International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 5, October 2023, pp. 5501~5516
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i5.pp5501-5516

Query expansion using novel use case scenario relationship for finding feature location

Achmad Arwan[1,2], Siti Rochimah[1], Chastine Fatichah[1]
[1]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[2]Department of Informatics, Faculty of Computer Science, Brawijaya University, Malang, Indonesia

Article Info

Article history:

Received Jan 30, 2023
Revised Apr 18, 2023
Accepted Apr 24, 2023

Keywords:

Feature location
Latent Dirichlet allocation
Query expansion
Topic model
Use case scenario relationship

ABSTRACT

Feature location is a technique for determining source code that implements specific features in software. It developed to help minimize effort on program comprehension. The main challenge of feature location research is how to bridge the gap between abstract keywords in use cases and detail in source code. The use case scenarios are software requirements artifacts that state the input, logic, rules, actor, and output of a function in the software. The sentence on use case scenario is sometimes described another sentence in other use case scenario. This study contributes to creating expansion queries in feature locations by finding the relationship between use case scenarios. The relationships include inner association, outer association and intratoken association. The research employs latent Dirichlet allocation (LDA) to create model topics on source code. Query expansion using inner, outer and intratoken was tested for finding feature locations on a Java-based open-source project. The best precision rate was 50%. The best recall was 100%, which was found in several use case scenarios implemented in a few files. The best average precision rate was 16.7%, which was found in inner association experiments. The best average recall rate was 68.3%, which was found in all compound association experiments.

Corresponding Author:

Siti Rochimah
Department of Informatics, Institut Teknologi Sepuluh Nopember
Teknik Kimia Street, Surabaya, East Java 60117, Indonesia
Email: siti@if.its.ac.id

1. INTRODUCTION

Feature location is a technique for tracing a function of software into specific sector of source code. The feature location is useful when a developer wants to fix, change or improve a method in a code. Changes to code appear to be easy if the amount of code is small or the programmer who wants to fix it is the same person, which means he or she already understands the variables, parameters, and logic of the code. However, the changing the code will be difficult when the source code of the project is large or the programmer who makes the changes is a different person than the creator who has never even worked on the code. For this reason, programmers need to perform program comprehension first, and studies show that programmers need approximately 21.5 hours a week (58% of 37.5 hours per week) [1].

There was a big gap between the usage of tokens on software requirement and the tokens of source code. It is greatest challenge in feature location research. The tokens in the software requirements level use abstract words such as billing, enter personal health records, and view prescriptions (health record domain). In contrast, the tokens in the source code are technical or specific on how logic works on source code (e.g., AddPHAAction, personnelDAO, and setPassword). Based on this fact, we need some techniques to create a pair among of tokens in the requirements and source code.

Survey research on feature locations was previously carried out by Razzaq *et al.* [2]. Razzaq *et al.* [2] collected many papers (170 papers) related to it. The research uncovered various of techniques, measurements, datasets. As a result, most research using technique of information retrieval from software repositories. The software repositories were identifier, comments, variables, parameters, and methods [3]–[5]. The algorithms were varied, such as latent semantic indexing (LSI) [4], latent Dirichlet allocation (LDA) [4], term frequency-inverse document frequency (TFIDF) [5], vector space model (VSM) [6] and some other deep learning [3].

The next feature location research was performed by using text/IR processing to research how to standardize feature locations [4]. It proposes how to standardize the techniques (eight techniques) of comparing on feature location by experiment in an empirical design. The eight techniques include LDA-Gensim, LDA-R, LDA-Gibbs, LSI-Gensim, LSI-MATLAB, VSM-Lucene, VSM-MATLAB, and VSM-TraceLab. These methods were evaluated in twelve case studies to measure the performance. A few arrangements use the recommendations several other studies. The results show that different underlying techniques perform variously and that VSM-Lucene and LSI-MATLAB perform superior among other techniques.

The next feature location research was done by applying weighting to structure of source code using LDA [7]. The study proposes an approach to add extra-information to the class (method, comments, and variables). It also set with many LDA configurations to define the extent to which set or weight influence the accuracy of feature locations. The result was comments weight and naming of method could gain better accuracy, while variables and parameter gain less accurate.

Another study was conducted by creating query repair automation [8]. The research aims to determine how to improve user queries in finding feature locations. They argue that if the user cannot provide the correct query, then the machine cannot recommend the correct answer. The techniques used were encoding and using genetic algorithms to automate query repair. The data used are also specific, namely, the rail system in Spain, which does not use the unified modelling language (UML) model. The results show that the level of accuracy is still not good, namely, 14% precision, and 37% recall.

The next research was done by using a structural approach to locate the bug [9]. The research seeks bug locations by using indexed classes, methods, variables, and comments. The structure was employed in the study to sum the matching results between the tokens in the query and the tokens in the class, method, and variable categories. The research also found that the tokens in the variable names have a great match with the tokens in the bug. The result of this research is that the accuracy of the location of the code associated with the bug is 60%.

Another study was conducted to determine how and when structural information could help find traceability links [10]. The structural information includes function calls, inheritance, or realization relationships on source code. The goal of the research was how to find the initial link point and continue to seek for links until there are no more points. The results of this study showed an accuracy rate of 58%.

The next research was intended to determine feature facets within the software [11]. It exposing the new approach by employing pull request, commits, analyzing logs, and systematic code reviews. The natural language processing was done to tackle the problem on pull request analysis. The logs were help to identifying the new feature from the pull request and commit and very helpful as entry point for several features. The results were they were able to use some parameters such as the commit messages, commits author, pull request, release log as information sources to determine the feature facets.

The next technique was comparing several search strategies using model to find feature location [12]. The idea to find something within abstraction layer such as model was easier rather than find something from source codes. It comparing several search algorithms such evolutionary algorithm, random search, hill-climbing, iterated local search, and the hybrid of evolutionary algorithm and hill-climbing (EHC) to determine what algorithm was the best amongst all. The EHC was the best search-based strategy which reach precision 81% and recall 83%. The limitation of research was the dataset originated from industrial based and the models were not common (induction hob domain-specific language) which mean cannot replicate easily.

