Mining Domain Knowledge from App Descriptions

Yuzhou Liu[a], Lei Liu[a], Huaxiao Liu[a]*, Xiaoyu Wang[a], Hongji Yang[b]

[a]College of Computer Science and Technology, Jilin University, Changchun 130012, China

[b]Centre for Creative Computing, Bath Spa University, Corsham SN13 0BZ, England, UK

Tel:+86-0431-85166810 liuhuaxiao@jlu.edu.cn

Abstract

Domain analysis aims at obtaining knowledge to a particular domain in the early stage of software development. A key challenge in domain analysis is to extract features automatically from related product artifacts. Compared with other kinds of artifacts, high volume of descriptions can be collected from app marketplaces (such as Google Play and Apple Store) easily when developing a new mobile application (App), so it is essential for the success of domain analysis to obtain features and relationship from them using data technologies. In this paper, we propose an approach to mine domain knowledge from App descriptions automatically. In our approach, the information of features in a single app description is firstly extracted and formally described by a Concern-based Description Model (CDM), this process is based on predefined rules of feature extraction and a modified topic modeling method; then the overall knowledge in the domain is identified by classifying, clustering and merging the knowledge in the set of CDMs and topics, and the results are formalized by a Data-based Raw Domain Model (DRDM). Furthermore, we propose a quantified evaluation method for prioritizing the knowledge in DRDM. The proposed approach is validated by a series of experiments.

Keywords: domain analysis, feature extraction, app descriptions, data analysis

1. Introduction

With the development of software industry, there always exist some products similar or related to the software to be developed. Whether these products come from inside of the company itself or extensive markets, they provide the software development with reference assets, such as requirements, use cases, architecture and frameworks[1] . Based on these assets, domain analysis is conducted to gain knowledge in the early stage of software development, helping the developers understand and define a particular domain accurately. This supports a quick start of the development process and benefits the reuse of source code or other higher level life cycle artifacts. In this way, the success of domain analysis can effectively improve the product competitiveness, including less time-to-market and high quality products [2] .

In domain analysis, feature is a basic notation used to describe a user-perceived characteristic of a system, and feature extraction is one of core work to obtain such information [3] . In the process of feature extraction, texts related to the products are always taken as an important analyzing object because they are a kind of main forms recording the information of software system. In general, software related texts can be classified into three types [4] : 1) the first type is given by the developers to support or record the software development process, such as requirement specifications; 2) the second type is also given by the developers but intended for potential users to introduce the software, such as descriptions, brochures; 3) feedbacks given by software users are the third type, which is often expressed as reviews or comments. Initially, most of domain analysis methods focus on utilizing the first type of texts, and rely upon analysts reviewing them to obtain domain knowledge. Thus, it is very labor-intensive [5] [6] . In recent years, fast growth of the second and third type texts from various sources makes more and more data can be collected easily. Such high volume of

texts makes it difficult to understand or analyze manually. By introducing data science into the field of domain analysis, many researches propose approaches that (semi-)automatically generate feature diagrams, clustered requirements, keywords or direct objects through systematic or quantitative processes of data analyzing. So as to mine domain knowledge from such natural language-based documents to better support the subsequent software development process.

Mobile application (App) is a kind of software system carried on mobile terminals and grows in popularity rapidly[7] . App descriptions are the introductions of app products and belong to the second type of texts introduced above. Compared with the first type of texts, app descriptions can be collected more easily due to the existence of app marketplaces. For example, when we want to develop an app for social communication, there are hundreds of apps within the same domain in Google Play. Meanwhile, because app descriptions are given by app providers, they contain denser domain knowledge than user feedbacks. Therefore, developers can obtain the main feature information of apps by analyzing their descriptions. Considering the description of an app 'Instagram' shown in Table 1, it contains the main features of the app, such as the common features 'discover accounts' and 'Post photos and videos', and the variable feature 'combine multiple clips into one video'. Similarly, more features with their attributes can be extracted from a huge number of related app descriptions. By using such information synthetically, the domain analysis can be performed to effectively support the development of a specific app. Hence, app descriptions, as a kind of abundant and useful data, cannot be ignored in the domain analysis of app products. However, due to their huge amount and lack of expression standard, intensive efforts are needed to analyze app descriptions.

To solve this problem, we propose an approach that automatically mines domain knowledge, including features and their relationship, from app descriptions. We define Concern-based Description Model (CDM) and Data-based Raw Domain Model (DRDM) to formally describe the knowledge in a single app description and overall knowledge in the domain separately. In addition, a quantified method is given to prioritize the knowledge for facilitating their application in practice. For the purpose of evaluation, we conducted a series of experiments with 574 app descriptions from Google Play.

Firstly, we conduct a quantitative evaluation of each method in our approach by comparing them with several well-established methods. The results show that our feature extraction method has a precision of 86.15% and a recall of 83.45% on average, which indicates the good performance of our method. Also, our topic modeling method can obtain more meaningful and understandable results than LDA, and our feature clustering method can achieve about 10 percent improvement over *K*-means in purity. These results verify that our approach can obtain domain knowledge from app descriptions effectively.

Secondly, we give a case study and surveys to compare the usefulness of report generated in our approach with raw data of app descriptions and the feature model constructed by the approach proposed in [8] . By analyzing the results of the surveys with statistical methods, we find that the participants using our report can complete the tasks with significantly shorter time than the ones using the other two materials. Furthermore, we also find that our approach are more adaptive for overall analyzing the whole domain and generating creative ideas, which are important tasks in domain analysis.

The paper is organized as follows. Section II presents the related work. Section III gives problem statement in our research and an overview of our approach. In Section IV, the automatic proceeding of feature extracting and modeling from app descriptions is introduced. The integration and prioritization of domain knowledge are provided in V. Finally, our experiments and the conclusion are shown in

Section VI and VII separately (respectively).

Table 1 An example of App descriptions

Instagram is a simple way to capture and share the world's moments. Follow your friends and family to see what they're up to, and discover accounts from all over the world that are sharing things you love. Join the community of over 500 million people and express yourself by sharing all the moments of your day—the highlights and everything in between, too. Use Instagram to: • **Post photos and videos, edit them with filters and creative tools, and combine multiple clips into one video.** • Share multiple photos and videos (as many as you want!) to your story. Bring them to life with text and drawing tools. They disappear after 24 hours and appear on your profile grid or in feed. • Watch stories from the people you follow in a bar at the top of your feed. View them at your own pace. • Discover photos and videos you might like and follow new accounts in the Explore tab. • Send private messages, photos, videos and posts from your feed directly to friends with Instagram Direct. • Instantly share your posts to Facebook, Twitter, Tumblr and other social networks.

2. Related work

In previous researches on domain analysis, the methods usually need to be done manually, such as Feature-Oriented Domain Analysis(FODA)[5] , Feature-Oriented Reuse method (FORM)[6] , and Organization Domain Modeling (ODM)[9] . Despite some tools[10] [11] [12] that have been proposed to support these methods, most of them can only help analysts to manage the process of feature extraction, whereas feature extraction itself and establishment of constrains still rely on intensive human interaction. This makes these methods only adapt to analyze relatively low volume of data.

With the abundance of data resources in recent years, the limitation of methods mainly depending on human analysis is obvious. Thus, technologies in data science, such as LDA, Clustering, N-gram, SVM, have been introduced to domain analysis to support the researches on automatic or semi-automatic methods. Based on the types of texts introduced in Section 1, we classify research works related to our work into three categories.

1) To support the development process, the first type of texts is expressed normatively and records complete information of software, so we define it as the inputs with high quality. Many researches focus on utilizing this type of texts to construct Feature Models (FM) for Software Product Line Engineering (SPLE) [4] . Some of these researches take the formal texts as the input. For example, from tabular data files, Acher et al. propose a semi-automated method to extract FMs through a dedicated language and a specific merging algorithm[13] . They further propose a tool-supported approach to extract and manage the evolution of software variability[14] . Additionally, other researches analyze this type of texts expressed in natural language and extract features by introducing data analyzing technologies. From text-based software requirements specifications (SRSs), Mu et al. extract functional requirements by analyzing the linguistic characterization of SRSs [15] ; Bagheri et al. propose an approach that employs natural language processing techniques to identify potential features and integrity constraints in the domain document for support domain engineering lifecycle[16] ; in [17] , Niu Nan propose a systems-oriented approach based on information retrieval (IR) techniques to extract functional requirements profiles automatically; based on natural language semantics analyzing and classification technology, Mefteh et al. propose a fully top-down method to extract potential features and group similar ones for structuring the FM [18] ; applying a sequence of machine learning steps, Rahimi M et al. present a data mining approach for extracting and modeling quality concerns from quality related requirements to generate goal graphs[19] ; Nili Itzik et al. proposing SOVA, which uses ontological and natural language semantic considerations to

generate feature diagrams for modelling variability in SPLE[20] . Although these approaches can achieve fairly good results, the limitation of inputs restraints their effectiveness. For the reason of secrecy and copyright, this kind of inputs is always limited inside of companies, so it is hard to be collected for the developers when they come to a new domain. Different from these researches, we analyze app descriptions that widely exist in various app marketplaces and data collection is no longer a problem.

2) Facing to users, the second type of texts is usually from web-sites and has a huge volume. Limited by the length, this type of texts cannot contain as complete and accurate information as the first type, whereas it only contains the most important features of the products to attract users. Thus, it is taken as the inputs with relatively good quality and the researches on it mainly center on feature extraction. Yue Yu et al. propose an approach to mine features from web repositories and organize these features by constructing a HESA[21] ; based on contrastive analysis, Alessio Ferrari et al. propose an approach to identify commonalities and variabilities from the brochures of a group of vendors[22] ; Negar Hariri et al. use incremental diffusive algorithm to discover common features from online product descriptions, and give a quantitative feature recommendation algorithms for domain analysis process[8] . Considering that feature is also a form to express software requirements, Lian et al. propose MaRK to identify and retrieve requirement knowledge from the documents that containing descriptions of functional features [23] . There are also researches on constructing FM based on this kind of texts, but they generally overlook the relationship between features. For example, the approach given by Jean-Marc Davril et al. can construct FMs from publicly available product descriptions, but it does not include exclusion clauses[24] . Although these methods can effectively utilize the information in the texts, there usually exists some constraints to their inputs, such as feature descriptors, which are not contained in the app descriptions. Thus, these methods may not achieve the same good results when analyzing app descriptions. Besides, we find that the colloquial and incomplete nature of app descriptions are even more serious, so it needs more effort for feature extraction.

3) The third type of texts mainly contain the information of user experiences, and it tends to be short, large volume, and noisy-nature, so we take it as the inputs with low quality. Many automatic methods are proposed to obtain effective information from the reviews or comments. The work by Claudia Iacob et al. define 237 linguistic rules manually and design MARA to extract feature requests from reviews[25] [26] ; while Phong Minh Vu et al. propose keyword-based/ phrase-based approaches to acquire user opinions[27] [28] . In addition, Emitza Guzman et al. emphasize the analysis of relationship between app features and user sentiments and present an approach for understanding app reviews[29] ; and Lorenzo Villarroel et al. introduce CLAP to prioritize the clusters of reviews for supporting released planning of apps[30] . These methods aim at supporting the modification of app existing features or identification of app evolution direction, while reviews also can be used in other aspects. For example, Noor Hasrina Bakara et al. propose a semi-automated approach FENL to extract features for initiating the requirements reuse process[31] . These researches can analyze and use reviews efficiently. However, most of them only focus on the extraction of features, without analyzing the relationship between features. Hence these approaches are restricted in domain analysis process.

From the related works presented in this section, it can be seen that the approaches of domain analysis focus on different type inputs. From this perspective, the natural difference between our work and these researches is that we focus on analyzing app descriptions. Specifically, in order to better support the process of domain analysis, we extract not only features but also relationship from app descriptions based on their characteristics to construct a model. Moreover, to the best of our knowledge,

there is no research describing this problem and the analyzing process in a formal way.

3. Problem statement and the overview of our approach

Many researches have defined the activities in the process of domain analysis[2] [32] [33] . We summarize them and divide the domain analysis process into three general stages, as shown in Fig.1: in the early stage (before time *a*), domain scope is identified and data is collected from different sources; in the middle stage (between time *a* and *b*), domain knowledge is obtained by data analysis, and it is refined and clustered; and in the last stage (after time *b*), the domain knowledge is evaluated by the developers to make further business decisions. It can be seen that the success of both the early stage and the last stage depends on developers' cognition of the product to be developed, so the activities in them need more manual effort; whereas the activities in the middle stage are the objective data handling, which can be done by data technologies. Thus, this research mainly focuses on completing the middle stage efficiently and accurately to reduce time-cost and improve the quality of domain analysis.

Considering the development of a new app, developers identify its domain scope and establish the dataset of related app products $D_{set} = \{d_1, \ldots, d_n\}$ before time *a*, where data d_i={text, A} is an app description text with a set of market attributes A={A_1,A_2…,A_m}, such as rating and downloads. At time *b*, the analysis of dataset D_{set} is completed, and the domain knowledge can be expressed as a three-tuple (F,NF,R), including functions related knowledge, non-functions related knowledge and relationship among them. As the result of the middle stage, (F,NF,R) provides the basis for the last stage of domain analysis. Based on the introduction above, we formalize our research question as "how to mine domain knowledge from D_{set} to construct (F,NF,R) automatically". In this paper，we propose an approach to solve this problem, as shown in Fig.2.

