
4.2. FENL evaluation results

The presentation of the results in this section was separated into three parts: The data sets, the feature extraction results, and the feature grouping results.

tem analysts’ perspectives on what can be considered as software features. Additionally, in organisations, System Analysts usually have experience dealing with various system requirements and are sometimes involved in system purchasing, including the learning system. In order to form the truth data set, features highlighted by teachers are combined with the one produced by system analysts. Some features highlighted by the system analysts but not highlighted by the teachers were added to the truth data set. The 32 reviews collected were divided according to categories (as stated earlier). The teachers were given the reviews that are related to their area of teaching expertise, namely teachers who taught Writing and Comprehension were asked to do manual extraction for the Creative Writing Software and Language and Reading Software, teachers teaching Mathematics were asked to perform manual extraction for Math and Algebra Software, and kindergarten teachers were asked to do manual extraction for the Preschool Learning software.

The following procedures which have been used by Carreno and Windbladh [5] for conducting the manual classification of user reviews is adapted Table 5.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Action</th>
<th>Role and Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Read reviews.</td>
<td>Teachers and System Analysts</td>
</tr>
<tr>
<td>2</td>
<td>Highlight potential features and record on spreadsheets.</td>
<td>Teachers and System Analysts</td>
</tr>
<tr>
<td>3</td>
<td>Manually consolidate features that have similar meanings</td>
<td>First Author</td>
</tr>
<tr>
<td>4</td>
<td>Manually group features that have similar meanings</td>
<td>First Author, System Analysts</td>
</tr>
<tr>
<td>5</td>
<td>Construct Feature Model based on the feature groupings</td>
<td>First Author, System Analysts</td>
</tr>
</tbody>
</table>

### Table 5

Steps for manual feature extraction.

The teachers were given user reviews and asked to extract features that represent the software features. The description on the definition of features, following the feature definition provided by Kang et al. [21] was provided. All of these were done through emails and when questions arise, questions are attended through emails and phone calls when necessary. As the output from the manual classification, the participants submitted list of features extracted from the reviews, recorded in spreadsheets. Time taken for the whole extraction process is recorded as well.

Following the completion of manual feature extraction performed by teachers, the system analysts were asked to read through and perform the manual feature extraction from the same reviews used by teachers. The first author will then consolidate the features extracted by system analysts and the one extracted by teachers, and completed the truth data set. The complete set is now compiled as the truth data set for the recall and precision comparison with the auto-generated feature extraction approach. However, it is essential to mention that the truth data set created for this experiment do not represent the absolute truth of the data, as manual judgment may vary from human to human, and this is identified as the internal threat to validity.

4.2. FENL evaluation results

The presentation of the results in this section was separated into three parts: The data sets, the feature extraction results, and the feature grouping results.
True Positives are obtained by calculating how many feature exists within the data set and extracted by the approach. False negatives are the actual features that are not extracted by the approach, while False Positives are features that are returned by the algorithm but they do not exist in the truth data set. The number of True Positives, False Positives and False Negatives are obtained by matching the features extracted by the approach as compared to the one exists in the truth data set. The Eqs. (4), (5) and (6) are modelled in Microsoft Excel, as the features extracted are stored in Microsoft Excel spreadsheets too.

Based on Table 7, The F-measure value is increased when consolidated data set from teachers and analysts were used. Relatively lower precision and recall were observed for category Language and Reading Software, (51.15% precision and 80.72% recall).

It is noted that the F-Measure records no improvement (62.61% for dataset from teachers as compared to the consolidated dataset, 62.62%). Both F-Measures coincidentally appear similar, however the Recall and Precision values are not – see details in Table 7. This might be due to the significant lower number of word lists (725 words) that resided within Language and reading software – see dataset details in Table 6. Some other possible reasons due to the language teachers only picked up the features that were visible to them from pedagogical aspects and not software features aspects. For example, some phrases extracted by the teachers in the review that only consider pedagogical aspects are like: “thorough explanation”, “direct approach”, or “reading for evidence”. These features seem to cater on the pedagogical aspects of learning instead of visible software functionality, as what defines features in this current research context (as explained earlier in this paper). Additionally, it is fortunate to see that some important features that were not highlighted by teachers were actually considered by analysts and also detected by the automated approach. For example, the following are some of the features extracted by analysts and the automated approach, but not extracted by teachers from the reviews: “renewal updates”, “score section” or “track progress”. Thus, having a consolidated data set (from teachers and analysts) does have the impact on improving the F-measure. Additionally, the time taken to complete the extraction process was recorded. Overall, the total time taken for the manual extraction ranges from 55 min to 2 h per category while the automated approach only consume less than 10 s.

4.3. Discussions

In this section, the findings from the experiment conducted are discussed. First, discussion pertaining to the data sets used in the experiment is provided. Second, discussion about the observation made towards the experiment results is provided. Thirdly, discussion on the research limitations and future plans is presented.