The recent research was how to know the effect of feature characteristic against the performance of feature location [13]. The study introduced several new metric characteristics such as relative feature size (RFS), tangled elements (TE), crosscutting in features (CIF), unique lexical coverage (ULC), lexical saturation (LS) and query size (QS) to evaluate performance of feature location. The evaluation measurement was precision, recall, mean average precision (MAP), mean reciprocal rank (MRR). As a results, the RFS was the main regressor of precision and MAP using several algorithms such Page rank, LDA, LSI and VSM. An ULC was the main regressor of recall using several algorithms such Page rank, LDA, LSI and VSM. The CIF was the main regressor to MRR several algorithms such Page rank, LDA, LSI and VSM.

The use case scenarios are software requirement documents that explain the input data, logic, rules, actor, and output data of a function [14]. The programmer implements the source codes based on the description of the use case scenario; therefore, querying specific area of feature could be done using the use case scenario. However, the number of research that use the use case scenario as a query is quite rare, which is caused by the minimum of datasets that include use case scenarios in the projects. Our previous research [15] was performed by applying NLP (noun tagging and verb tagging) to use a case scenario and used it as a query for information retrieval. It could predict the feature location with an average precision of 11% and a recall of 4%. Another our previous research also used clustered use case scenarios as query expansion to find feature location [16]. The results were quite good on recall rate (56%).

The story on use case scenario is sometimes described another sentence in other use case scenario. Or it needs more explanation which could be found on other use case scenario. Based on the facts on iTrust data [17], the usage of use case scenario relationships may be advantageous in finding feature locations.

This study contributes to creating a novel method for feature location by making expansion queries in feature locations by finding the relationship between use case scenarios. The inner association, outer association and intratoken association were the original ideas to capture additional tokens from other use case scenarios. As a result, the query becomes more numerous and could increase precision and recall rate of feature locations. The expanded use case scenarios were used as queries for information retrieval based on topic modeling of source code.

2. BACKGROUND

In this section, we will provide a brief introduction to information retrieval, topic modeling, and query expansion. Additionally, we will elaborate on our approach of the use case scenario relationship model. The final, we also elaborate how to implement it in a feature location case study.

2.1. Information retrieval

Information retrieval is a prevalent technique in feature location. It composed by a number of processes, including preprocessing and NLP, and creates a group of token into a corpus [18]. Then, users could prompt some queries. The token from query was compared against the token with specific similarity methods such as cosine and Jaccard to gain the high-rank of precision and recall. To ensure the validity results of information retrieval methods, we employed precision and recall as a common techniques for many researchers [8], [19].

$$Precision = \frac{items\ relevant \cap\ items\ retrieved}{items\ retrieved} \quad (1)$$

$$Recall = \frac{items\ relevant \cap\ items\ retrieved}{items\ relevant} \quad (2)$$

Items relevant is the amount of files that were attached to an individual feature, whereas the items retrieved were the number of files that were recommended in this research.

2.2. LDA

LDA [20] is an unsupervised probabilistic procedure to determine the topic distribution on a corpus. The corpus is extracted from documents (e.g., Source code), which consist of tokens. Each document was given the probabilistic distribution to determine the topic proportion.

LDA inputs are the documents (D), the number of topics (K), and a set of hyperparameters. The hyperparameters are:

- k is the amount of topics that must be generated from the data,
- α is the influence on the topic distributions per document. A lower α value results in fewer topics per document,
- β is the affects of the distribution of terms per topic, Lower β value results in fewer terms per topic, as a result need to increase in the number of topics.

2.3. Query expansion

Query expansion is the method of adding the original query with additional words, which could help to obtain actual user intent [21]. Query expansion has been applied in many applications, such as question answering, multimedia information retrieval, information filtering, and cross-language information retrieval. Query expansion was the crucial part of this research by utilizing the use case relationship as an additional word to help the system understand the actual user intent.

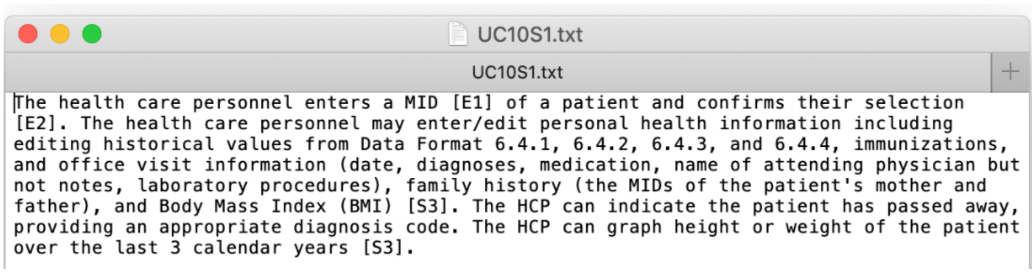
2.4. Use case scenario relationship model

Use case scenarios are sets of sentences that describe the step, data input, logic and sometimes data output of a feature in software. It is used by developers as a key to develop the specific feature of software. The terms that are used are usually high-level language. The term in the use case scenario is sometimes repeated and has the same meaning from the software engineer's perspective.

The use case relationship model is our original and novel approach to capture the indirect query from the user. This model build intended to enrich the query with additional words from other use case scenarios that were associated. Based on observations from the iTrust [17] project, the use case scenario has a relationship with other use case scenarios. This section will elaborate on the concepts and characteristics of use case relationships that might exist.

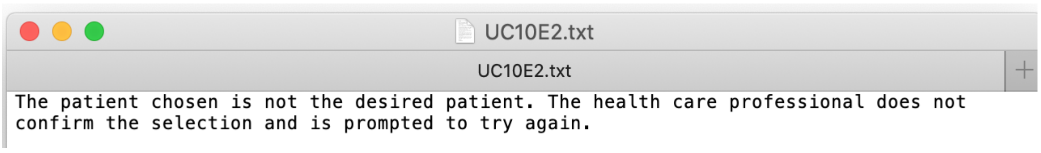
2.4.1. Inner association

The first concept was an inner association. An inner association is the kind of association that describes a use case scenario associated with another use case scenario with the same actor, but it is used as an alternative or exception. This concept was found based of our deep inspection on use case scenarios, which some use case scenarios were the alternative of others. The example of inner association were shown on Figures 1 to 3 (e.g., [E1] [E2] [S3]). The UCS 10 Scenario 1 have two alternative or exception on UCS 10 Error 2 and Error 1. The logical reason was the code of error exception of a function should be related with the main code of function.