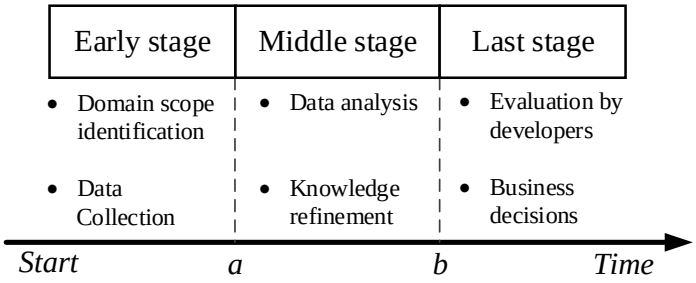

Fig.1. The process of domain analysis

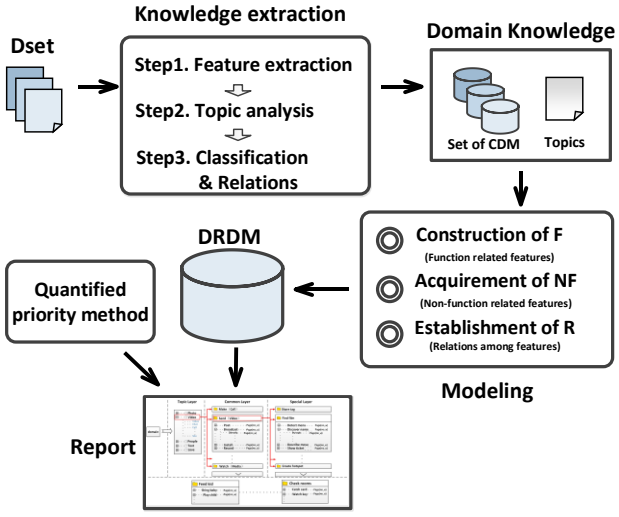

Fig.2. Overview of our approach

Firstly, a Concern-based Description Model (CDM) is defined to formally describe the domain knowledge in an element of D_{set}. We obtain and express the domain knowledge in D_{set} with a set of CDMs and Topics. Three sub-steps are conducted to complete this step: 1) feature extraction rules based on sentence syntax are acquired by reviewing and summarizing samples of data in D_{set}, so that for each $d_i \in D_{set}$, we can automatically extract function related and non-function related features from $d_i.text$ by utilizing these predefined rules and formalize them in the form of concerns; 2) furthermore, we change the process of LDA to make it more adaptive to our problem and use it to obtain the topics of domain knowledge in D_{set}; 3)based on these topics, we further classify the function related concerns into two types, according to whether or not the features commonly exist in app marketplaces, and the relationship among the concerns are gained to complete the modeling of each d_i.

Secondly, we analyze the data obtained above and define a Data-based Raw Domain Model (DRDM) to formalize the final domain knowledge (*F,NF,R*). In this step, we obtain and define different levels of knowledge in *F* by data classification, clustering and merging the function related concerns in CDMs; and *NF* is gained from non-function related concerns in CDMs; meanwhile, R is obtained by analyzing the relationship among different kinds of knowledge in F and NF. Moreover, in order to improve the usability of knowledge expressed by DRDM, we give a quantified method based on the attributes of each app $d_i.A$ in D_{set} to prioritize the knowledge in DRDM, so that a report is generated to express the results of our approach.

4. Automated modeling of app descriptions

A *concern* refers to a specific goal, concept or region of interest, corresponding to one or more problems or features that software developers care about. 'Separate of Concern' is a fundamental principle complied in software engineering, and the software development methods and modeling methods depending on it have been widely studied and applied. In our earlier work, we use concern as the basis to describe requirements and achieve good results[34] [35] [36] . Thus, we formalize a feature extracted from App descriptions by a concern, and the domain knowledge in one app description can be formally described by a CDM. This section gives the detailed definitions and the automatic modeling method.

4.1 Definitions of model

A CDM is formally defined as: CDM=(*App_Name,Cset,CRset*). Let *App_Name* be the unique identifier of the app product corresponding to CDM; and the set of concerns is defined as Cset; the set of relationship between concerns is CRset. The definition in each part is given as below.

Definition 4.1. Concern：each concern C is specified as a 4-tuple (*id*, *Des*, *type*,*Nums*),where

- *id* refers to the identifier of C;
- *Des* is the description of a feature expressed by C;
- *type* is the type of C in category theory;
- *Nums* is the set of sentence numbers that are relied by C.

A concern represents the information of a feature in the domain, and it can be classified into three types: common, which represents the feature related to the function that is usually contained by products in the domain; special, which describes the feature related to the function that is extraordinary

in products of the domain; property, which records the non-function related feature.

Definition 4.2. relationship between concerns: the relationship between two concerns is defined as a 4-tuple CR=(C_i,C_j,*sentN*,*kind*), where C_i and C_j are two concerns that have relationship in CDM; sentN is a set of sentence numbers, and it represents the sentences which are the basis of the relationship. So we have sentN=C_i. *Nums* ∩ C_j. *Nums*.

The relationship is categorized into four kinds: Common_of, Special_of, Common_Special and Property_of, where Common_of/Special_of is the relationship between two concerns with the same type, that is common or special； Common_Special is the relationship between common-concerns and special-concerns; Property_of is the relationship between common-concern or special-concern and property-concern.

It should be noted that, for the concerns whose types are properties, there is no relationship between them and there must exist relationship from common/special-concern to them. This is because property-concern represents the non-functional information related to the functions, such as attributes or constraints, thus it relies on other types of concerns.

Based on the above definitions, the automatic modeling method is introduced step by step.

4.2 Feature extraction based on syntax

Feature extraction is the central process of mining domain knowledge from app descriptions. In software related texts, features are expressed by the words with particular POS (Part of Speech), such as nouns, verbs, and/or adjectives[8] [31] . In app descriptions, features are always related to certain structures in a sentence. For example, the function related features are expressed by the verb with nouns, such as ‘post video’ and ‘share photos’ in Table 1, whereas the non-functional features are always given as a limitation structure. Thus, we can extract features by analyzing the structures of sentences and POS of words. Furthermore, since the syntax in app description tends to be simple for readability, we can identify the relationship between structures in sentences and features by analyzing the natural language semantic meaning of some app description texts manually, and construct feature extraction rules by summarizing them. Based on these rules, the knowledge of features can be mined from any data unit $d_i \in \mathrm{D_{set}}$. This process consists of the following three steps.

Firstly, we preprocess $d_i.text$. We remove the non-English or non-text parts from $d_i.text$ and split it into a set of sentences. Since our process of feature extraction depends on syntax analysis of sentence, the stop word removing and stemming are not contained in our preprocessing.

Secondly, syntax analysis is done by taking the sentence as a unit. We choose Stanford Parser, which is one of the state of art parsers in the general English domain, as the tool to transform a sentence to a parsing tree. For example, for the sentence ‘Post photos and videos, edit them with filters and creative tools, and combine multiple clips into one video’ shown in in Table 1, its parsing tree is given in Fig.3.

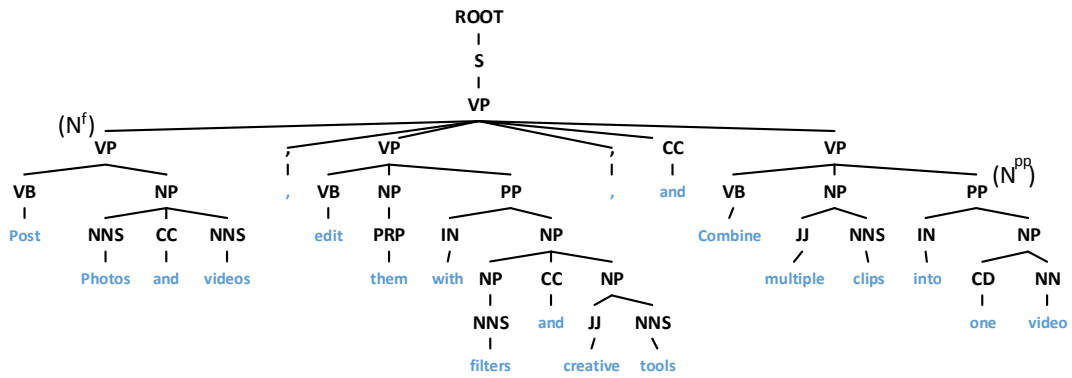

Fig.3. Parsing tree of a sentence

Finally, the features are extracted from the parsing tree and expressed by a set of concerns. This is the most important step in the process of feature extraction, and it relies upon the predefined extraction rules. The detailed information of this step is given in this sub-Section.

4.2.1 The acquirement of feature extraction rules

In order to acquire the feature extraction rules, we collect descriptions from 140 app products covering 7 mainstream classes in Google Play, and there are 2245 sentences in total. By analyzing some examples of these sentences, we summarize the set of rules, denoted by $R\text{set}$ ($R\text{set} = \emptyset$ initially).

The $R\text{set}$ is acquired in an iterative way. Specifically, we select a sentence *s* from a description text in order and transform it into a parsing tree *t* by Stanford Parser. Then we extract features from *t* based on the rules in $R\text{set}$, and the result is evaluated by domain analysts manually: if the results can reflect the features in *s* correctly, $R\text{set}$ is not changed and we proceed to the next sentence; otherwise, we analyze the syntax structures in *s* to gain a new rule *r* and add it into the $R\text{set}$. In this process, if there is a confliction between the new added rule *r* and existing rules in $R\text{set}$, we eliminate it by adding constrains or setting different priorities to the conflicted rules.

With the enrichment of $R\text{set}$, it is difficult to discover new rules. We use $R\text{set}$ on large-scale app descriptions to evaluate its completeness, and the results are also used to complete $R\text{set}$ constantly.

4.2.2 The basic definitions for formal descriptions of feature extraction rules

In order to describe feature extraction rules clearly, some basic definitions are given. In a parsing tree:

G(n) describes the grammar symbol in the node n. For example in the node N^f in Fig.3, G(N^f)=VP.

The set of grammar symbols $\text{Gset}(g) = \{g' | \text{string}(g') = \text{string}(g) :: \alpha\}$, where g, g' are grammar symbols in the non-leaf nodes of a parsing tree; α is a string of characters; symbol :: represents string connection. For example, $\text{Gset}(\text{VB}) = \{\text{VB}, \text{VBD}, \text{VBG}, \dots\}$.

Simple-tree is the tree composed of a node n and its sub-nodes, where n has only one level of sub-nodes.

The result of feature extraction process is described by $result = [\text{A}, \text{B}]$, where A,B are the set of words or phrases. For a *result*, it is further handled as follow:

- $\text{L}([\text{A}, \text{B}]) = \{[x, y] | \text{x} \in \text{A} \wedge \text{y} \in \text{B}\}$;
- $\text{L}([\text{A}, \text{B}]) = \{\text{xy} | \text{x} = \text{A} \wedge \text{y} = \text{B}\}$.

Then, a *result* is changed into phases, each of which describes a feature. Note that $[\text{A}] = [\text{A}, \emptyset]$.

Besides the above definitions, we also define functions to support the rules as follow:

In order to judge the structure of tree *t* which takes node N as the root, we construct two functions $f^{judge}(N, g_i, g_j)$ and $f^{simple}(N, g_i, g_j)$. Suppose the child nodes of N from left to right are ($\text{n}_1,\dots,\text{n}_k$), then $f^{judge}(t, g_i, g_j) = \{(\text{n}_l, \text{n}_{l+1}) | (l \in [1, k-1]) \wedge (\text{G}(\text{n}_l) = g_i) \wedge (\text{G}(\text{n}_{l+1}) = g_j)\}$. Similarly, suppose the roots of simple-tree in *t* are ($\text{n}_1,\dots,\text{n}_k$), if there exists $l, m \in [1, k], l < m, \text{G}(\text{n}_l) = g_i, \text{G}(\text{n}_m) = g_j$ and $\forall h \in (l, m), \text{G}(\text{n}_h) \neq g_i, \text{G}(\text{n}_h) \neq g_j, f^{simple}(t, g_i, g_j) = (\text{n}_l, \text{n}_m)$, otherwise, $f^{simple}(t, g_i, g_j) = \emptyset$.

Furthermore, we have two functions to gain words/phases from the phasing tree. For a Node N, if there exists a path ($\text{n}_1,\dots,\text{n}_k$) where n_1=N, $\forall l \in [1, k-2]), \text{G}(\text{n}_l) = g_i, \text{G}(\text{n}_{k-1}) = g_j,$ and n_k is the leaf, $f^{leaf}(N, g_i, g_j)$ returns the set of words in all the n_k that meet the conditions above. $f^{phase}(N)$

returns a phase in the tree t, which takes node N as the root, thus the words in all the leaves of t from left to right compose the phase, such as $f^{phase}(\mathrm{N^{pp}}) = \{(\text{into one video})\}$ in Fig.3.

4.2.3 The classification and use of feature extraction rules

Based on the definitions and functions introduced in 4.2.2, the rules in *R*set are described formally. As shown in Table 2, the final *R*set contains 24 rules, which are classified into 2 kinds with 4 sub-kinds and 3 priorities (1-3 from high priority to low priority). The details are given as follow.

The rules in the first classification R_1 aim at overall analyzing and transforming the parsing tree to improve the accuracy of feature extraction. Specially, there are two conditions to be considered:

1) Before feature extraction, the pronouns need to be replaced with their anaphors according to the structure of sentence. For example, the phrase 'edit them' shown in Fig.3 is meaningless itself until it is replaced by "edit photos and videos". In order to achieve this goal, we analyze the positions of pronouns and find the verbs near them. Based on this information, the relationship between pronouns and their corresponding nouns are obtained to generate the rules in R_1.1. Such as the rule 1 in Table 2, it can handle one of the simple but common conditions: for a tree *t* that takes node *n* as root, we analyze the simple-tree in *t* to judge if there are pronouns and reduce the pronouns to the noun right before it.
2) In a sentence of app description texts, there are always grammar structures that are useless or even misleading for feature extraction, such as the structure that represents the tense of sentences. Since such grammar structures have their fixed presentation forms, we summarize these forms to generate the rules in R_1.2 for eliminating them. For example, the rule 2 in Table 2 can judge whether the tree *t* taking node *n* as root has passive pattern (which is presented as a be-verb with a passive verb) and deletes the be-verb if necessary.

After handling the parsing tree based on R_1, the rules in the second classification R_2 are utilized to extract the concrete features. Since the features can be classified into function and non-function related, the R_2 is also further divided into two sub-kinds.

1) For extracting function related features, R_2.1 identifies the function related structures in a tree and acquire the effective words/phrases in them. According to practical conditions, different rules are generated. Considering two most common cases, if the verb in a sentence is transitive, it and the noun phase after it together can construct a feature. The rule 3 in Table 2 can get such features, for example, phases 'post photos' and 'post videos' can be extracted from the sub-tree that takes $\mathrm{N^f}$ as the root; otherwise, if the verb is intransitive, there need to be a preposition after it connect (connecting it to) a noun to express a function. The rule 4 in Table 2 gets features in such condition, for example, the sentence 'you can search for broadcasts by location or topic' satisfies the above rule and yields the phase 'search broadcasts'.
2) Similarly, the rules in R_2.2 identify the non-function related parts to extract non-functional features. Our research shows that the non-functional features are always expressed as the limitation or supplement of the function features, so we get such structures based on the parsing tree. The rule 5 in Table 2 judges if a preposition phrase is the supplement of a verb object phrase and extract such information as a non-function feature, for example, the phase 'into one video' under the note $\mathrm{N^{pp}}$ is extracted by this rule.

Table 2 shows five typical rules, which can support most feature extraction tasks (about 60% to 70%). Other rules are expressed in the same formal way to cope with other special structures of sentences, such as object ahead of predicate, object ahead of verb, modifier behind central word.