4.3.1. Discussion on the data sets used

Reviews that are compiled by experts have been chosen because the research purpose is to extract software features that can aid the RR process. Hence, the focus was to extract the software features that are available in the software reviews which are free from customer complaints and sentiments about a software product. In this section, discussion about the data sets used in the experiment is provided.

4.3.1.1. Smaller data sets when compared to related works. Due to limitation to access the SRS requirements, compilation of reviews containing software features available on the web can help domain analysts to gauge the idea about the features for a software product prior to development. For example, the reviews in Topenreviews.com provide expert reviews of the top ten software (according to categories), which are compiled periodically. Referring to topreviews.com, the reviews are independent reviews by experts, which emphasised four important issues including hands-on use and evaluation, scoring and ranking, editorial independence, and updates. Therefore, the reviews provided for each software in the topreviews.com or superkids.com are not the firsthand user feedback such as the user reviews from mobile apps used by related works Guzman and Maalej [12]; Iacob and Harrison [17]; Carreno and Windbladh [5]. For example, Guzman and Maalej [12] evaluated their approach with 32210 reviews from 7 apps, Iacob and Harrison [17] used 3279 reviews from 161 apps, while Carreno and Windbladh [5] used data sets that contain 2651 reviews from 3 apps (as shown in Table 8). The firsthand reviews from user feedback appeared in these three related works are important for developers who want to redesign the features in the product or improve/remove the current features with negative sentiments. However, the RR intention is not the main focus for their research.
The use of compiled reviews in the case of current research justifies the needs to focus on extracting software features for reuse.

In this work, the length of expert reviews extracted varies between the review documents; ranging between 400 words to 1200 words each. When compared to the firsthand user comments used in the first three related works in Table 2, each comment left by users is relatively shorter (of about 30–40 words each, see Fig. 15 for sample mobile apps review and Fig. 16 is the sample of expert review used as the input in this research). This reason makes the current research data set look smaller when compared to the related works. Furthermore, the data used in this research are reviews compiled by experts, in which most sentiments and user complaints are already minimised.

4.3.1.2. Data sets used are from manual extraction. First, unstructured reviews from the web are used as input to the experiment in this research. These data are raw, unedited, and have not been used in the RR area, to the best of the authors knowledge. The reviews might contain more user opinions and complaints instead of product features. The reviews may also be outdated and new features have been added to the software at the time these reviews are being used for the current experiment.

In manual extraction, first, teachers manually extracted what constitute as software features. Second, system analysts verified the features extracted by teachers to satisfy what constitutes features from technological perspective. The purpose of having teachers and system analysts to perform the manual extraction is because the manually extracted data are intended to be used as truth data sets. However, it is necessary to point out that the extracted data might not 100% accurately describe the absolute truth. This may have some effects on the recall and precision measurement. Although selected teachers are domain subject matter experts in the subject for online learning software assigned to them, they however might not have the technical speciality in terms of identifying software features. It has been observed that the number of features increase when the system analysts performed the verification.

4.3.1.3. Data sets used are for various groups of audiences. In the experiment, reviews from four learning categories have been extracted. These four categories of software are dedicated for various age groups of users. For example, the software within preschool learning categories are dedicated to children aged six and below, and usually the design of user interface is different to cater to children, which may constitute bigger font sizes or colourful animations to attract the attention of children. When looking at the other categories of reviews such as algebra learning, in which the audience may range from older primary school to teenagers in secondary schools, thus the user interface should be a little more mature when compared to the user interface for the preschool software categories. This research however is not focusing on the user interface extraction (non functional requirements). Here, the authors only interested in extracting software functionality (software features) by picking up the combinations of nouns, verbs, and adjective. Therefore, since the focus is to extract features, using various dataset from software categories for various age groups of users in this research experiment will not effect the features being extracted.

4.3.2. Discussion on the extraction results

As the final output from the experiment, the FENL process has extracted software features from the reviews. The word overlap grouping result indicates possible combination of features that can be fed to domain analysts as early features in a similar product development. The phrases extracted by the FENL can be manually transformed to a feature model, as shown in Fig. 17.

This information can be beneficial to the domain analysts. Domain analysts will be notified regarding important features for the software, namely features relating to algebra principles, printable colouring activities, and video tutorials. This recommendation, although not 100% accurate, can help analysts to have the main features that exist in publicly available software reviews. Obviously, the proposed FENL process may reduce the time spent for requirements engineers to read the entire customer reviews in order to find reusable software features.

4.3.3. FENL result comparison with related works

In terms of clustering similar documents (Phase 3a), the experiment in this research indicates that FCM to produce the best results, in terms of silhouette values and the F-Measure obtained. Although K-Means is not the worst, but it produces some negative results in the silhouette values, giving the sign that some items might not be correctly clustered. Other works that uses clustering algorithms to cluster similar documents include [18] that uses SOM to cluster similar documents, [19] that enhances the FCM with the help
of Particle Swarm Optimizations. However, these works cannot be used for results comparisons since the dataset used are totally from different domains.