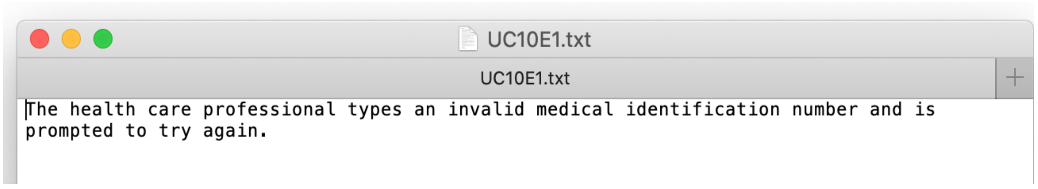
```
The health care personnel enters a MID [E1] of a patient and confirms their selection
[E2]. The health care personnel may enter/edit personal health information including
editing historical values from Data Format 6.4.1, 6.4.2, 6.4.3, and 6.4.4, immunizations,
and office visit information (date, diagnoses, medication, name of attending physician but
not notes, laboratory procedures), family history (the MIDs of the patient's mother and
father), and Body Mass Index (BMI) [S3]. The HCP can indicate the patient has passed away,
providing an appropriate diagnosis code. The HCP can graph height or weight of the patient
over the last 3 calendar years [S3].
```

Figure 1. UCS 10 scenario 1



```
The patient chosen is not the desired patient. The health care professional does not
confirm the selection and is prompted to try again.
```

Figure 2. UCS 10 error 2

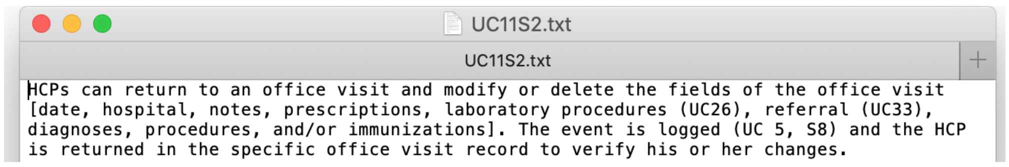


```
The health care professional types an invalid medical identification number and is
prompted to try again.
```

Figure 3. UCS 10 error 1

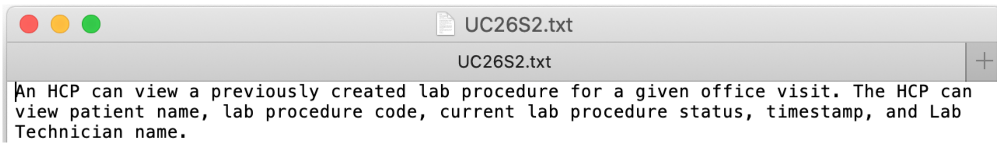
2.4.2. Outer association

The second concept was an outer association an outer association was the kind of association that described a use case scenario associated with other use case scenarios with different actors, but it was used as a reference. This concept was found after we inspect on the use case scenario which has specific tags such as "(UC26)" on it. The iTrust software analyst might want to give the mark that show the use case have reference with others. Based on data, as shown in Figures 4 and 5, the use case scenario 11 mentioned use case 26, which is marked with "(UC26)".



```
HCPs can return to an office visit and modify or delete the fields of the office visit
[date, hospital, notes, prescriptions, laboratory procedures (UC26), referral (UC33),
diagnoses, procedures, and/or immunizations]. The event is logged (UC 5, S8) and the HCP
is returned in the specific office visit record to verify his or her changes.
```

Figure 4. Use case scenario 11

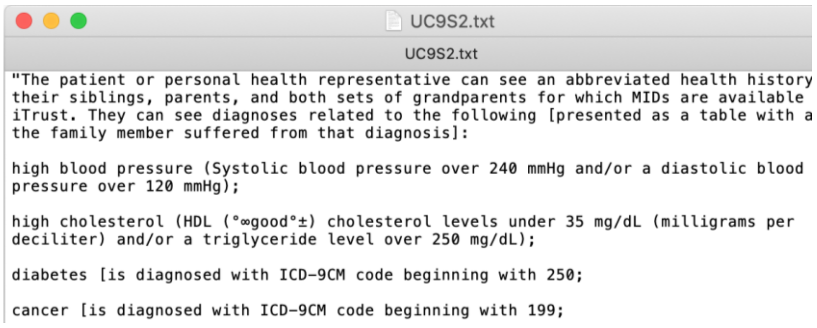


```
An HCP can view a previously created lab procedure for a given office visit. The HCP can
view patient name, lab procedure code, current lab procedure status, timestamp, and Lab
Technician name.
```

Figure 5. Use case scenario 26

2.4.3. Intratoken association

An intratoken association is the kind of association in which a token of a use case scenario has a relationship with semantic similarity meaning. For example, the keyword “Patient” might have semantic similarity with the keyword “blood pressure” (score 0.3750). Based on the dataset, UC10 S1 as shown in Figure 1 has a relationship with UC9S2 as shown in Figure 6.



```
"The patient or personal health representative can see an abbreviated health history
their siblings, parents, and both sets of grandparents for which MIDs are available
iTrust. They can see diagnoses related to the following [presented as a table with a
the family member suffered from that diagnosis]:

high blood pressure (Systolic blood pressure over 240 mmHg and/or a diastolic blood
pressure over 120 mmHg);

high cholesterol (HDL (°∞good°±) cholesterol levels under 35 mg/dL (milligrams per
deciliter) and/or a triglyceride level over 250 mg/dL);

diabetes [is diagnosed with ICD-9CM code beginning with 250;

cancer [is diagnosed with ICD-9CM code beginning with 199;
```

Figure 6. Use case scenario 9

3. RESEARCH METHOD

This section explains how our research framework finds the feature location based on query expansion using the use case relationship. There were five segments of our framework: dataset definition, modeling the topics of source codes, use case scenario relationship modeling, query expansion using use case relationships, and the evaluation process. All the structure shown on the Figure 7. The details will be explained in this section.

3.1. Dataset definition

The dataset we used was iTrust [17]. An iTrust is a Java-based Electronic Health Record system that was developed at North Caroline State University (NCSU) as a primary case study in a software engineering class. The version was version 19 (https://github.com/ncsu-csc326/iTrust/tree/v19/iTrust). The dataset contains approximately forty use cases mapped into 478 trace links of health record features such as personal health records, patients, diseases, safe drugs, visits, and lab procedures. The iTrust projects are equipped with complete data such as a use case, use case scenario, and codes. It also has a traceability link, which function as ground truth. It contains many files of source code (354 files). It also used by many researchers [22], [23].

The dataset was filtered into 20 use case scenarios that could be categorized into a seven kinds of use case scenarios Table 1. The selection of seven features was done based on the assumption that those features were the most common electronic health record. After the selection, the inner association, outer association and intratoken association of the use case scenario were defined manually by the researcher which described in detail on section 3.3. The trace links chosen were reduced into 102 trace links related to those 20. Source codes were also reduced into only 68 files since many trace links used the same files.