Based on the rules in *Rset*, features can be extracted from the parsing tree of a sentence. A tree T is analyzed in a top-down traversal strategy. Firstly, the structures of *T* are matched with the rules in R_1 (Tree transform rules), so that *T* is transformed to a new tree *T'*; secondly, *T'* is further analyzed to extract concerns, and the results are divided into two conditions for the current node *N* in *T'*:

- If N satisfies the rules in R_2.1, a common/special-concern is generated to record the result. Note that the type of these concerns is distinguished in Sub-section 3.3.
- If N satisfies the rules in R_2.2, a property-concern is generated to express the result.

When the traversal of tree *T'* is completed, a set of concerns is gained to record all the features in the sentence. In this set, the *Nums* of any one concern *C* is the number of this sentence. For example, from the parsing tree in Fig.3, we can get a set of concerns {(Instagram_12, post photos, common/special, {4}), (Instagram_13, post videos, common/special, {4}), (Instagram_14, edit photos, common/special, {4}), (Instagram_15, edit videos, common/special, {4}), (Instagram_16, with filters and creative tools, property, {4}), (Instagram_17, combine clips, common/special, {4}), (Instagram_18, into one video, property, {4})}.

After all the sentences in an app description are analyzed in turn, the set of concerns *CDM.Cset* is established to describe all the features gained in this process. However, some important features may be mentioned in an app description for multiple times, and hence there are concerns in *CDM.Cset* that are generated from different sentences but express the same content. In order to delete such redundant information, we stem the words in *C.Des* to their root form and merge the similar concerns. Specifically, if all the words in C_i.*Des* appear in C_j.*Des*, we have $C_j.Nums = C_j.Nums \cup C_i.Nums$ and delete C_i.

Table 2 Description of feature extraction rules

Kinds	Sub-kinds	Feature extraction rules
R_1: Tree transform rules	R_1.1： Rules of pronouns reduction.	*(Rule 1)：* $\forall \text{node n}, ((f^{simple}(\text{n}, NN, PRP) = (\text{n}_l, \text{n}_m))) \Longrightarrow f^{leaf}(n_m, PRP, PRP)] = f^{leaf}(\text{n}_l, NN, NN)]$. (priority-3).
	R_1.2： Rules for judgement of sentence pattern.	*(Rule 2)* $\forall \text{node n}, ((G(\text{n}) = VP) \wedge \left((\text{n}_l, \text{n}_{l+1}) \in f^{judge}(\text{n}, VB, VP)\right) \wedge (f^{leaf}(\text{n}_l, VB, VB) \subseteq \text{be}_{\text{Set}})) \Longrightarrow \text{ignore}(\text{n}_l)$, where $\text{be}_{\text{Set}} = \{\text{be}, \text{am}, \text{is}, \text{are}, \text{was}, \text{were}\}$. (priority-1).
R_2: Information extraction rules.	R_2.1： Rules for extracting of function related features.	*(Rule 3)* $\forall \text{node n}, \left((G(\text{n}) = VP) \wedge \left((\text{n}_l, \text{n}_{l+1}) \in f^{judge}(t, VB, NP)\right)\right) \Longrightarrow [f^{leaf}(\text{n}_l, VB, VB), f^{leaf}(\text{n}_{l+1}, NP, NN)]$. (priority-1).
		(Rule 4) $\forall \text{node } t, \begin{pmatrix} (G(t) = VP) \wedge \left(\exists(\text{n}_l, \text{n}_{l+1}) \in f^{judge}(t, VB, PP)\right) \wedge \\ \left(\exists(\text{n}_k, \text{n}_{k+1}) \in f^{judge}(n_{l+1}, IN, NP)\right) \end{pmatrix} \Longrightarrow f^{leaf}(\text{n}_l, VB, VB), f^{leaf}(\text{n}_{k+1}, NP, NN)$.

		(priority-2).
	R_2.2： Rules for extracting non-function related features.	*(Rule 5)* $\forall \text{node } t, \left(\begin{array}{c} (G(t) = VP) \wedge \left(\exists (\mathrm{n}_l, \mathrm{n}_{l+1}) \in f^{judge}(t, VB, NP) \right) \wedge \\ \left(\exists (\mathrm{n}_{l+1}, \mathrm{n}_{l+2}) \in f^{judge}(t, NP, PP) \right) \end{array} \right) \Rightarrow$ $\{[f^{phase}(\mathrm{n}_{l+2})]\}$. (priority-1).

4.3 The construction of CDM

In order to identify the type of concerns in *CDM.Cset*, we analyze the topics of $\mathrm{D_{set}}$, and distinguish type of common/special concerns based on the correlation between them and topics. Then the set of relationship between concerns *CRset* can be established to complete the construction of *CDM*.

4.3.1 Topic analysis of $\mathrm{D_{set}}$

We tried to use LDA to analyze the topics in $\mathrm{D_{set}}$. LDA relies on statistical models to discover the topics that occur in a collection of unlabeled text [37] , and it has been used to analyze text data in many researches. However, when we use LDA to analyze $\mathrm{D_{set}}$ directly, the meanings of gained topics are unclear. Table 3(a) shows a result of LDA in experiments (top-10 keywords are presented), the example is chosen because it gives similar topics as the final result we used in our case study (shown as Table 3(b)). It can be seen that the result contains some domain irrelative topics, for example 'Use(T1)' is a topic of words expressing activity; also the difference between T6 and T7 is not obvious and the meaning of T8 is unclear. These topics would affect the correctness of our domain model and hinder the developers understanding the domain knowledge.

To solve this problem and get better results, we give a method (denoted by O-LDA) based on basic LDA process by handling its input and output.

Firstly, we extract features from $\mathrm{D_{set}}$ by the approach given in Sub-section 3.2. And the verb parts of the features are deleted, because they are not helpful for describing the domain knowledge in a highly abstract way. The results are used as the input of LDA instead of using texts in $\mathrm{D_{set}}$ directly. This can be viewed as a data filtering process to improve information density of the input.

Secondly, the input is handled by a basic LDA process, and we get a topic set $\mathrm{Topic_{set}}$=$\{topic_1, topic_2, \ldots, topic_n\}$. In $\mathrm{Topic_{set}}$, each topic is described by a set of words, that is $topic_i = \{w_1, \ldots, w_m\}$, where each word *w* has a P(*w*/*topic*$_i$) to represent the presence probability of *w* appeared in *topic*$_i$.

Finally, further analysis is conducted to the above results. We only reserve the nouns for each topic to make it clearer to represent the related domain knowledge. In this process, for the word describing different topics, we reserve it in only one topic to increase the differences between topics. That is if $\exists w \in topic_i \cap topic_j (i \neq j) \wedge P(w/topic_i) > P(w/topic_j)$, delete w from $topic_j$.

We get the final set of topics $\mathrm{Topic_{set}}$ from $\mathrm{D_{set}}$ through the above process. Table 3(b) gives the result of topic modeling in our case study. It divides the domain into five parts, and the keywords of topics are meaningful and clear for describing the domain knowledge.

Note that in practice, the final result may be identified with the help of app developers if necessary, so as to improve the quality of O-LDA.

Table 3 Topics on 'social' apps and top-1 word is used as the name of topic.

(a) The results of basic LDA (K=8)

Use(T1)	video(T2)	People(T3)	Text(T4)	love(T5)	Chat(T6)	Chat(T7)	Input (T8)
Use	Video	**People**	Text	Love	**Chat**	**Chat**	Input
Chat	**Chat**	Like	**People**	Woman	Photo	**People**	Support
Find	**Use**	Follow	Message	Girl	**Use**	Love	Emoticon
Share	Call	**Use**	Email	**Use**	Picture	Use	Free
Live	find	**Chat**	**Use**	Data	Like	Call	Share
Meet	Voice	Network	Emoticon	Phone	Love	Date	Smiley
Call	Convers	Find	Smiley	**People**	Share	Status	**People**
Like	Media	Person	Language	Message	Image	World	Message
Get	Application	Profile	Time	Locate	Friend	Live	Like
Free	People	Game	Status	Fun	**People**	Single	Number

(b) The results of O-LDA(K=5).

Photo(T1)	video(T2)	People(T3)	Text(T4)	love(T5)
Photo	video	people	Text	love
picture	chat	follow	message	woman
image	call	network	email	girl
world	voice	person	emoticon	data
group	convers	profile	smiley	phone
moment	media	game	language	number
event	application	share	time	locate
life	inform	birthday	status	fun
family	tag	experience	update	caller
parent	site	Date	flower	idea

4.3.2 Classify the concerns and establish relationship

The $\text{Topic}_{\text{set}}$ represents the main information that app developers concern about in a particular domain. We define the type of concerns that are directly related to $\text{Topic}_{\text{set}}$ as common. By calculating correlation degree between a common/special type concern and topic, we can identify the concrete type of concerns.

Definition 4.3 Correlation degree: For a concern $C \in CDM.Cset$ and a $topic \in Topic_{set}$, suppose that the set of words in *C.Des* is $\{w_1,\ldots,w_n\}$ and *topic*=$\{w_l,\ldots,w_k\}$, the correlation degree between *C* and *topic* is the maximum value of words similarity in two sets:

$$\text{Cor}(C, topic) = \text{Max}_{i=1,j=l}^{i=n,j=k} sim(w_i, w_j),$$

where $sim(w_i, w_j) \in [0,1]$, which is the similarity between w_i and w_j and calculated based on WordNet[38] .

Using the formula above, the type of concerns in *CDM.Cset* can be identified through setting a threshold θ (θ=0.9 in our case study). $\forall \text{C} \in \text{CDM. Cset} \wedge \text{C. type} \neq \text{property}$,

$$\text{C. type} = \begin{cases} Common & \exists t \in T \wedge Cor(C,t) > \theta, \\ Special & else. \end{cases}$$

The *CDM.Cset* is obtained when the types of all concerns are determined. And each common-type concern in D_{set} can be assigned to a topic according to the highest value of correlation degree, so we can obtain a set of concerns for each *topic t* in $\text{Topic}_{\text{set}}$, denoted by $\text{C}^t set$, which is the preparation for

the subsequent analysis.

The relationship between concerns can be established. $\forall C_i, C_j \in CDM.Cset, if\ C_i.Nums \cap C_j.Nums \neq \emptyset$, there is relationship CR between C_i and C_j, we have:

- If $C_i.type = Common \wedge C_j.type = Common, CR = (C_i, C_j, C_i.Nums \cap C_j.Nums, Common\_of)$;
- If $C_i.type = Common \wedge C_j.type = Special, CR = (C_i, C_j, C_i.Nums \cap C_j.Nums, Common\_Special)$;
- If $C_i.type = Speical \wedge C_j.type = Special, CR = (C_i, C_j, C_i.Nums \cap C_j.Nums, Special\_of)$;
- If $(C_i.type = Common \vee C_i.type = Special) \wedge C_j.type = property$,

$$CR = (C_i, C_j, C_i.Nums \cap C_j.Nums, Property\_of).$$

Till now, the set of relationship *CDM.CRset* is generated, and the construction of CDM is completed. Since each step in this process can be conducted automatically, the transformation from an app description text to a model CDM is automatic. After modeling all the app descriptions in $\mathrm{D_{set}}$, we can obtain a new dataset D'_{set}, $D'_{set} = \{d'_1, \dots, d'_n\}$,where $d'_i = (CDM_i, di.Attribute)$. Also, the $\mathrm{Topic_{set}}$ is also reserved to support the subsequent analysis of D'_{set}.

5. Modeling and Prioritizing domain knowledge

To deeply mine and organize the domain knowledge from $\mathrm{D'set}$, the approach of data integration is given in this section, and the results are used to construct a Data-based Raw Domain Model (DRDM). Moreover, we give a quantified method to prioritize the information in DRDM, and a report of DRDM is generated based on the priorities to help developers use the model and get key domain knowledge quickly.

5.1 Construction of DRDM

The central idea of DRDM constructing is to group the similar concerns. Before introducing the detailed process, we give the structure of DRDM and related definitions. For the purpose of clarity, we use CDM^{set} to represent the set of all models in $\mathrm{D'set}$, that is $CDM^{set} = \{d'.CDM | d' \in \mathrm{D'set}\}$.

A DRDM is the representation of knowledge to a particular domain, it classifies the domain knowledge into three types and can be specified as a 3-tuple (F,NF,R). The detailed illustrations for each part are given.

F is the representation of function related domain knowledge, and $\mathrm{F} = \{\mathrm{T}, \mathrm{Cc^T set}, Cc^S set\}$, where

- $\mathrm{T} = \{t_1, \dots, t_n\}$ is the set of domain topics;
- $\mathrm{Cc^T set} = \{Cc^{t_1} set, \dots, Cc^{t_n} set\}$, where $Cc^{t_i} set = \{Cc_1^{t_i}, \dots, Cc_n^{t_i}\}$, and $Cc_j^{t_i}$ is a cluster of concerns and represents a kind of features that is related to topic t_i in T;
- $Cc^S set = \{Cc_1^S, \dots, Cc_n^S\}$, where Cc_i^S is a cluster of concerns and represents a kind of features that are not ordinary in the domain;

In F, T is the $\mathrm{Topic_{set}}$ obtained in section 3.3.1, while $Cc^T set$ and $Cc^S set$ are the results of grouping common-concerns and special-concerns separately.

NF is the representation of non-functional domain knowledge, which includes the features related to non-functional attributes or constraints of app products in the descriptions. We define NF as the set of property-concerns, that is, $\mathrm{NF} = \{\mathrm{C} | \mathrm{C} \in \mathrm{CDM.Cset} \wedge \mathrm{C.type} = \mathrm{property} \wedge \mathrm{CDM} \in \mathrm{CDM}^{set}\}$. Because the presentations of such non-functional features are diverse, we do not cluster this kind of information to avoid misguiding developers.

R is the set of relationship between domain knowledge and contains two parts:

- $\mathrm{R_{f-f}(A, B)} = (\mathrm{dgree, rel})$, where $\mathrm{A} \in Cc^{t_i} set$ and $\mathrm{B} \in Cc^S set$, and $Cc^{t_i} set$ is an arbitrary element in $\mathrm{Cc^T set}$. $\mathrm{R_{f-f}(A, B)}$ reflects correlation degree and inner links between knowledge A

and B. For a $\mathrm{R_{f-f}(A,B)}$, its $\mathrm{rel}=\{\mathrm{CR}|\mathrm{CR}=(C,C',\mathrm{sentN},\mathrm{Common\_Special})\wedge C\in \mathrm{A}\wedge C'\in \mathrm{B}\wedge \mathrm{CR}\in \mathrm{CDM}\wedge \mathrm{CDM}\in \mathrm{CDM^{set}}\}$, and its *degree* has two values: if *degree*='High', it indicates that the relationship between A and B is strong; otherwise, *degree*='Low' shows that such relationship is weak.