The average Recall, Precision and F-measure results obtained by the FENL are compared with related works that reported a similar evaluation measure, for similar data domain — user reviews. The aim is to observe how comparable the proposed approach is, at least on the overall performance. Table 9 records the comparison made.

From Table 9, FENL reported to record lower F-Measure compared with Khan et al. [22] and Carreno and Windbladh [5], but performed slightly better compared to work by Guzman and Maalej [12]. In general, the larger F-Measure is, the better the proposed solution is. Observing from Table 9, F-Measure of 0.708 (although not the highest among all works) is a good indication that FENL to perform comparably with other related works. However, it is noted that the results may varies according to the size of data set used as well.

4.3.4. Research limitations
This section discusses about the challenges faced in this research and some future plans to be taken up.

4.3.4.1. Clustering semantically similar phrases remains as a challenge
In this experiment, the word overlap metrics was used while none of the clustering algorithms were utilised in order to suit to the needs in demonstrating Phase 3c. Although clustering algorithms were mentioned in other related works such as Alves et al. [2], Weston et al. [28], and Davril et al. [9], clustering based on keywords occurrences is not logical when experimented with the current data set used in this research. Grouping single words based on the occurrences in bag of words is possible; however, it will not suit to the current research needs as features represent characteristics of software products (which are represented by more than

<table>
<thead>
<tr>
<th>Feature Extraction Approach by related works</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>[12]</td>
<td>0.582</td>
<td>0.520</td>
<td>0.549</td>
</tr>
<tr>
<td>[5]</td>
<td>0.941</td>
<td>0.67</td>
<td>0.782</td>
</tr>
<tr>
<td>[22]</td>
<td>0.79</td>
<td>0.717</td>
<td>0.752</td>
</tr>
<tr>
<td>FENL</td>
<td>0.62</td>
<td>0.822</td>
<td>0.708</td>
</tr>
</tbody>
</table>

Fig. 15. Sample of user comments from mobile app (Angry Birds) that are relatively shorter, and come with complaints and sentiments.
Fig. 16. Sample of expert reviews from superkids.com for Algebrator software.

("Note each expert review is lengthier when compared to user comments").

Fig. 17. Grouping formed by word overlap for the head word “recognition”. Suggested feature model that can be manually constructed with the “recognition” group phrases.

one word). However, to semantically group features (bigrams or trigrams words) by using automated approach in NLP is not that simple. Thus, this research only choose to find the feature similarity in terms of matching the similar head words by using the word overlap metrics and group them in semi-automated manner. It is beyond the current research scope to find the solution for clustering similar sentences or phrases at this point of time. This challenge is left for future NLP research to solve.

4.3.4.2 Integration of the FENL process. The current implementation of semi-automated process, the FENL, is conducted in a laboratory setting and relies on three applications: Python, MicrosoftExcel, and Matlab. This implementation requires the researcher to have skills in Python programming, Microsoft Excel, and Matlab programming. More people can benefit from this implementation better when the process is integrated into a single platform.

5. Conclusions and future works

A Feature Extraction process for Reuse of NL requirements has been presented in this paper. This approach utilises the techniques from IR and NLP. Thirty-two software reviews for online learning software compiled by experts that are available on the Internet have been used as the input to the proposed feature extraction process. Latent Semantic Analysis with Singular Value Decomposition has been used to find similar review documents. This is followed by applying various clustering algorithms to cluster similar review documents, with Fuzzy C-Means clustering to produce the best clustering solution for the current data set used (Phase 3a). To extract software features, linguistic tags are used (Phase 3b). The proposed extraction approach is then validated by using precision, recall, and F-measure. A precision of up to 87% (62% average) and a recall of up to 86% (82% average) were obtained. Additionally, word overlap metrics and its modified version were used in finding similar head-words in the phrases as an effort to semi-automatically group similar features (Feature 3c).

The extracted software features when grouped manually may provide early input to domain analysts in the requirements reuse process. In the near future, this approach can be enhanced with other machine learning techniques to improve the modified word overlap metrics for automating the grouping of extracted features, semantically (Phase 3c). Additionally, the authors plan to integrate the proposed approach in this paper into a single platform with an interactive visualisation. Lastly, the automation for the construction of feature model to illustrate relationships between features extracted in Phase 4, is also left for future works possibly by employing the association rules mining.

Acknowledgement

This research is funded by the Ministry of Higher Education Malaysia with Research Grant # FP050/2013A, with Assoc. Prof. Dr. Zarina Baharuddin as the principal investigator from the Department of Software Engineering, FSKTM of the University of Malaya.

References