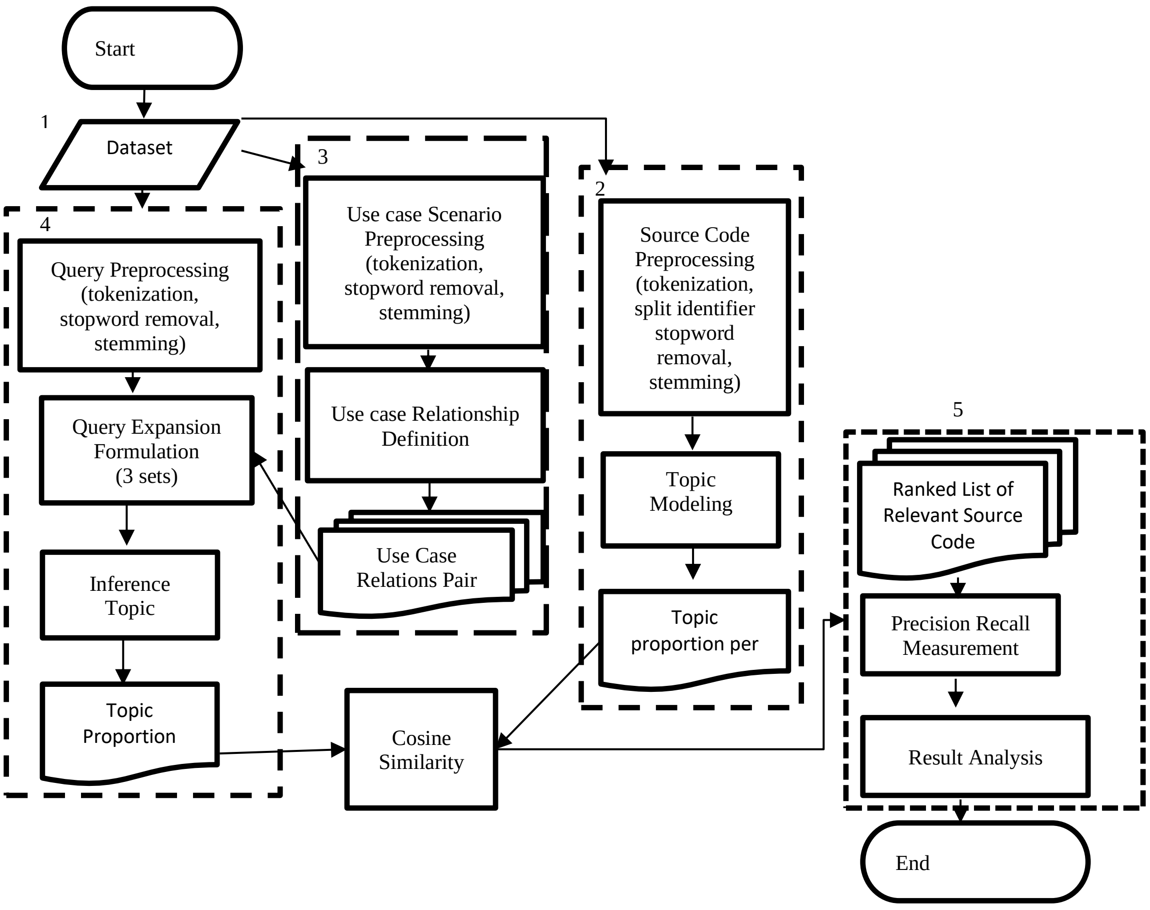

Start
1
Dataset
2
Source Code Preprocessing (tokenization, split identifier stopword removal, stemming)
Topic Modeling
Topic proportion per
3
Use case Scenario Preprocessing (tokenization, stopword removal, stemming)
Use case Relationship Definition
Use Case Relations Pair
4
Query Preprocessing (tokenization, stopword removal, stemming)
Query Expansion Formulation (3 sets)
Inference Topic
Topic Proportion
Cosine Similarity
5
Ranked List of Relevant Source Code
Precision Recall Measurement
Result Analysis
End

Figure 7. Research methodology

Table 1. Use case dataset

No	Use case name	Use case description
1	UC10E1.TXT	ENTER/EDIT PERSONAL HEALTH RECORDS
2	UC10E2.TXT	ENTER/EDIT PERSONAL HEALTH RECORDS
3	UC10S1.TXT	ENTER/EDIT PERSONAL HEALTH RECORDS
4	UC10S2.TXT	ENTER/EDIT PERSONAL HEALTH RECORDS
5	UC16.TXT	IDENTIFY RISK OF CHRONIC DISEASES
6	UC1E1.TXT	CREATE AND DISABLE PATIENTS
7	UC1S1.TXT	CREATE AND DISABLE PATIENTS
8	UC1S2.TXT	CREATE AND DISABLE PATIENTS
9	UC23S1.TXT	COMPREHENSIVE PATIENT REPORTS
10	UC23S3.TXT	COMPREHENSIVE PATIENT REPORTS
11	UC23S4.TXT	COMPREHENSIVE PATIENT REPORTS
12	UC26S1.TXT	VIEW/EDIT LAB PROCEDURE STATUS
13	UC26S2.TXT	VIEW/EDIT LAB PROCEDURE STATUS
14	UC26S3.TXT	VIEW/EDIT LAB PROCEDURE STATUS
15	UC26S4.TXT	VIEW/EDIT LAB PROCEDURE STATUS
16	UC28.TXT	VIEW PATIENTS
17	UC9S1.TXT	VIEW RECORDS
18	UC9S2.TXT	VIEW RECORDS
19	UC11S1.TXT	DOCUMENT OFFICE VISIT
20	UC11S2.TXT	DOCUMENT OFFICE VISIT

3.2. Modelling the topics of source codes

The source codes from the selected dataset were preprocessed using several subprocesses, such as tokenizing, stop word elimination, stemming and modeling their topics. The first subprocess was tokenizing, which was the process of splitting the source code into tokens. The methods include punctuations removal (.,'-_) using regex and split method/variable name which has camelCase format into token (e.g., "updateAllergies split into update allergies").

The next subprocess was stop word elimination and stemming [24]. To eliminate a stop word on the source codes, we picked the tokens that were too common English (i.e., “is, the, and”). The stemming was the subprocess that eliminated suffixes or prefixes of the source codes. It intended to determine the root form of the word. The stemming was done by using the porter algorithm.