- $\mathrm{R_{n-f}}=\{\mathrm{CR}|\mathrm{CR.type}=\mathrm{Property\_of}\wedge \mathrm{CR}\in \mathrm{CDM.RCset}\wedge \mathrm{CDM}\in \mathrm{CDM}^{set}\}$, it reflects the correlation between functional and non-functional domain knowledge, and it is the set of all the Property_of type relationship in elements of CDM^{set}.

In order to define the process of constructing DRDM clearly, we give the following expression rules. According to the process of feature extraction introduced in Section 4.2, the descriptions of a common or special concern *C* include two parts: verb part and noun part. We use $C.Des^{verb}$ and $C.Des^{noun}$ to denote them respectively, and take them as attributes of such concern *C*.

Definition 5.1 A_{-p}**:** Suppose that *A* is a set and P={p_1,…,p_n},where p_i is one attribute of the elements in A, A_{-p_i} represents a sub-set of A and is constructed by dividing the elements in A based on p_i.

Note that: $\bigcup_{i=1}^{i=n} A_{-p_i}=A$ and $A_{-p_i}\cap A_{-p_j}=\emptyset\ (i\neq j)$. A_{-p} can be used in an iterative way, that is $A_{-p_i-q_j}$ donates a sub-set of A_{-p_i} constructed by further dividing A_{-p_i} according to q_j.

Based on the illustrations above, the specific process of DRDM construction is given next.

Construction of F As some concerns generated from different app descriptions but may express the same content in $\mathrm{CDM^{set}}$, we eliminate such redundant information in $\mathrm{CDM^{set}}$ by preprocessing: for any two concerns C_i and C_j in the elements of $\mathrm{CDM^{set}}$, if all the words in C_i.*Des* appear in C_j.*Des* and C_i.*type*= C_j.*type*, C_i and C_j are merged: $C_j.id=C_j.id\cup C_i.id, C_j.Nums=C_j.Nums\cup C_i.Nums$; simultaneously, the relationship with C_i are delivered to C_j, that is if $\mathrm{CR}=(C_i,C_l,\mathrm{sentN},\mathrm{type})$ in CDM.RCset, new relationship $\mathrm{CR'}=(C_j,C_l,\mathrm{sentN},\mathrm{type})$ is generated. Then C_i and CR are deleted. After the preprocessing, we conduct data integration to common-concerns and special-concerns respectively.

Construction of $\mathrm{C}c^T set$ ***in F*** According to the classification method of concerns introduced in Section 4.3.2, a common-concern must relate to a special topic *t* in $\mathrm{Topic_{set}}$. Thus, for an arbitrary topic $t\in \mathrm{Topic_{set}}$, its related concerns are obtained to establish a set $\mathrm{C}^t set=\{C_1,\dots,C_n\}$. Here, we take part of $\mathrm{C}^t set$ related to the topic Photo(T1) shown in Table 3(b) as a simple example, as shown in Fig.4 (Note that only the descriptions of concerns are presented). Initially, there are 8 concerns related to T1 in $\mathrm{C}^t set$, that is $\mathrm{C}^t set$={Take photo, Take picture, Select picture, choose photo, share photo, send picture, download picture, tag photo}.The data integration takes $\mathrm{C}^t set$ as the unit and the process is divided into three steps: classification, clustering and merging.

Firstly, $\mathrm{C}^t set$ is classified according to the $C.Des^{noun}$ of concerns to obtain its sub-set $\mathrm{C}^t set_{-noun}$. Topic *t* is expressed by a set of keywords, that is *t*={*keyword*$_1$,…,*keyword*$_k$}. For a concern C in $\mathrm{C}^t set$, we classify it to the *keyword*$_i$ which has the maximum correlation degree with $C.Des^{noun}$. In this way, the $\mathrm{C}^t set$ is divided into *k* classes, and we define $\mathrm{C}^t set=\{\mathrm{C}^t set_{-noun_1},\dots,\mathrm{C}^t set_{-noun_k}\}$, where $\mathrm{C}^t set_{-noun_i}$ is the set of concerns related to the *keyword*$_i$ in *t*. In Fig.4, $\mathrm{C}^t set$ is classified into two sub-set $\mathrm{C}^t set_{-photo}$ and $\mathrm{C}^t set_{-picture}$ based on the two keywords 'photo' and 'picture' in the topic T1.

Secondly, for each $\mathrm{C}^t set_{-noun}\in \mathrm{C}^t set$, we use hierarchical clustering method[39] to group its concerns according to the $C.Des^{verb}$. In the process of clustering, the distance between any two concerns C_i and C_j is calculated by $\mathrm{Cor}(C_i.Des^{verb},C_j.Des^{verb})$, and the number of clusters is adjusted by setting the threshold β of distance among concerns in one cluster, that is for an arbitrary

concern C in a result of clustering $C^t set_{-noun-verb}$, it satisfies $\frac{\sum_{c' \in C^t set_{-noun-verb}} Cor(C.Des^{verb}, c'.Des^{verb})}{|C^t set_{-noun-verb}|} > \beta$ (β=0.6 in our case study). We handle all the elements in $C^t set_{-noun}$ in the above way. Based on the results, $C^t set$ is further divided, and we have $C^t set = \{C^t set_{-noun-verb_1}, \dots, C^t set_{-noun-verb_l}\}$, where $C^t set_{-noun-verb}$ is a cluster of concerns. After this step, $C^t set_{-photo}$ in Fig.4 is divided into $C^t set_{-photo-verb1}$ and $C^t set_{-photo-verb2}$. And $C^t set_{-picture}$ is divided similarly. This step is not obvious in Fig.4 because the volume of data in the example is too small.

Finally, we merge the elements in $C^t set$ based on the relationship between concerns. In $C^t set$, if the concerns in any two elements have the relationship of Common_of, there exists relationship between the two elements. And when the percentage that such relationship occupy in any element is higher than the threshold δ, it indicates that the relationship between these two elements is close and we merge them together. Specifically, for any $C^t set_{-noun-verb_i}, C^t set_{-noun-verb_j} \in C^t set$, a set $Cset$ is generated, $Cset = \{(C, C') | (C \in C^t set_{-noun-verb_i}) \wedge (C' \in C^t set_{-noun-verb_j}) \wedge ((C, C', \text{sentN}, \text{Common\_of}) \in \text{CDM.RCset}) \wedge (\text{CDM} \in CDM^{set})\}$, if $\text{MAX}\left(\frac{|Cset|}{|C^t set_{-noun-verb_i}|}, \frac{|Cset|}{|C^t set_{-noun-verb_j}|}\right) > \delta$ (δ=0.3 in our case), $C^t set_{-noun-verb_i}$ and $C^t set_{-noun-verb_j}$ are merged into one cluster. By merging all the elements that meet the above conditions in $C^t set$, the data integration of $C^t set$ is completed. A new set $Cc^t set = \{Cc_1^t, \dots, Cc_n^t\}$ is constructed to record the results. In the example shown in Fig.4, $C^t set_{-photo-verb1}$ and $C^t set_{-picture-verb1}$ are merged together to form Cc_1^t, while $C^t set_{-photo-verb2}$ and $C^t set_{-picture-verb2}$ are merged together to form Cc_2^t in the final $Cc^t set$ based on the formula above. Thus the result of data integration of $C^t set$ is $Cc^t set = \{Cc_1^t, Cc_2^t\}$.

The essential idea of the above process is taking $C^t set$ as a whole and dividing it step by step. To improve the accuracy of the process, the classification step with a higher accuracy is conducted first, so that the clustering of concerns can be conducted with relatively fewer concerns.

By acquiring $Cc^t set$ for each topic *t* in $\text{Topic}_{\text{set}}$, the construction of $Cc^T set$ is completed.

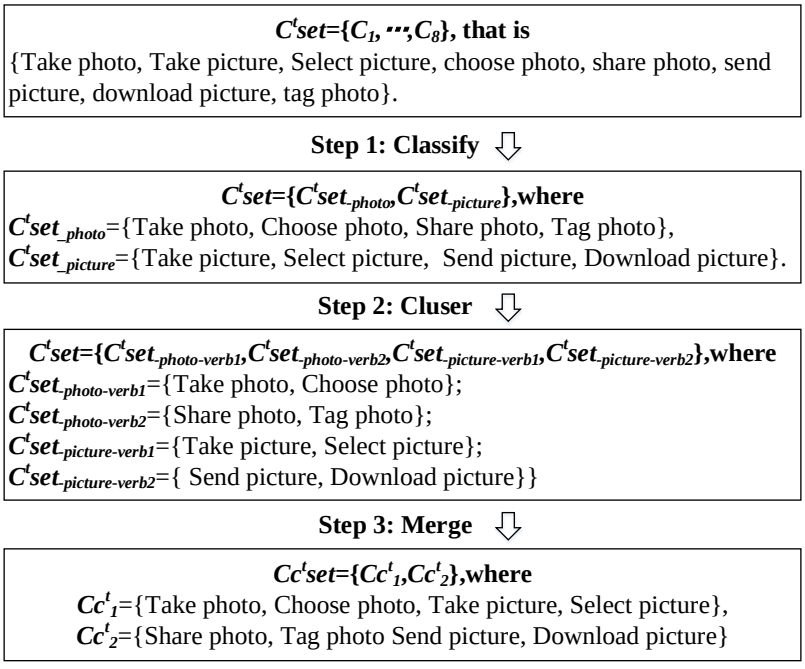

Fig.4. An example of the generation of $Cc^t set$.

Construction of $Cc^s set$ in F For the concerns belong to special type, since all the topics gained from O-LDA are not related to them directly, they can be viewed as all under another topic t′. However, different from common-concerns, there are no keywords in topic t′ for the classification step of specific-concerns. So we propose a modified classification method for specific-concerns.

Given the set of all the specific-concerns $C^s set = \{C_1, \dots, C_n\}$, the noun part of all its concerns generates a set $Des\_set = \{C_1.Des^{noun}, \dots, C_n.Des^{noun}\}$, where $C.Des^{noun}$ is a set of words. Let Wordset $= \bigcup_{i=1}^{i=n} C_i.Des^{noun} = \{word_1, .., word_m\}$, we sort the words in Wordset according to the frequency of the word appearing in Des_set. Then top-k words are obtained to form the Keyset $= \{\text{keyword}_1, \dots, \text{keyword}_k\}$(k=10 in our case), where the similar degree between any two words in Keyset is smaller than the threshold μ (μ=0.5 in our case). Keyset is taken as the set of keywords of topic t′. The concerns in $C^s set$ can be classified according to these keywords: for a concern C, if the correlation degree between $C.Des^{noun}$ and *keyword* in Keyset is higher than the threshold η(η=μ=0.5 in our case), it is classified to the most related keyword; otherwise, $C.Des^{noun}$ is added to Keyset as a new keyword. In this way, the concerns in $C^s set$ can be classified, and we have $C^s set = \{C^s set_{-noun_1}, \dots, C^s set_{-noun_m}\}(m \geq k)$. Based on this condition, the subsequent process of data integration is the same as common-concerns, and we get the final results of data integration is $Cc^s set = \{Cc_1^s, \dots, Cc_n^s\}$.

Construction of R After obtaining $Cc^T set$ and $Cc^s set$, the construction of DRDM.F is completed. Next, $R_{f\text{-}f}$ in DRDM.R can be gained by analyzing the relationship between the elements in each $Cc^t set$ of $Cc^T set$ and the elements in $Cc^s set$. For $Cc_i^t \in Cc^t set$ and $Cc_j^s \in Cc^s set$, suppose that $Cc_i^t = \{C_1, \dots, C_n\}$ and $Cc_j^s = \{C_1', \dots, C_m'\}$, we can generate a set of relationship rel $= \{\text{RC} | C \in Cc_i^t \wedge C' \in Cc_j^s \wedge \text{RC} = (C, C', sentN, Common\_Special) \wedge \text{RC} \in CDM.RCset \wedge CDM \in CDM^{set}\}$. If rel $\neq \emptyset$, generate relationship $R_{f-f}(Cc_j^{t_i}, Cc_k^s) = (dgree, rel)$, where

$$\text{dregee} = \begin{cases} High & \text{MAX}\left(\frac{|\text{rset}|}{|Cc_i^t|}, \frac{|\text{rset}|}{|Cc_j^s|}\right) > \delta, \ (\delta = 0.3 \text{ in our case}). \\ Low & otherwise. \end{cases}$$

To illustrate DRDM clearly, we give its structure shown in Fig.5. For each topic t_i, there must exist a related $Cc^{t_i} set$; and the relationship between common-concerns and special-concerns are the basis of the relationship between the clusters containing them respectively; additionally, the concerns in NF must relate to a concern related to functional feature in F. Such structure can support the generation of a report (details are given in 4.3) to help developers retrieve and understand the domain knowledge in DRDM.

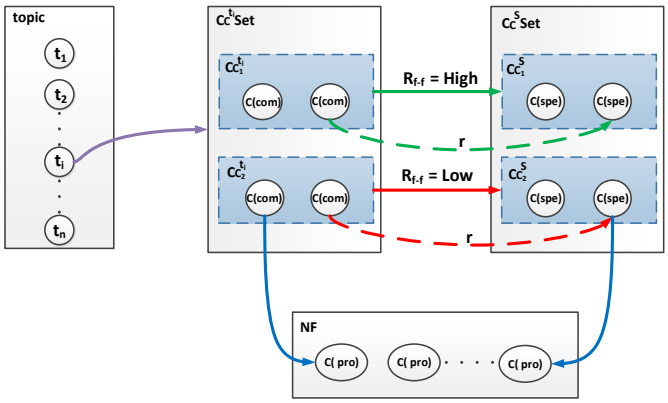

Fig.5. Structure of DRDM

5.2 Determine the priority of information

In data of D'set, there is a set of attributes obtained from app marketplace (introduced in Section 3). These attributes not only indicate the market value of app products, but also reflect the value of features contained in the apps. Thus, the priority of information is determined based on these attributes.

In D'set, we collect the attributes related to the market value of apps to generate the set $\mathrm{Attibute} = \{\mathrm{A}_1, \ldots, \mathrm{A}_\mathrm{n}\}$, then the value of CDM established from a description can be quantified by the formula generally expressed as follow:

$$\mathrm{V(CDM)} = f_v\big(g_1(A_1), \ldots, g_n(A_n)\big),$$

where function $f_v()$ is generated based on the relationship between the attributes; $g_i(A_i)$ is a transformation function for $\mathrm{A_i}$ to adjust its change tendency. All these functions balance the range of various attributes.

In this paper, we set Attribute=(Rating, Downloads). Rating and Downloads of app products are widespread attributes existing in almost every app marketplace, and they can indicate the market value of an app directly. Thus, we give the concrete function expressions in the formula of V(CDM) based on them.