The last subprocess was modeling the topics of source codes that had already been preprocessed. The research use a Mallet [25] which implement LDA to make model topics. The topic parameters variable of LDA were set to 5 topics that correlated with 7 kinds of use case descriptions. The iteration number parameter was set to 4,000. It produced several files, a model of topics, an inference file and topic proportion of files (68 files). It also produces keywords per topic that are used further as translator tokens. The files contain of topic proportion which used as a ranking recommendation in process of query comparison. As a result, Mallet created five topics with the top 15 keywords, as shown in Table 2.

Table 2. Top 15 keywords per topic

Total Topics: 5	Topic 0 Word & Frq.	Topic 1 Word & Frq.	Topic 2 Word & Frq.	Topic 3 Word & Frq.
0 long 4:50 2:14 1:6 0:6	patient 86	visit 66	mid 45	factor 67
1 add 3:22 0:18 2:6 1:3	bean 85	offic 57	user 20	risk 43
2 patient 0:86 4:78 3:28	famili 50	record 38	type 18	patient 28
3 bean 0:85 4:61 1:19	mid 41	form 32	log 16	add 22
4 form 1:32 2:2	pid 38	health 25	role 15	disea 19
5 valid 1:23 2:3	trust 36	ov 25	transact 15	checker 10
6 trust 0:36 1:18 2:6	member 36	valid 23	long 14	mid 7
7 mid 4:64 2:45 0:41	parent 34	id 23	password 14	factori 7
8 dao 0:29 1:12 2:5	allergi 31	bean 19	patient 12	health 6
9 empti 2:3	dao 29	trust 18	set 10	record 6
10 set 4:61 2:10 1:10	log 23	patient 17	db 10	current 6
11 pwd 2:2	add 18	log 15	param 9	ethnic 4
12 auth 2:3	fam 17	report 13	pstmt 9	arrai 4
13 user 2:20	grandpar 15	request 13	case 8	american 4
14 role 2:15	prescript 13	pid 13	factori 7	histori 3
15 random 2:4	db 11	dao 12	ad 7	famili 3

3.3. Use case scenario relationship modelling

The use case scenarios relationship modelling process were the most crucial parts of this research. The use case scenario has a relationship with others through our concept of inner association, outer association and intratoken associations. To implement the concept, we created a relational database to record the use case scenario associations Figure 8. The entities were use cases and tokens. The use cases were the entities to use to save the use case scenario data, while tokens entity was used to save the key/terms from use case scenario for intratoken association. The use cases have both inner and outer relations.

The inner relation and outer relation association were defined by inspecting the use case scenarios one by one manually. The inspection includes find specific tags on the use case scenarios. The tags [E1] represent the inner relation, while tag (UCxx) represents the outer relation, as shown in Figures 1 to 5. The pairs of use cases were saved into a table in the database as illustrates in Figures 9 and 10.

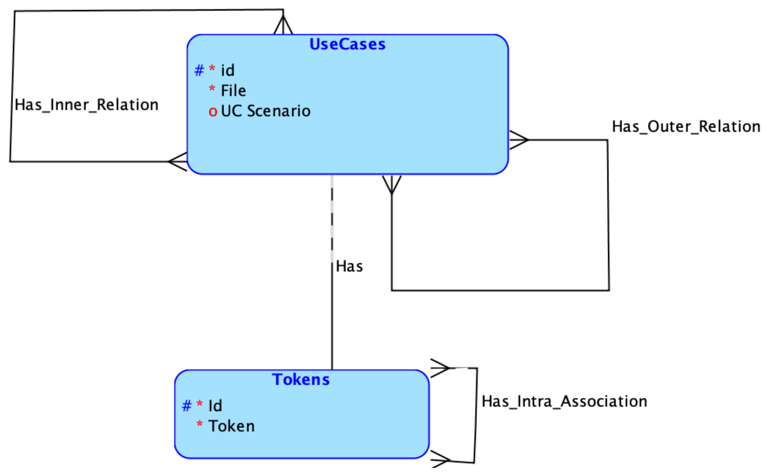
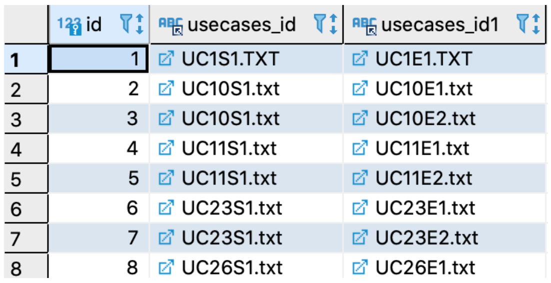
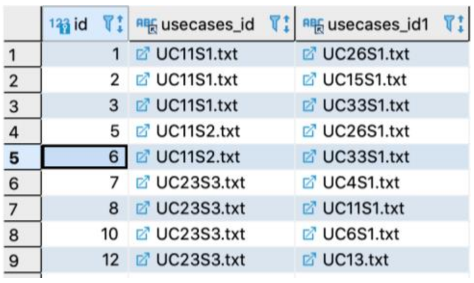


Figure 8. E-R model of use case relationship models



	id	usecases_id	usecases_id1
1	1	UC1S1.TXT	UC1E1.TXT
2	2	UC10S1.txt	UC10E1.txt
3	3	UC10S1.txt	UC10E2.txt
4	4	UC11S1.txt	UC11E1.txt
5	5	UC11S1.txt	UC11E2.txt
6	6	UC23S1.txt	UC23E1.txt
7	7	UC23S1.txt	UC23E2.txt
8	8	UC26S1.txt	UC26E1.txt

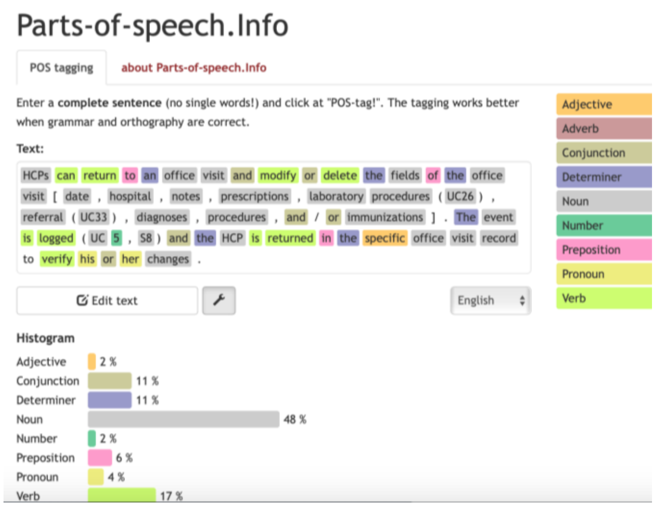
Figure 9. Inner association of use case scenarios



	id	usecases_id	usecases_id1
1	1	UC11S1.txt	UC26S1.txt
2	2	UC11S1.txt	UC15S1.txt
3	3	UC11S1.txt	UC33S1.txt
4	5	UC11S2.txt	UC26S1.txt
5	6	UC11S2.txt	UC33S1.txt
6	7	UC23S3.txt	UC4S1.txt
7	8	UC23S3.txt	UC11S1.txt
8	10	UC23S3.txt	UC6S1.txt
9	12	UC23S3.txt	UC13.txt

Figure 10. Outer association of use case scenarios

To define intratoken association, we perform several subprocesses. The first was the extraction of meaningful tokens from use case scenarios. The meaningful tokens were extracted using Post Tagger [26] (https://parts-of-speech.info) as illustrated in Figure 11. PostTagger [27] needs a complete sentence to determine the tag of words. Therefore, in this case, we used unpreprocessed use case scenarios to extract correct tags. Noun and verb only words were used as token association candidates since it could help reduce the number of words and increase the success rate [15], [16], [28]. As a result, meaningful tokens were saved on tables to be processed further Figure 12.



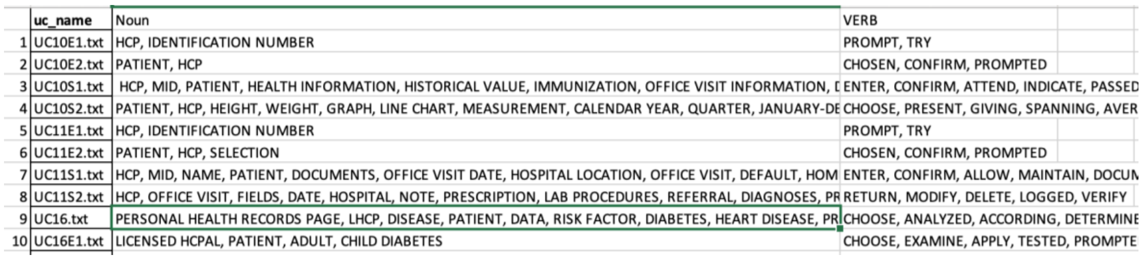
Enter a **complete sentence** (no single words!) and click at "POS-tag!". The tagging works better when grammar and orthography are correct.

Text:

HCPs can return to an office visit and modify or delete the fields of the office visit [date , hospital , notes , prescriptions , laboratory procedures (UC26) , referral (UC33) , diagnoses , procedures , and / or immunizations] . The event is logged (UC 5 , S8) and the HCP is returned in the specific office visit record to verify his or her changes .

Adjective	2 %
Conjunction	11 %
Determiner	11 %
Noun	48 %
Number	2 %
Preposition	6 %
Pronoun	4 %
Verb	17 %

Figure 11. Determine meaningful tokens using pos tag



	uc_name	Noun	VERB
1	UC10E1.txt	HCP, IDENTIFICATION NUMBER	PROMPT, TRY
2	UC10E2.txt	PATIENT, HCP	CHOSEN, CONFIRM, PROMPTED
3	UC10S1.txt	HCP, MID, PATIENT, HEALTH INFORMATION, HISTORICAL VALUE, IMMUNIZATION, OFFICE VISIT INFORMATION, [	ENTER, CONFIRM, ATTEND, INDICATE, PASSED
4	UC10S2.txt	PATIENT, HCP, HEIGHT, WEIGHT, GRAPH, LINE CHART, MEASUREMENT, CALENDAR YEAR, QUARTER, JANUARY-DE	CHOOSE, PRESENT, GIVING, SPANNING, AVER
5	UC11E1.txt	HCP, IDENTIFICATION NUMBER	PROMPT, TRY
6	UC11E2.txt	PATIENT, HCP, SELECTION	CHOSEN, CONFIRM, PROMPTED
7	UC11S1.txt	HCP, MID, NAME, PATIENT, DOCUMENTS, OFFICE VISIT DATE, HOSPITAL LOCATION, OFFICE VISIT, DEFAULT, HOM	ENTER, CONFIRM, ALLOW, MAINTAIN, DOCUN
8	UC11S2.txt	HCP, OFFICE VISIT, FIELDS, DATE, HOSPITAL, NOTE, PRESCRIPTION, LAB PROCEDURES, REFERRAL, DIAGNOSES, PR	RETURN, MODIFY, DELETE, LOGGED, VERIFY
9	UC16.txt	PERSONAL HEALTH RECORDS PAGE, LHCP, DISEASE, PATIENT, DATA, RISK FACTOR, DIABETES, HEART DISEASE, PR	CHOOSE, ANALYZED, ACCORDING, DETERMINE
10	UC16E1.txt	LICENSED HCPAL, PATIENT, ADULT, CHILD DIABETES	CHOOSE, EXAMINE, APPLY, TESTED, PROMPTE

Figure 12. Samples of meaningful tokens

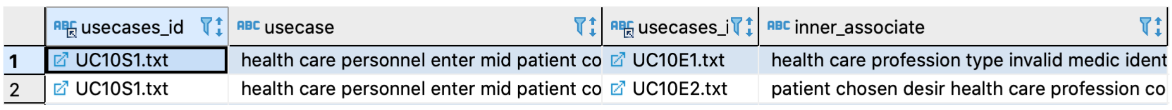
The second subprocess of intratoken association was created a matrix of word-based semantic similarity [29] to facilitate the intratoken association easily. All the words were compared one by one and calculated based on their semantic similarity. The words and similarity degree were saved to a table named the matrix of semantic similarity tokens. All similar words are used for expansion, which saves fields named intratoken association.

The results of the use case scenario relationship were the data of the inner association of the use case scenario, outer association of the use case scenario, and intra token association of the use case scenario. All association have saved on database to make experiment easier to do. Each relationship tested one by one for their performance of precision and recall.

3.4. Query expansion sets

The words in the use case scenario were used as the initial query to our information retrieval approach. All words were preprocessed include tokenization, elimination of stop word and stemming. These processes were intended to ensure that the remaining tokens were meaningful and in the root form of words.