Firstly, the rating ranges from 1 to 5, and we formalize the Rating in V(CDM). In other words, the transformation function for Rating is $g_r(Rating) = (Rate - 1)/4$. However, the range of Downloads can be from zero to thousands or even larger, so we design the following function to balance its value with Rating:

$$g_D(Downloads) = Cos\left(\frac{\pi}{2 \times (Downloads/K + 1)}\right),$$

where $g_D()$ utilizes the property of cosine function to complete the formalization of Download. More often an app has been downloaded, the closer its value is to 1. K is an integral number, which is used to adjust the change tendency of $g_D()$ for adapting different app domains. For a hot domain, K can be increased to discriminate the value of different apps with high downloads.

Based on the functions above, we define function $f_v()$ as:

$$f_v(Rating, Download) = w_1 g_r(Rating) + w_2 g_D(Downloads),$$

where w_i is the weight determined according to the focus of developers and $w_1 + w_2 = 1$, so the range of $f_v()$ is (0,1).

We can use function $f_v()$ to quantify the market value of an CDM in $\mathrm{CDM^{set}}$, and the value of V(CDM) can be delivered to the concerns in this CDM: for $C \in$CDM.Cset, the value of C (donated by *val*(*C*)) is proportional to V(CDM). Additionally, there are some concerns that they are mentioned repeatedly in the app descriptions, that is $|C.Nums| > 1$. This means the information in the concerns are highly emphasized by its developers, and their value should be increased correspondingly. Thus, we design the function to quantify the value of concerns as follow: $\forall C \in \mathrm{CDM.Cset}$,

$$\mathrm{val(C)} = \mathrm{V(CDM)} \times \left(1 + \frac{|C.Nums| - 1}{\sum_{C_i \in CDM.Cset} |C_i.Nums|}\right).$$

Utilizing the formula above, we can calculate the value of each concern in $\mathrm{CDM^{set}}$. This provides the basis for determining the priority of concerns in DRDM.

In DRDM, NF is the set of concerns and the value of its elements can be calculated by *val*(*C*) directly. Since the elements of $Cc^t set$ or $Cc^s set$ in DRDM.F are sets of concerns, we further propose the method to calculate their values. Suppose *e* is an element in $Cc^t set$ or $Cc^s set$, *e* is Cc_i^t or Cc_j^s, it can be expressed as $e = \{C_1, \ldots, C_n\}$ in a unified way. We analyze the key factors that affect the value of *e*(donated by V_e(e). |e| is the number of elements in *e*, and it reflects the frequency that

the information appears in original dataset, so V_e(e) is proportional to |e|. The average of element's value $E(e) = \frac{\sum_{i=1}^{i=n} val(C_i)}{|e|}$ indicates the general value of elements in e in the market and V_e(e) is also proportional to $E(e)$. The variance of element's value $\mathrm{D}(e) = \frac{1}{|e|}\sum_{i=1}^{i=n}(val(C_i) - E(e))^2$ can reflect the fluctuation of the values of elements in e, the larger $\mathrm{D}(e)$ is, the more intense the fluctuation of e is in the market and the smaller value of e to developers, so V_e(e) is inversely proportional to $\mathrm{D}(e)$. According to the analysis above, we calculate e by:

$$\mathrm{V\_e(e)} = f_v^e(\left(\frac{E(e)}{1 + D(e)}\right), g_e(|\mathrm{e}|)).$$

In function $f_v^e()$, there are two parameters $\left(\frac{E(e)}{1+D(e)}\right)$ and |e|. The two parameters can indicate the value of a data unit and frequency of data separately and we find that they are similar with Rating and Downloads in V(CDM) after Rating is changed to its formalization. So we define transformation function $g_e() = g_D()$. In addition, $f_v^e()$ is defined according to developers to balance between these parameters, and we define $f_v^e() = f_v()$. In this way, the formula that quantifies the value of CDM is reused to calculate $\mathrm{V\_e}(e)$.

Based on the quantified values, the information with same type in DRDM can be sorted and its priority can be determined.

5.3 The report of DRDM

In order to facilitate app developers to retrieve and read the information in DRDM, we generate a report based on the priorities. Fig.6 shows the report generated in our case study, and we use it to show the overall framework of our report. The top in Fig.6 is the main part of report and it includes three layers, each of which takes a kind of information in DRDM.F as the main subject and takes the information in DRDM.NF as complement to establish a catalog.

1) The 'Topic Layer' contains the information of F.T and its catalog is organized to two levels: the first level is the name of topics, and it can be unfolded to the second level which contains the keywords of each topic. This layer shown in Fig.6 is the five topics with their keywords given in Table 3(b).
2) The 'Common Layer' is the exhibition of F.$\mathrm{Cc^T}$set. As each element in $\mathrm{Cc^T}$set is related to a topic, the information in this layer is shown taking $Cc^{t_i}set$ as a unit. The 'Common Layer' in Fig.6. shows the information in $Cc^{t_2}set$ which is related to Video T(2). In a unit, the catalog is divided into three layers: the first level is the name of concern cluster $Cc_j^{t_i}$ in $Cc^{t_i}set$, gained by combining the most frequent verbs and nouns in the bag of words of $Cc_j^{t_i}$; the second level is the concerns in $Cc_j^{t_i}$; the third level is the property-concerns in DRDM.NF which has the Property_of relationship with the concerns in the second level, that is if there exists $(\mathrm{C_i}, \mathrm{C_j}, \mathrm{sentN}, \mathrm{Property\_of}) \in \mathrm{R.R}_{n-f}$, the C_i in the second level can be unfold to C_j in the third level. An example of these three levels from top to bottom is 'Send(Video)⟶Broadcast video⟶Directly' in Fig.6.
3) The 'Special Layer' is the exhibition of F.$\mathrm{Cc^s}$set, and its structure is similar as the 'Common Layer' except the third layer cannot be divided into units. In each layer, the developers can unfold the current information level by level as introduced above.

On top of that, developers can jump to the next layer by clicking the label of the first level and get

the related part of the next layer. Specially, by choosing the topic t_i in the 'Topic Layer', developer can obtain the information of $Cc_j^{t_i}$ in 'Common Layer'. Furthermore, when developers select $Cc_j^{t_i}$, the information Cc_k^s satisfying the condition that there exists $\mathrm{R_{f-f}}(Cc_j^{t_i}, Cc_k^s) = (\mathrm{High}, \mathrm{rel})$ in DRDM.R is shown in 'Special Layer'. For example, Cc_k^s 'Find film' is shown when developers select $Cc_j^{t_2}$'Send video' in Fig.6.

For an element Cc_k^s in $\mathrm{F}.Cc^s set$, if there is no relationship $\mathrm{R_{f-f}}(Cc_j^{t_i}, Cc_k^s)$ or $\mathrm{R_{f-f}}(Cc_j^{t_i}, Cc_k^s) = (\mathrm{Low}, \mathrm{rel})$ in DRDM.R, Cc_k^s cannot be linked to the main part of report (Such as 'Check room' in Fig.6). These elements together with its related information in DRDM.NF construct a supplement part of report. Its organization shown in the bottom of Fig.6 is the same as the third layer of main part of report.

In the report, information in each layer and each level is ranked based on its priority determined by our method (introduced in Section 5.2). The top-k ones are shown to developers each time to help developers discover the key knowledge they want, and the next top-k ones will be shown according to developers' needs. Besides, if some information cannot be well understood in the process of using report, the developers can get the related app descriptions in which the concrete sentences are highlighted by opening the Page(PID_id). In some particular situations, developers may want more detailed information related to a given common-concern. So it can jump to related specific-concerns for a common-concern in our report. That is for a certain common-concern C_i, if there exists $(\mathrm{C}_i, \mathrm{C_j}, \mathrm{Common\_Special}, \mathrm{sentN}) \in \mathrm{R_{f-f}}.rel$ in DRDM.R, developers can gain the information of C_j from C_i directly.

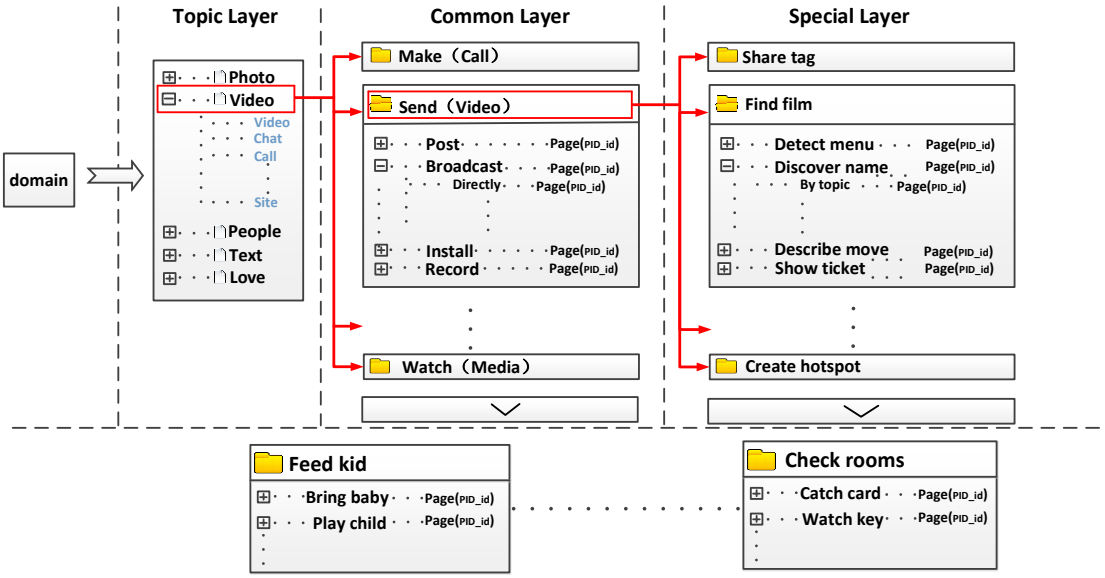

Fig.6. Report of DRDM

6. Evaluation and Results

To evaluate the effectiveness of proposed approach, we conducted experiments that primarily aimed at answering the following two questions:

Q1: What is the performance of each method used in constructing DRDM?

Q2: Whether DRDM is useful for supporting domain analysis?

Specifically, *Q1* focused on ensuring whether the proposed approach can gain and express the domain knowledge in app descriptions correctly. The goal of *Q2* was to analyze the usefulness of DRDM in helping developers make business decisions in the last stage of domain analysis, and it

focused on examining whether DRDM have a sufficiently high level of comprehensibility and usability.

This section reports the design and results of our experiments.

6.1 Participants

The experiments were conducted at the University of Jilin from the fall semester of 2016 to the spring semester of 2017. The participants included the following three parts.

1) 3 domain analysts who had more than five years' experience in the development of software system. They created the ground truth set used in the experiments and gave the final evaluation and suggestions from a professional view.

2) 10 graduates majoring in software engineering, especially domain analysis, modeling languages and techniques, data mining. They were the participants in the experiments that required professional knowledge. 3 of them had industrial experience in software engineering while others did not.

3) 60 Computer Science students, all of them had taken the undergraduate course on software engineering and had studied modeling and requirements engineering. They were the participants of surveys, and they were not expected to be familiar with feature modeling prior to the experiments.

6.2 Dataset

We had collected data from Google Play and created two datasets for our experiments.

Dataset_1 was a set of 120 app description texts, which were collected from six classes of apps ('Communication', 'Travel', 'Game', 'Photography', 'Sports', 'Education') in Google Play. It covered the products from popular mainstream to relatively professional domain. To make the data more typical, The 20 app products with different *Ratings* were selected from each class. Then, three domain analysts extracted the features from each text in Dataset_1 individually for establishing a truth set, in which the final results are determined by the 'majority rules'.

Dataset_2 was a set of data collected from 454 apps from the class of 'social', which included the apps for social activities in Google Play. The reasons that we select 'social' as the object are: firstly, there are a huge number of such apps in Google Play with diffident popular degrees, so it can be able to cover all kinds of situations better; and secondly, since these apps aim at common people, the knowledge in this domain is relatively more understandable, which is beneficial for conducting our surveys.

6.3 Experimental Design

The process of evaluation was divided into two parts for answering Q1 and Q2 respectively. The design of experiments are given as follow.

6.3.1 The experiments for Q1

Our approach constructs two kinds of models: CDM and DRDM. CDM is constructed based on two methods: feature extraction and topic modeling. DRDM is constructed by integrating the set of CDMs, which mainly uses feature clustering or feature grouping. Thus, we conducted experiments to evaluate the performance of each method, and the three experiments together can answer *Q1*.

Firstly, we constructed CDM for each description text in Dataset_1 automatically and compared the features extracted in this process with the truth set. Three evaluation parameters are used in experiments: *TP*, the number of features which are included in both CDM and truth set; *FP*, the number of features which are contained in CDM but not in truth set; *FN*, the number of features which are

contained in truth set but not in CDM. Then we evaluate the performance of our feature extraction method by calculating: Precision, Recall, and F-measure, which are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP};\ \ \text{Recall} = \frac{TP}{TP + FN};\ \ \text{F} - \text{measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Secondly, to study whether O-LDA is effective for our topic modeling task, we compared the outputs of O-LDA with the ones of the basic LDA. The topics of app description texts in each class of Dataset_1 and Dataset_2 were obtained by using O-LDA and the basic LDA separately. And 10 graduates were asked to evaluate each result from two aspects: whether the result is understandable; and whether the result is reasonable. The responses were reported by the participants on a scale of 1 to 5, where 1 indicates 'strongly disagree' and 5, 'strongly agree'. In this way, each result got its final score from 2 to 10. There were totally 10 paired samples for each class. Because they was not distributed normally as the number of samples is less than 30, we adopted the Wilcoxon signed ranks test to analyze them with the null hypothesis $H0_{effectiveness}$: there is no difference in effectiveness between the results of O-LDA and the basic LDA.

Thirdly, to evaluate our feature clustering method, we compare it with *K*-means, which is a common baseline used in many researches. Since the unsupervised measures such as cluster cohesion and cluster isolation evaluate the clustering method without reference to external information, they fail to evaluate how well the generated clusters support specific tasks [40] . In contrast, supervised measures evaluate the extent to which generated clusters match to an external ideal structure or ground truth (called answer set). Thus, we use supervised measures in our experiment. In the evaluation, clusters of features in each class of Dataset_1 were generated by our method and *K*-means separately, and the ideal clusters were produced by domain analysts manually. Then Purity is measured by comparing generated clusters to the answer set clusters: let the total number of features to be clustered be N, $W = \{w_1, \dots, w_n\}$ be the set of clusters found by the clustering algorithm and $CLU = \{clu_1, \dots, clu_m\}$ be the set of answer set, purity can be calculated as follow:

$$\text{Purity}(W, CLU) = \frac{1}{N}\sum_{k} \max_j \left| w_k \cap clu_j \right|.$$

The range of Purity is 0-1, and the closer it to 1, the better clustering algorithm is.