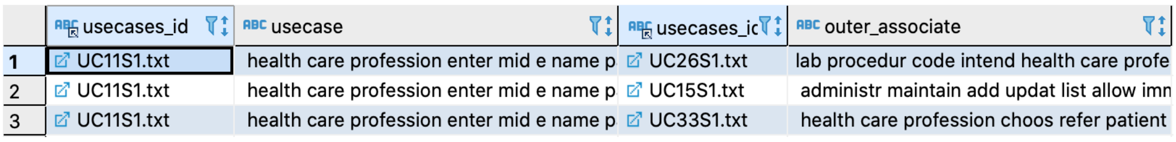
The first subprocess was query expansion using the first step of our novel use case scenario relationship called the inner relation association. It was done by finding the pairs of use case scenarios that comply with the rule given in section 2.4.1. For example, the query given was use case scenario 10 (UC10S1.txt). Based on the inner relation association pair in Figure 9, UC10S1 had pairs with both UC10E1 and UC10E2, so their tokens were included as query expansions of UC10S1. Figure 13 illustrated that the step to obtain the tokens was performed by applying a join query to produce the token pair of inner relation associations.



	usecases_id	usecase	usecases_i	inner_associate
1	UC10S1.txt	health care personnel enter mid patient co	UC10E1.txt	health care profession type invalid medic ident
2	UC10S1.txt	health care personnel enter mid patient co	UC10E2.txt	patient chosen desir health care profession co

Figure 13. The inner relation pairs of UC10S1

The second subprocess was query expansion using the second step of our novel use case scenario relationship called the outer relation association. It was done by finding the pairs of use case scenarios that comply with the rule given in section 2.4.2. For example, the query given was use case scenario 11 (UC11S1.txt). Based on the outer relation association pair Figure 10, UC11S1 had pairs with UC26S1, UC15S1, and UC33S1, so their tokens were included as query expansions of UC11S1. Figure 13 illustrated that the step to obtain the tokens was performed by applying a join query to produce the tokens pair of outer relation association.



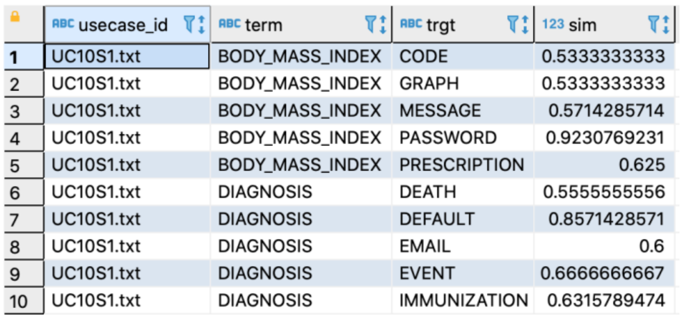
	usecases_id	usecase	usecases_ic	outer_associate
1	UC11S1.txt	health care profession enter mid e name p	UC26S1.txt	lab procedur code intend health care profe
2	UC11S1.txt	health care profession enter mid e name p	UC15S1.txt	administr maintain add updat list allow imn
3	UC11S1.txt	health care profession enter mid e name p	UC33S1.txt	health care profession choos refer patient

Figure 14. Outer relation pairs of UC11S1

The third subprocess was query expansion using the third step of our novel use case scenario relationship called the intra token association. It was done by finding the pairs of semantically similar tokens of use case scenarios that comply with the rule given in section 2.4.3. The tokens are extracted and put on the matrix of word-based semantic similarity. For example, the query given was use case scenario 10 (UC10S1.txt). Each token is compared against all tokens from the matrix similarity. Similar tokens with degree > 0.5 are used for query expansion. The step to obtain the tokens was performed by applying a join query to produce the token pair of intratoken associations, as illustrated in Figure 15. The Latent Dirichlet Allocation algorithm defines the topics unsupervised by iterating to give topics to both documents and tokens. At the end, the documents and the tokens are assigned to specific topics.

3.5. Evaluation process

The final process where the result evaluation and analysis process. The recommendation of source codes was generated by comparing topic proportion of query expansion against the topic proportion of all source code files. Each topic from the query was calculated to measure the Euclidian distance with the topic of each file using cosine similarity. The ranking of recommendation presented by sorting the cosine similarity the nearest to furthest. The threshold of similarity was set to 0.3. Precision and recall were employed to determine the success rate of our methods.



	usecase_id	term	trgt	sim
1	UC10S1.txt	BODY_MASS_INDEX	CODE	0.5333333333
2	UC10S1.txt	BODY_MASS_INDEX	GRAPH	0.5333333333
3	UC10S1.txt	BODY_MASS_INDEX	MESSAGE	0.5714285714
4	UC10S1.txt	BODY_MASS_INDEX	PASSWORD	0.9230769231
5	UC10S1.txt	BODY_MASS_INDEX	PRESCRIPTION	0.625
6	UC10S1.txt	DIAGNOSIS	DEATH	0.5555555556
7	UC10S1.txt	DIAGNOSIS	DEFAULT	0.8571428571
8	UC10S1.txt	DIAGNOSIS	EMAIL	0.6
9	UC10S1.txt	DIAGNOSIS	EVENT	0.6666666667
10	UC10S1.txt	DIAGNOSIS	IMMUNIZATION	0.6315789474

Figure 15. The intratoken association of UC10S1.txt

4. RESULTS AND DISCUSSION

The experiments were performed in several sets. The first experiment was queried using a use case scenario without expansion as baseline experiments. The second experiment used query expansion of the inner association relationship, which means that the use case scenario concatenates with the inner association token. The third was the experiment using query expansion of the outer association relationship, which means that the use case scenario concatenates with the outer association token. The fourth was the experiment using query expansion of the intratoken association relationship, which means that the use case scenario concatenates with the intratoken association relationship. The final experiment used the query compound of all elements, the use case scenarios, inner association token, outer association token, and intratoken association. This section discusses the details of the experiments.

4.1. Experiment without query expansion

The first experiment was performed by using all word from a use case scenario as the query without query expansion. It used as the baseline of the testing. The words were preprocessed using tokenize, stop word elimination, and stemming using the porter algorithm. The rest of the words are then put into the query of the research. The result is depicted in Table 3.