6.3.2 The experiments for Q2

Evaluating the usefulness of DRDM is a nontrivial task. We firstly analyzed a case study based on Dataset_2, on which we conducted two surveys for answering *Q2*. The first survey focused on evaluating the priority of concerns (describing features) in DRDM; and the second one focused on evaluating the comprehensibility and usability of the report of the DRDM generated in the case study.

The case study was conducted step by step according to our approach, so we do not explain its process here. The design of two surveys is introduced next.

(1) *The survey on the priority of domain knowledge in DRDM (Survey_1)*

It is difficult to evaluate the performance of our quantified method in determining the priority of domain knowledge directly, because even domain analysts cannot give us the right answers exactly. For example, it is hard to compare the market value of feature 'share message' with the feature 'share video'. Thus, we designed a survey to study the performance of our method indirectly.

Firstly, we selected the top-20 concerns from the same keyword in DRDM. These concerns satisfied two conditions: they were titles of concern clusters, so that they can describe a cluster of features, such

as ‘take photo’; they belonged to the same *keyword*, which could be the keyword of topics or the ones we gained in the process of data integration of special concerns (introduced in 5.1). These conditions were used to guarantee the concerns were comparable.

Secondly, for each concern, we used a sentence to describe its meaning. There were constraints: the sentence was modified based on the app descriptions in Dataset_2 to guarantee its authenticity; furthermore, the sentence was simple by eliminating the redundant information as much as possible to avoid that it may mislead the readers. For example, we find a sentence ‘you can edit selfies and photos so that you can add some effects and some great masks to them’ for feature ‘edit photo’, and we simplify it to ‘You can edit photos so that you can add some effects.’

Finally, we gave the concerns and their illustrations in a random order to 60 students, and asked them to judge ‘whether it describes a basic/important feature for the product in this domain’ for each concern separately, the answer can be selected from ‘Yes’, ‘No’, or ‘I do not understand this concern’.

The answers were collected and divided into two groups: one group was the answers of Top 10 concerns, the other was the answers of Top 11-20 concerns. We defined the percent of answer ‘Yes’ as the precision of a group. The survey included the concerns from 7 different keywords, 5 from topic and 2 from data integration of special concerns, to guarantee it could cover all kinds of conditions in DRDM. Because there were 60 paired of results for each keyword, we assumed that they was distributed normally and used the Student’s t-test to analyze them. The null hypothesis $H0_{priority}$ evaluated in the survey was: there are no difference in precision between two groups.

(2) *The survey on the usefulness of report (Survey_2)*

This survey aimed at evaluating the usefulness of DRDM by comparing it with raw data of app descriptions and the approach proposed in [8] .

The approach proposed in [8] mines feature descriptors from publicly available software repositories of product descriptions, then it analyzes the relationship between features and products, and generate a Feature Model (we donated it as FM_{-8}) to recommend features for supporting domain analysis of a specific project. Due to this approach is typical and has been widely confirmed, besides it uses inputs and has goals similar to those in our approach, we choose it as the compared object to evaluate DRDM. In our experiment, two models (DRDM and FM_{-8}) were generated automatically from Dataset_2 and used as the materials for the survey.

Based on the materials (raw app descriptions, DRDM, or FM_{-8}), we designed an online questionnaire to evaluate their usefulness. The questionnaire included three parts:

1) The first part was the questions about the notions in domain analysis. This part aimed at ensuring the participants have basic knowledge of domain analysis to finish the tasks in the survey. Only when the participants had given the right answers to these questions, they could proceed to the next part.
2) The second part was 10 questions on the domain knowledge of ‘social apps’. The participants needed to answer these questions with the help of the given material. For each question, we offered a multiple answer choice, and the participants could also give the open answer by listing the elements from the material. To prevent guessing, one option was given: ‘I don’t know’. At the end of this part, the participants were required to grade their agreement on the questions with two parameters: the difficulty in answering and the confidence in response.
3) The third part was an open task, asking the participants to make business decisions for a specific app. The participants needed to describe the app and gave the features of the app. Also, they were

required to classify the features into two types: which features are basic for the app; and which ones are creative for attracting users. Moreover, the participants needed to mark each feature with 'whether it comes from the material or from their own knowledge'.

In the survey, the 60 students were randomly divided into 3 groups, each of which was given different materials (raw app descriptions, DRDM, or FM_{-8}) for answering the questionnaire. Four variables were obtained from each questionnaire by the domain analysts.

- Score is the evaluation of the answers in the questionnaire given by the domain analysts. Its range is 0-100, in which the second and third part of questionnaire occupy 50 separately.
- Time to complete the questionnaire was recorded.
- Difficulty in answering and confidence in response were reported by the participants on a scale of 1 to 5.

Because the number of samples is less than 30, we did not assume that the data was distributed normally and adopted the Mann-Whitney U test for analyzing the data collected in this survey. Two null hypotheses are given in Table 4.

Table 4 Null hypotheses for evaluating the usefulness of DRDM

Null hypotheses	Independent variables	Dependent variables
$H0^1_{score}$/ $H0^1_{time}$ / $H0^1_{diff}$ / $H0^1_{con}$: There is no difference, in terms of score/time/difficulty/confidence, between responses answered using raw app descriptions and those answered using DRDM.	Materials: Raw app descriptions DRDM	Score Time Difficulty Confidence
$H0^2_{score}$/ $H0^2_{time}$ / $H0^2_{diff}$ / $H0^2_{con}$: There is no difference, in terms of score/time/difficulty/confidence, between responses answered using DRDM and those answered using FM_{-21}.	Materials: DRDM FM_{-8}.	Score Time Difficulty Confidence

Moreover, we used two-tailed statistical tests for all comparisons (whether when comparing raw app descriptions and DRDM or comparing FM_{-8} and DRDM) due to the non-directionality of the hypotheses.

Note that in all the statistical tests, we accepted a probability of 5% as the threshold to reject the null hypothesis.

6.4 Results and Analysis for Q1

To support the automation of our approach, we develop a tool based on Python. The tool contains three parts: the first part transforms app description to concerns, using *Stanford Parser* to obtain parsing trees of sentences and extracting features from the tree to generate concerns based on predefined abstract rules (introduced in Section 4.2); the second part implements O-LDA to obtain topics, and constructs CDM for each description by classifying concerns and establishing relationship between them; the third part groups the features to construct DRDM. With the help of this tool, we conducted the experiments for Q1 (introduced in 6.3.1), and the results for each experiment are analyzed next.

6.4.1 Evaluation of our feature extraction method

Since the features are expressed in natural language, it is difficult to evaluate the features extracted by our method with the truth set automatically. Thus, the comparisons were conducted by three domain analysts manually in the experiment.

Table 5 summarizes the results of experiments. It can be seen that the precision and recall of our feature extraction method are 86.15% and 83.45% on average respectively. The values of F-measure in all but one class are above 80% (except in 'Travel'). It indicates that our method can achieve a good performance. Moreover, the variation of all evaluation measures is stable, as shown in Fig.7. It verifies that our method is suitable for different classes of app description texts.

Table 5 The performance of feature extraction in our approach

Class	Precision	Recall	F-measure
Communication	86.37%	76.37%	81.06%
Photography	93.75%	78.95%	85.72%
Sports	84.93%	93.75%	89.12%
Education	81.79%	86.43%	84.05%
Game	93.36%	83.34%	88.07%
Travel	76.67%	81.88%	79.19%
Average	86.15%	83.45%	84.54%

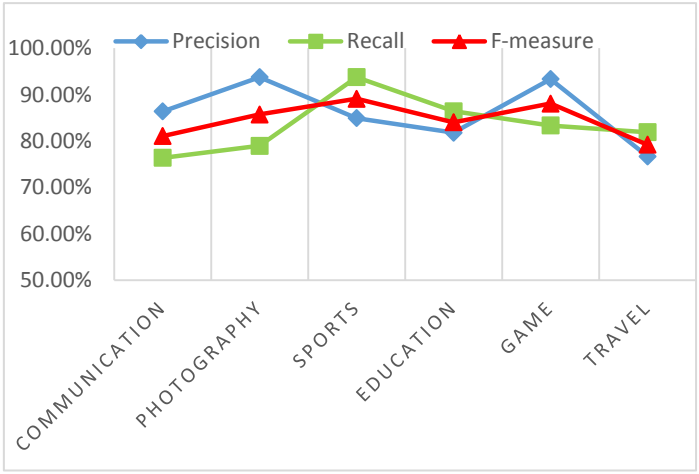

Fig.7. Variation tendency of the performance of our feature extraction method.

Due to the importance of the feature extraction method in our approach, we compared it with existing methods to test its performance. The ideal approach would use these methods to handle the texts in Dataset_1 for getting the compared results. However, the feature extraction methods always take a certain kind of texts as input and there is no research focusing on app descriptions as we discussed in Section 2, so these methods cannot achieve good results in our dataset and it is not fair to evaluate our method by comparing with such results. Moreover, we extract non-functional features which are usually overlooked in many methods. This makes it difficult to compare them with our method. To alleviate the above problems, we choose four typical methods that also use precision, recall and F-measure as the evaluation measures in their research papers, so that we can quote the results directly. Table 6 shows the literatures we used in the comparison. In these methods, S1 and S2 take the first type of texts as input, whereas S3 and S4 take the third type of texts as input (introduced in Section 2). Thus, they can cover the whole research field well.

The results of comparison are given in Table 7. Because the values of evaluation measures of the

compared methods are quoted from their papers, we just give the aggregated results rather than showing them on each category as Table 5. It can be seen that our method achieves the best results on average. The performance of our method is also better in the worst condition (recall is 78.95% and F-measure is 79.19%), which is an important factor affecting the effectiveness of a method. Thus, it indicates that our feature extraction with such performance can be used in practice.

Note that the comparison can validate our method, but it does not mean our method is superior to the compared ones. Because the success of our method depends on pre-defined rules, which need extra efforts to achieve such a good performance.

Due to the literatures of the compared methods aim at solving different problems from ours and feature extraction is just one step in their approach, we did not use them as the objects in the following comparisons.

Table 6 The compared methods.

S1	Extracting software functional requirements from free text documents[15] .
S2	Decision Support for the Software Product Line Domain Engineering Lifecycle[16] .
S3	How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Review[29] .
S4	Extracting features from online software reviews to aid requirements reuse[31] .

Table 7 The results of comparison

		Our method	S1	S2	S3	S4
Precision	Lowest	76.67%	39.30%	**82.00%**	33.50%	51.15%
	Highest	93.75%	**95.10%**	88.00%	91.00%	87.39%
	Average	**86.15%**	80.27%	85.00%	59.15%	62.63%
Recall	Lowest	**78.95%**	18.60%	73.00%	32.10%	78.49%
	Highest	93.75%	**98.50%**	87.00%	73.40%	86.08%
	Average	**83.45%**	66.19%	78.00%	51.30%	82.17%
F-measure	Lowest	**79.19%**	28.50%	73.00%	33.40%	62.62%
	Highest	89.12%	**96.80%**	86.00%	81.30%	82.70%
	Average	**84.54%**	70.68%	80.00%	54.90%	70.80%

In our experiments, we find some factors influencing the performance of our method. Firstly, the parsing tree gained by Stanford Parser is not correct completely, which is a problem in nature language researches. We alleviate this problem by shortening the sentence to be handled and designing more feature extraction rules, and more parsing methods and tools will be utilized in our approach. Secondly, some description texts with low quality affect the performance. In our feature extraction method, we assume that the syntax of app description given by providers is basically correct, but there are some texts containing too much mistakes in fact. Since there are no patterns of such mistakes, it is difficult to be solved directly in our feature extracting process. Thus, we consider to give quantified assessment methods of app descriptions to support the filtering of dataset in our further work.

6.4.2 Evaluation of O-LDA by comparing with the basic LDA

In this comparison, we varied the number of topics (donated as K) and chose the value of K according to the results themselves by analysts for each class in Dataset_1 and Dataset_2. In this way, totally 7 paired topic modelling results were generated and they were evaluated by 10 graduates. Table 8 summarizes the results of the comparison. As noted, we used the Wilcoxon signed ranks test (2-tailed test) for analyzing the results. We can see that the mean scores of O-LDA are higher than LDA in all

the 7 classes of app descriptions, especially in 'Communication' (8.2 for O-LDA and 5.4 for LDA). This difference is significant except in 'Sports', resulting in all but one p-values are smaller than 0.05 (except 0.136 in 'Sports'). Based on this analysis, we can reject hypothesis $H0_{effectiveness}$ in all the classes except 'Sport'. Thus, we conclude that O-LDA is more effective for our problem than the basic LDA.

We analyze the reasons of phenomena above as follows. The basic LDA does not have process of information extraction, so there are redundant words affecting the results of LDA; moreover, we find that the kinds of verbs are much fewer than nouns in the descriptions of a particular app domain (about 1:9 in Datase_2), which means that the frequency of a verb is much higher and causes difficulty to using LDA in our problem. In our further work, we will compare O-LDA with other data analysis techniques and improve the performance by deeply researching its combination with other techniques.

Table 8 The comparison of O-LDA with the basic LDA

	O-LDA		LDA		P-Value
	Mean	Sd	Mean	Sd	(2-Tailed)
Communication	8.2	0.79	5.4	0.84	0.007
Photography	7.0	1.05	4.3	1.41	0.009
Sports	6.8	1.48	5.3	1.70	**0.136**
Education	7.2	1.03	5.2	1.05	0.005
Game	6.5	1.08	4.1	0.99	0.008
Travel	7.1	1.45	5.1	1.10	0.016
Social	7.8	0.92	6.0	0.66	0.004

$H0_{effectiveness}$:There is no difference in effectiveness between the results of O-LDA and the basic

6.4.3 Evaluation of our feature clustering method

In this experiment, each generated cluster of our method and *K*-Means was matched with the answer set cluster with which it shares the most features, and the purities of two methods were calculated for the 7 classes of apps. Fig. 8 shows the results of comparison. It is clearly that the purity of our method is higher than the value of *K*-Means in all conditions, and there is about 10 percent improvement on average. This comparison indicates that our feature grouping method is more appropriate for our problem than *K*-Means.

By analyzing the clusters in this experiment, we summarize the reasons for results of this comparison: firstly, the description of features in our approach can be divided into two parts providing a condition for grouping them, and our method make full use of this condition; secondly, the relationship between features are also utilized in our method, while these factors are not considered in other clustering methods.

Fig.8. Comparison of purity for our feature grouping method and *K*-Means

6.5 Analysis of the case study for Q2

A case study on dataset_2 is conducted and finial DRDM is constructed utilizing our approach to evaluate its usability. In order to construct DRDM automatically, five parameters (introduced in Section 4 and 5) need to be pre-set. However, there is no theory to calculate the values of these parameters, so we firstly assignment their initial values based on our pervious experiments; then randomly sample date from the results of each step in the whole process and evaluate them manually to modify the related parameters until the results are acceptable; and the final values of parameters are set as default values in our tool to support the automatic process of more analysis tasks. The concrete values of parameters we used have been given when the parameters are defined, and the results of our case study are shown and analyzed in this sub-section.

6.5.1 An example of CDM in our case

The app descriptions in Dataset_2 are transformed to CDMs automatically by the tool we developed. Table 9 gives an example of concerns in one CDM constructed from the description text shown in Table 1 based on the results of O-LDA shown Table 3(b). There are total 36 concerns generated from the app description, where 17 common-concerns, 10 special-concerns and 9 property-concerns. It can be seen that almost all the features in the description text are contained in the concerns. Meanwhile, a property-concern is meaningless itself, but we can gain meaningful features when combine it with common/special-concerns. For example, combining concern 6 with concern 5, we can get a meaningful phase 'discover accounts from all over the world' to describe a feature of the app.

Although the relationship between concerns are not shown in the table, they can be established by the process introduced in Section 4.3.2. For example, concern 19 'share photos' and 20 'share videos' are extracted from the same sentence 'Share multiple photos and videos (as many as you want!) to your story', so there is a Common_of relationship between them.

Table 9 Concerns in one CDM

type	The descriptions in concerns				
Common	1.capture moments	2.share moments	4.follow family	10.sharing moments	12.post photos
	13.post videos	14.edit photo	15.edit videos	19.share photos	20.share videos
	21.Photo disappear	22.videos disappear	27.discover photos	28.discover videos	29.send messages
	31.send photos	32.send videos			
Special	3.follow friends	5.discover account	7.sharing things	8.join community	17.combine clips
	24.watch stories	25. view stories	26.follow accounts	33.send posts	34.share posts
Property	6.from all over the world		9.of over 500 million people		11.of your day
	16.with filters and creative tools		18.into one video	30.in the Explore tab	23.after 24 hours
	35.from your feed directly to friends		36.to Facebook, Twitter, Tumblr and other social networks		

6.5.2 Introduction and analysis of the DRDM constructed in our case

In the DRDM constructed in our case study, there are total 13308 concerns generated from Dataset_2: 8768 of them are related to functional features, while 4540 are related to non-functional features. After data integration (introduced in Section 5.1), we obtain 335 clusters of common-concerns and 553 clusters of special-concerns.

Table 10 gives the statistic number of clusters and concerns. The 8768 function related concerns are further divided into two types: 2310 of them are common concerns, and 6458 are special. Furthermore, the common concerns are related to the topics (shown in Table 3(b)). For example, 487 concerns are

related to the Photo(T1) and are grouped into 72 clusters, whereas the special concerns are analyzed as a whole, and they are grouped into 553 clusters.

In order to analyze the DRDM more intuitively, we give some examples of each part in DRDM next.

Table 10 Statistics of DRDM in our case.

Total: 13308		
Common: (2310)	T1: $\|Cc^{t_1}set\|$=72	$\sum\|Cc^{t_1}\|$=487
	T2: $\|Cc^{t_2}set\|$=75	$\sum\|Cc^{t_2}\|$=567
	T3: $\|Cc^{t_3}set\|$=66	$\sum\|Cc^{t_3}\|$=507
	T4: $\|Cc^{t_4}set\|$=59	$\sum\|Cc^{t_4}\|$=348
	T5: $\|Cc^{t_5}set\|$=63	$\sum\|Cc^{t_5}\|$=401
Special:(6458)	$\|Cc^{s}set\|$=553	
Property:(4540)		

Table 11 shows part of elements in $\mathrm{Cc^T}$set, which is the results of data integration of common concerns. The titles in Table 10 are as same as the topics in Table 3(b). For a top t_i, we give five examples of elements in $Cc^{t_i}set$. For example, in $Cc^{t_1}set$ which is related to Photo(T1), the first element is $Cc_1^{t_1}$={'Take photo', 'Select photo', 'Choose photo', 'Get photo', 'Bring photo',…}, where 'Take photo' is taken as the Name of $Cc_1^{t_1}$ for its high priority (determined by the method introduced in 5.2), and the **verbs** is the set of $C.Des^{verb}$ clustered in our method. In addition, after classifying and clustering the concerns related to T1, we have gained two kinds of clusters related to 'Photo' and 'Picture' separately, but at the merging step of data integration, they are merged to form a kind of clusters for the intense relationship between the concerns in them. Thus, the concerns—such as 'Take picture', 'select picture '— are also contained in $Cc_1^{t_1}$. From Table 10, we can see that each cluster in $Cc^{t_i}set$ is meaningful and describes a kind of function features under one topic, so it can help developers discover common knowledge to a particular domain quickly.

Table 11 Part of $\mathrm{Cc^T}$set

Photo(T1)		**Video(T2)**		**People(T3)**		**Text(T4)**		**Love(T5)**	
Name	**verbs**	**Name**	**verbs**	**Name**	**verbs**	**Name**	**verbs**	**Name**	**verbs**
Take (photo, picture)	Select	Make (video)	Take	Chat (people)	Talk	Send (text, message)	Post	Want (love)	wish
	Choose		Get		Communicate		Deliver		care
	Get		Add		Invite		Throw		make
	Bring		Need		interact		Convey		Deserve
Share (photo, picture)	Send	Watch (video)	Enjoy	Find (people)	Discover	Reply (text, message)	Receive	Fall (love)	Receive
	Post		Tag		See		Add		Get
	Download		Follow		Hear		Generate		Find
	Tag		Use		Learn		Get		Start
Find (photo, picture)	Discover	Send⑥ (video)	Post	View (people)	Look	Search (text, message)	Find	Express (love)	Present
	Look		Broadcast		Want		See		Show
	See		Install		Like		Know		Tell
	Poke		Record		expect		Love		allow
Edit (photo, picture)	Publish	Save (video)	Download	Connect (people)	Join	Manage (text, message)	Show	Share (love)	Send
	Quote		Keep		Meet		Set		Flirt
	Save		Support		Let		Define		Spread
	add		Help		Install		Do		Write

$Cc^{s}set$ in DRDM.F is constructed by data integration of special concerns. The elements in $Cc^{s}set$ are clusters of concerns. Each cluster expresses a function that is extraordinary in the domain. Table 12 gives some examples of elements Cc_i^s in $Cc^{s}set$, where each cell in it shows a cluster of concerns. We can see that some of clusters can be easily understood (such as ① in Table 9), whereas others are not (such as ② in Table 9) and need to be further analyzed. Meanwhile, some of them may not be discovered in a simple domain analysis (such as ③ in Table 9), (but) these information may be helpful for generating creative ideas. Overall, we find that the information in $Cc^{s}set$ is not as obvious as the one in $Cc^{t_i}set$ in our case study. The reason is that: there is more detailed information in $Cc^{s}set$

(such as ④ in Table 9), and these features need be combined with other features together to express domain knowledge completely.

Table 12 Examples of elements in $Cc^s set$

Find film ① Discover name Detect menu Describe move Show ticket …	Meet neighborhood② Choose group Take dinner Choose collection Consume power …	Share tag ⑤ Chat sticker Use symbol Look arrow Use trademark …	Feed kid ③ Bring baby Play child Work adult Meet boy …	Read feedback Show smile Express dislike Understand talk Usher sign …
Help husband Facilitate boyfriend Assist player Serve lover Attend companion …	Create hotspot Make combination Create institution Build team Clear junto …	Enjoy question Match problem Touch hair Apply beta Love problem …	Listen audience Attend universe Come campus Manage environment Attend class …	Save option Write idea Support choice Keep knowledge Maintain layout …
Leave partner Allow driver Offer employee Provide stranger Benefit visitor …	Develop craft Grow path Go business Become development Manage site …	connect friend Direct developer Send viewers Send employee Transmit messenger …	Check rooms Catch card Watch key View list See position …	Hit clip ④ Remove keyboard Choose button Remove memory Remove window …

The set of relationship R between the clusters of concerns is established to construct DRDM. This step relies on the sentences that contain different types of concerns (common-concerns and special-concerns). For example, in the sentence 'The app has features such as to **send video** and enjoy to **chat with anime sticker.**', there are two features 'send video' and 'chat with anime sticker', which belong to the common type and the special type respectively. Specially, 'sent video' is an element in $Cc_3^{t_3}$ (⑥ shown in Table 10) and 'chat with anime sticker' is an element in cluster ⑤ shown in Table 11, so the Common_Special (defined in Section 3.1) relationship between the two concerns is part of $R_{f\text{-}f.}$rel between the two clusters above. By calculating the percentage of such relationship in ⑥ and ⑤, the $R_{f\text{-}f.}$degree is identified as *High*. We review ⑥ and ⑤ manually, and find that ⑤ is a feature which supports the better realization of feature ⑥.

Based on the DRDM constructed in the case study, a report is generated. Its framework is shown in Fig.6 and its usefulness is evaluated in our surveys.

6.6 Results and Analysis of the surveys for Q2

The surveys in the experiments for *Q2* were based on the report generated in our case study and took place separately during the lesson hours of a software engineering course. To ensure the participants have high motivation, they were promised to get extra scores in the course. At the same time, they could quit the experiment without any negative effect. Before experiment, we gave a detailed introduction about the tasks and the basic notations of features. Moreover, before Survey_2 (the survey on the usefulness of report), the three groups were isolated from each other and given additional introductions on the material given to them(raw app descriptions, DRDM, or FM_{-8}). Then, the participants were asked to complete the tasks in the survey, and they could change their answers at any time before closing the surveys. The analysis of results is given in this sub-section.

6.6.1 Survey for evaluating the determination of priority

The Survey_1 evaluated the determination of priority in our DRDM by 7 pairs of comparisons. We applying the Student's t-test (2-Tailed) when the samples distributed normally, and the results are shown in Table 13. The first column is the type of the features in each comparison and the second column is the concrete keywords corresponding to the features. From Table 13, we can get to observations as follow.

Firstly, the average precisions for all keywords are higher than 70%, whether for Top 10 concerns or for Top 11-20 concerns (except one condition). This means the concerns with high priorities are valuable for domain analysis.

Secondly, the precisions of Top-10 concerns are higher than the ones of Top 11-20 in all keywords, and this differences is significant except in 'Love' and 'Hotel', resulting in the p-values are smaller than 0.05. Moreover, this observation is much clearer when we take all samples as a whole (p-value<0.001 totally). It indicates the top-10 concerns are more important than the top 11-20 ones.

Based on these analysis, we can reject hypothesis $H0_{priority}$ and conclude that the determination of priority in our method is reasonable.

Table 13 The statistic results of Survey_1

type	Keyword	Average precision of Top-10	Average precision of Top 11-20	p-value (2-Tailed)
Common	Photo	79.7%	72.0%	0.001
	Video	78.1%	71.8%	0.010
	People	77.8%	74.0%	0.028
	Text	80.8%	74.2%	0.004
	Love	71.5%	70.2%	**0.507**
Special	Film	77.5%	75.0%	**0.185**
	Hotel	73.5%	58.2%	<0.001
Total		77.0%	70.8%	**<0.001**

$H0_{priority}$: there are no difference in precision between two groups.

6.6.2 Survey for evaluating the usefulness of our report

To evaluate the report of DRDM, we compared it with raw app descriptions and FM_{-8} based on four variables collected from each questionnaire in Survey_2. The analysis is given next.

6.6.2.1 Comparing DRDM with raw app descriptions

Table 14 summarizes the results regarding the differences between raw app descriptions and DRDM in helping answer the questionnaire. We used the Mann-Whitney U test (2-tailed) for all the hypotheses ($H0^1_{score}$/ $H0^1_{time}$ / $H0^1_{diff}$ / $H0^1_{con}$). We get the following observations:

Firstly, the mean score is higher when the participants answering the questionnaire using DRDM than using raw app descriptions (75.8 for DRDM and 72.3 for descriptions), but this difference is not significant due to the p-value is 0.393;

Secondly, the participants based on DRDM spent significantly shorter time on completing the tasks than the ones based on raw app descriptions (110.15 versus 153.15 on the mean and p-value is less than 0.001);

Finally, from the point of subjective feelings, the participants felt more confident and less difficult in answering the questionnaire when using DRDM than using raw app descriptions (4.05 and 2.7 for DRDM versus 3.30 and 3.4 for raw app descriptions, respectively). This difference is significant (the values of p-value are 0.033 and 0.010).

Based on the analysis above, hypotheses $H0^1_{time}$, $H0^1_{diff}$ and $H0^1_{con}$ can be rejected, whereas hypothesis $H0^1_{score}$ should be kept.

Table 14 Results of Comparison (Raw app descriptions versus DRDM)

	Raw App Descriptions		DRDM		P-Value
	Mean	Sd	Mean	Sd	(2-Tailed)

Score	72.3	13.22	75.8	11.79	0.393
Time	153.15	21.19	110.15	11.13	<0.001
Confidence	3.30	1.12	4.05	0.82	0.033
Difficulty	3.4	0.88	2.7	0.65	0.010

6.6.2.2 Comparing DRDM with FM_{-8}

Table 15 summarizes the results on the differences between FM_{-8} and DRDM, applying Mann-Whitney U test (2-tailed) to analyze the data for all the hypotheses ($H0^2_{score}$/ $H0^2_{time}$ / $H0^2_{diff}$ / $H0^2_{con}$). It can be seen that the participants answering the questionnaires based on DRDM got slightly higher scores than the ones based on FM_{-8} (75.8 for DRDM and 73.3 for FM_{-8}), and they responded a higher confidence and a lower difficulty (4.05 versus 3.95 and 2.7 versus 2.9, respectively). However, these differences are insignificant in statistics (p-value are 0.626, 0.659 and 0.390 respectively).With respect to time to complete the tasks, the participants using FM_{-8} needed a significant longer time than using DRDM (p-value is 0.026) for answering questionnaire. Thus, we can only reject the hypothesis $H0^2_{time}$.

Table 15 Results of Comparison (FM_{-8} versus DRDM)

	FM_{-8}		DRDM		P-Value
	Mean	Sd	Mean	Sd	(2-Tailed)
Score	73.3	14.63	75.8	11.79	0.626
Time	121.15	15.04	110.15	11.13	**0.026**
Confidence	3.95	0.60	4.05	0.82	0.659
Difficulty	2.9	0.72	2.7	0.65	0.390

In order to further study the differences between DRDM and FM_{-8}, we analyzed the scores of the second and third parts in the questionnaires separately, and the results are presented in Table 16. With respect to the second part of questionnaire, it was the questions that the answers could be obtained from the material (FM_{-8} or DRDM) directly. It can be seen that the mean score of this part is slightly higher when the participants using DRDM than using FM_{-8} (39.3 versus 38.1 where full score is 50), although the difference is insignificant (p-value of 0.095). With respect to the third part of questionnaire, it was a totally open task. Since the participants were asked to classify the features in their answers into basic and creative ones, we also evaluated the responds from these two aspects: for the basic features, the answers based on FM_{-8} obtained slightly higher scores than the ones based on DRDM (30.3 versus 29.4 where full score is 40); while for the creative features, the answers based on DRDM obtained higher scores than the ones based on FM_{-8} (7.1 versus 4.9 where full score is 10) and this difference is significant(p-value of 0.001). This means DRDM is better in supporting the generation of creative ideas.

Table 16 Comparison FM_{-8} and DRDM from different parts of questionnaire.

		FM_{-8}		DRDM		P-Value
		Mean	Sd	Mean	Sd	(2-Tailed)
Second part		38.1	6.97	39.3	5.85	0.095
Third part	Basic	30.3	7.35	29.4	6.23	0.469
	Creative	4.9	2.10	7.10	1.52	**0.001**

6.6.2.3 Discussion

From the analysis above, the usefulness of raw app descriptions and DRDM is significant different. When using DRDM, participants could complete the tasks more easily within shorter time, and their

confidence is higher. We believe that is because DRDM organizes the domain knowledge in a tree-like form and the features in it are specified explicitly, so it is faster for the users to obtain the information they need. With the insignificant differences in scores, we analyze the reasons from two points: one is that DRDM stems from the raw app descriptions, so it contains the same domain knowledge as raw app descriptions; the other is that the number of app description is still small, so the participants could search the whole dataset for completing the tasks although this needed more time. However, when there are a huge number of app description texts, such 'searching strategy' may become very difficult to conduct. Our further research on a larger set of app descriptions will investigate this hypothesis.

However, when comparing DRDM with FM_{-8}, the differences are not so significant as above. The main difference is that the time for answering the questionnaire required was shorter when using DRDM than when using FM_{-8}. The phenomenon may be caused by different ways of using the two models: when obtaining information, the participants used the DRDM in a top-down way, which searches for features from the abstract to the concrete; while they interacted with FM_{-8} by providing feedback on candidate features for acquiring information. This difference makes FM_{-8} and DRDM adapt to the questions with different conditions. When the question contained explicit and concrete initial information about the task, the participants could use this information as the inputs of FM_{-8} to obtain the answers; but if the question was open, the participants needed to give the initial information themselves firstly for using FM_{-8}, which may cost much time for the participants with limited background knowledge. This problem became more serious when answering the third part (a totally open task) of the questionnaire. Whereas our DRDM could alleviate this problem because it provides the information in a way of choices, so that the participants only need to choose the related one, step by step based on their understanding.

In addition, we also found significant differences between the usefulness of FM_{-8} and DRDM with respect to creative tasks. To analyze the reason of this difference, we further investigated the response based on DRDM and found that most of creative features given by the participants came from the supplement part of the report (the presentation of DRDM). Due to the incomplete nature of app descriptions, some features extracted from them are useful but the relationship between them and others are not gave clearly in descriptions. This causes difficulty to discovering such features from FM_{-8}, which recommends the information based on relationship; Whereas DRDM provides a more open way for information searching by giving a supplement part in the report.

In summary, it seems that DRDM is more useful for overall analyzing the domain in the start-up stage of software development, when the idea of app is not mature. And the flexibility of DRDM seems to be beneficial for generating creative ideas.

6.7 Threats to Validity

Although the results were good in our experiments, the validity of our study suffers several threats. We analyze these threats and give the actions taken to alleviate them from two aspects.

6.7.1 Threats to the validity of methods in our approach

There are two threats in this aspect concerning the limitations of methods themselves in our approach.

Firstly, the authors have limited experience of app developments, so the feature extraction rules defined in this paper may be not adapt to all the app descriptions. Therefore, we evaluated the rules in different kinds of app descriptions as much as possible for completing them. Additionally, the

extendibility of rules can also alleviate this threat. We define the expressions of rules and give their classifications and priorities, this provides the basis for generating new rules to make our method adaptive to various conditions.

Secondly, the market attributes used in our quantified method are limited. We now use two attributes (Rating and Downloads) for determining the priority of domain knowledge, but there are more attributes (such as 'Price', 'Time') in real app markets. The changing of our formulas is uncertain when these attributes are introduced. However, since the rating and downloads are two common and important attributes which determine the vitality of an app directly, the formulas based on them can reflect the actual value of the product to some extent. This have been proved in our survey. Moreover, we have given a more general formula for extending our method to incorporate other attributes.

6.7.2 Threats to the validity of experiments

The threats from this aspect concern the factors affecting the validity of our experiments, especially the surveys. We analyze these threats as follow.

The first threat stems from the participants, whose knowledge and skills in domain analysis might impact the results. Hence, we choose the participants with similar background, that is they are from the same major and the same grade, and they got similar scores in their previous classes. We also guaranteed the participants were distributed among three groups uniformly. To further mitigate this threat, we deigned the first part in the questionnaire to ensure the participants comprehended the basic notations of domain analysis.

The second threat is about the data used in the experiments. As two datasets are created based on Google Play, it is unclear whether our work can achieve similar results when being applied to other app marketplaces. Due to the complexity of our experiments, we could not evaluate our approach with additional datasets in a short time. But we find app descriptions are similar in different app markets by reviewing them manually, so we believe that our approach can also be suitable for them. In the future, we wish to extend our experiments to large-scale datasets in different domains of app to generalize the results.

The third threat focuses on the analysis of outcomes in experiments. We analyzed the outcomes using a T-test for paired samples which has normal distribution; while we adopted the Mann-Whitney U test for independent samples and the Wilcoxon signed ranks test for dependent samples when the outcomes was not distributed normally[41] [42] . These tests are well suited for small samples and are robust. They guarantee the effectiveness of our conclusions in experiments.

7. Conclusion and Future Work

A key issue in domain analysis is to extract features automatically from related product artifacts. In this paper, we take app descriptions as inputs and propose an approach utilizing data technologies to mine domain knowledge from them automatically. In our approach, we establish CDM to describe the information of features in a single app description based on predefined feature extraction rules and a modified LDA process (O-LDA). We also construct DRDM to formalize the overall knowledge in the domain based on our data integration process, including classifying, clustering and merging. In addition, the priorities of obtained knowledge are determined by our quantified method. The results of experiments shows that the performance of automatic construction of CDM is good: F-measure is above 0.8 and its variation tendency is stable. We expect that the results from our approach can be applied to the feature extractions in other domain analysis. Meanwhile, the surveys show that DRDM

can support for domain analysis effectively, especially for overall analyzing the whole domain and generating creative ideas.

References

[1] Lian X, Rahimi M, Cleland-Huang J, Zhang L, Ferrari Remo, Smith M. Mining Requirements Knowledge from Collections of Domain Documents[C]// International Requirements Engineering Conference. 2016.

[2] Lisboa L B, Garcia V C, Lucrédio D, et al. A systematic review of domain analysis tools[J]. Information & Software Technology, 2010, 52(1):1-13.

[3] Mefteh M, Bouassida N, Benabdallah H. Mining Feature Models from Functional Requirements[J]. 2016.

[4] Bakar N H, Kasirun Z M, Salleh N. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review[J]. Journal of Systems & Software, 2015, 106:132-149.

[5] Kang K C, Cohen S G, Hess J A, et al. Feature-oriented domain analysis (FODA) feasibility study (cmu/sei-90-tr-21[J]. Georgetown University, 1990, 4(4):206-207.

[6] Kang K C, Kim S, Lee J, et al. FORM: A feature-oriented reuse method with domain specific reference architectures[J]. Annals of Software Engineering, 1998, 5(1):143-168.

[7] Martin W, Sarro F, Jia Y, et al. A Survey of App Store Analysis for Software Engineering[J]. IEEE Transactions on Software Engineering, 2016, PP(99):1-1.

[8] Hariri N, Castro-Herrera C, Mirakhorli M, et al. Supporting Domain Analysis through Mining and Recommending Features from Online Product Listings[J]. IEEE Transactions on Software Engineering, 2013, 39(12):1736-1752.

[9] Coplien J, Hoffman D, Weiss D. Commonality and Variability in Software Engineering[J]. IEEE Software, 1998, 15(6):37-45.

[10] Benavides D, Segura S, Trinidad P, et al. FAMA: Tooling a Framework for the Automated Analysis of Feature Models.[C]// International Workshop on Variability Modelling of Software-Intensive Systems, Vamos 2007, Limerick, Ireland, January 16-18, 2007. Proceedings. 2007.

[11] Antkiewicz M, Czarnecki K. FeaturePlugin: feature modeling plug-in for Eclipse.[C]// OOPSLA Workshop on Eclipse Technology Exchange, Etx 2004, Vancouver, British Columbia, Canada, October. 2004:67-72.

[12] Sharma A K, Sharma A. Tool support for feature-oriented software development: featureIDE: an Eclipse-based approach[C]// OOPSLA Workshop on Eclipse Technology Exchange, Etx 2005, San Diego, California, Usa, October. 2005:55-59.

[13] Acher, M. Cleve, A., Perrouin, G., Heymas, P., Vanbeneden, C., Collet, P., Lahire and P. (2012) On Extracting Feature Models from Product Descriptions. Proc. VaMoS 12, Leipzig,Germany, January 25–27, pp. 45–54. ACM, New York, USA.

[14] Acher, M., Cleve, A., Collet, P., Merle, P., Duchien, L. and Lahire, P. (2014) Extraction and evolution of architectural variability models in plugin-based systems. SoSyM, 13, 1367–1394.

[15] Mu Y, Wang Y, Guo J. Extracting Software Functional Requirements from Free Text Documents[M]. 2009.

[16] Bagheri E, Ensan F, Gasevic D. Decision support for the software product line domain engineering lifecycle[J]. Automated Software Engineering, 2012, 19(3):335-377.

[17] Niu N, Savolainen J, Niu Z, et al. A Systems Approach to Product Line Requirements Reuse[J]. Systems Journal IEEE, 2014, 8(3):827-836.

[18] Mefteh M, Bouassida N, Benabdallah H. Mining Feature Models from Functional Requirements[J]. 2016.

[19] Rahimi M, Mirakhorli M, Clelandhuang J. Automated extraction and visualization of quality concerns from requirements specifications[J]. 2014:253-262.

[20] Itzik N, Reinhartzberger I, Wand Y. Variability Analysis of Requirements: Considering Behavioral Differences and Reflecting Stakeholders Perspectives[J]. IEEE Transactions on Software Engineering, 2016, 42(7):1-1.

[21] Yu Y, Wang H, Yin G, et al. Mining and recommending software features across multiple web repositories[C]// Asia-Pacific Symposium on Internetware. 2013:9.

[22] Ferrari A, Spagnolo G O, Dell'Orletta F. Mining commonalities and variabilities from natural language documents[C]// International Software Product Line Conference. 2013:116-120.

[23] Lian X, Rahimi M, Cleland-Huang J, Zhang L, Ferrari Remo, Smith M. Mining Requirements Knowledge from Collections of Domain Documents[C]// International Requirements Engineering Conference. 2016.

[24] Davril J M, Delfosse E, Hariri N, et al. Feature model extraction from large collections of informal product descriptions[C]// Joint Meeting on Foundations of Software Engineering. ACM, 2013:290-300.

[25] Iacob C, Harrison R. Retrieving and analyzing mobile apps feature requests from online reviews[J]. 2013:41-44.

[26] Iacob C, Harrison R, Faily S. Online Reviews as First Class Artifacts in Mobile App Development[C]// International Conference on Mobile Computing, Applications, and Services. 2013:47-53.

[27] Vu P M, Nguyen T T, Pham H V, et al. Mining User Opinions in Mobile App Reviews: A Keyword-based Approach[J]. 2015.

[28] Vu P M, Pham H V, Nguyen T T, et al. Phrase-based extraction of user opinions in mobile app reviews[C]// The, Ieee/acm International Conference. 2016:726-731.

[29] Guzman E, Maalej W. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews[C]// Requirements Engineering Conference. IEEE, 2014:153-162.

[30] Villarroel L, Bavota G, Russo B, et al. Release planning of mobile apps based on user reviews[C]// The, International Conference. 2016:14-24.

[31] Bakar N H, Kasirun Z M, Salleh N, et al. Extracting features from online software reviews to aid requirements reuse[J]. Applied Soft Computing, 2016.

[32] Almeida E S D, Cavalcanti A P C, Alvaro A, et al. The Domain Analysis Concept Revisited: A Practical Approach[J]. Lecture Notes in Computer Science, 2006:43-57.

[33] Lucrédio D, Fortes R P D M, Almeida E S D, et al. Performing Domain Analysis for Model-Driven Software Reuse[M]// High Confidence Software Reuse in Large Systems. Springer Berlin Heidelberg, 2008:200-211.

[34] Yin Jin,Huaxiao Liu , An approach to analyzing and verifying aspect-oriented requirements model[J]. Chinese Journal of Computers,2013,36,63-73

[35] Huaxiao Liu, Ying Jin, Pengfei Ma. An Approach to Requirements Change Impact Analysis[J]. Journal of Computer Research and Development, 2013, 50(8):1769-1777.

[36] Liu H, Liu Y, Liu L. The verification of program relationships in the context of software cybernetics[J]. Journal of Systems & Software, 2016.

[37] Nigam K, McCallum A, Thrun S, and Mitchell T, Text classification from labeled and unlabeled documents using EM. Machine Learning, 2000. 39(2): 103-134.

[38] Miller G A. WordNet: a lexical database for English[M]. ACM, 1995.

[39] Zhao Y, Karypis G. Evaluation of hierarchical clustering algorithms for document datasets[C]// Eleventh International Conference on Information and Knowledge Management. ACM, 2002:515-524.

[40] C.D. Manning, P. Raghavan, and H. Schütze. Introduction to Information Retrieval. Cambridge Univ. Press, 2008.

[41] Itzik N, Reinhartzberger I, Wand Y. Variability Analysis of Requirements: Considering Behavioral Differences and Reflecting Stakeholders Perspectives[J]. IEEE Transactions on Software Engineering, 2016, 42(7):687-706.

[42] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B. and Wesslén, A. (2000) Experimentation in Software Engineering –An Introduction. Kluwer Academic Publishers.