Table 3. Experiment results without query expansion

Query	Items Retrieved	Items relevant & items retrieved	Items relevant	Recall	Precision
UC1S1	9	3	5	60.0%	33.3%
UC9S1	55	8	8	100.0%	14.5%
UC9S2	12	3	3	100.0%	25.0%
UC10S1	19	7	15	46.7%	36.8%
UC10S2	18	0	1	0.0%	0.0%
UC10E1	29	0	2	0.0%	0.0%
UC10E2	37	0	2	0.0%	0.0%
UC11S1	56	6	6	100.0%	10.7%
UC11S2	21	5	6	83.3%	23.8%
UC16	12	6	8	75.0%	50.0%
UC23S1	43	2	2	100.0%	4.7%
UC23S3	30	12	26	46.2%	40.0%
UC23S4	30	1	2	50.0%	3.3%
UC26S1	34	2	2	100.0%	5.9%
UC26S2	40	3	6	50.0%	7.5%
UC26S3	43	2	4	50.0%	4.7%
UC26S4	37	1	1	100.0%	2.7%
UC28	16	1	3	33.3%	6.3%
	(Sum) 541	Sum (62)	(Sum) 102	(Avg) 60.8%	(Avg) 15.0%

The total number of items retrieved was 541 documents, and the total number of items relevant and retrieved was 62 documents of 102 documents relevant. The average recall was 60.8%, which means that the baseline approach could provide 60 documents out of 100 correct documents. The average precision was 15%, which means it could recommend 15 documents of 100 documents.

The best recall was 100%, which comes from several use case scenarios (UC9S1, UC9S2, UC11S1, UC23S1, UC26S1, UC26S4). The reason was those files used many words which quite technical, e.g., “immunization, diagnoses, and office visit.”, which could also be found on the source codes as implemented in the field of persistent files such as databases/tables.

The worst recall was 0, which appeared on UC10S2 and UC10E2. The reason was that it was implemented in few files of source code, and as a result, the item relevance became limited (1 & 2 documents only). The words on the use case scenario UC10S2 were "HCP choose height weight graph. presented chart chosen measurements patient spanning 3 calendar years data, averaged quarters (January-March, April-June, July September, October-December)". It is quite specific and directed and might not be shared among use case scenarios.

The best precision was 50%, which came from UC16. It was also the most ideal since the recall was quite superior (75%). The reason was that the sentences of UC16 contain balanced words on both abstract and detail (technical) topics. Another reason was that UC16 was implemented in some files (8); as a result, the number of relevant items became 8 documents. The words on UC16 were "Personal Health Records LHCP chooses chronic disease patient. data database analyzed risk factors disease determine exhibits risk factor. Risk factors for chronic diseases included diabetes type 1 and type 2 heart disease. chosen patient satisfies preconditions chosen chronic disease, the LHCP warning message patient exhibits risk factors. message display risk factors patients exhibit". It contains many words (e.g., personal health records, disease, patient, diabetes, type 1, type 2, and heart disease). shared among use case scenarios, which is why it could obtain the best results.

4.2. Experiment using query expansion based on the inner use case relationship

The second experiment was a query using token of use case scenario with expansion from the inner use case relationship. It was preprocessed using tokenize, stop word elimination, and stemming using the porter algorithm. The preprocessed tokens originating from the use case scenario were concatenated with additional tokens from the inner use case scenario to build a query for information retrieval. The result is depicted in Table 4.

Table 4. Experiment results using inner relationships as query expansion

Query	Items Retrieved	Items relevant & items retrieved	Items relevant	Recall	Precision
UC1S1	4	2	5	40.0%	50.0%
UC9S1	14	2	8	25.0%	14.3%
UC9S2	12	3	3	100.0%	25.0%
UC10S1	39	12	15	80.0%	30.8%
UC10S2	18	0	1	0.0%	0.0%
UC10E1	31	2	2	100.0%	6.5%
UC10E2	37	0	2	0.0%	0.0%
UC11S1	21	5	6	83.3%	23.8%
UC11S2	13	3	6	50.0%	23.1%
UC16	12	6	8	75.0%	50.0%
UC23S1	23	2	2	100.0%	8.7%
UC23S3	29	11	26	42.3%	37.9%
UC23S4	28	1	2	50.0%	3.6%
UC26S1	33	2	2	100.0%	6.1%
UC26S2	51	3	6	50.0%	5.9%
UC26S3	44	2	4	50.0%	4.5%
UC26S4	37	1	1	100.0%	2.7%
UC28	12	1	3	33.3%	8.3%
	(Sum) 458	(Sum) 58	(Sum) 102	(Avg) 59.9%	(Avg) 16.7%

The amount of items retrieved was reduced to 458 documents, and the amount of items relevant and retrieved was also decreased to 58 documents of 102 documents relevant. The average recall was 59.9%, which means that the baseline approach could provide 60 documents out of 100 correct documents. The average number of precisions was increased to 16,7%, which means it could recommend 17 documents of 100 documents.

The best recall was 100%, which comes from several use case scenarios (UC9S2, UC10E1, UC23S1, UC26S1, UC26S4). The reason was about the same, which those files used many words which quite technical, e.g., "immunization, diagnoses, and office visit", which could also be found on the source codes as implemented in the field of persistent files such as databases/tables.

The worst recall was also the same as 0, which appeared on UC10S2 and UC10E2. The reason was about the same, which it implemented in few files of source code, and as a result, the item relevance becomes limited (1 & 2 documents only). The words on the use case scenario UC10S2 were "HCP choose height weight graph. presented chart chosen measurements patient spanning 3 calendar years data, averaged quarters (January-March, April-June, July September, October-December)". It is quite specific and directed and might not be shared among use case scenarios.

The best precision was 50%, which came from UC16 and UC1S1. The reason for UC16 is still the same, which contains balanced words on both abstract and detail (technical) topics. UC1S1 had an intra extension from UC1E1, which could help capture additional tokens. The words on UC1S1 and UC1E1 were merged into "health care profession enter patient user iTrust medic record email provide assign mid secret key initial password person reset edit accord data format value default null appropriate number edit enter view secure question prompt enter editor correct format requires data field input match specific patient". The words also on stemmed form. It contains many words (e.g. patient, medical record, mid, key person, and password) shared among use case scenarios, which is why it could obtain the best results.

4.3. Experiment using query expansion based on the outer use case relationship

The third experiment was a query using tokens of use case scenario with expansion from the outer use case relationship. It were preprocessed using tokenize, stop word elimination, and stemming using the porter algorithm. The preprocessed tokens originating from the use case scenario were concatenated with additional tokens from the outer use case scenario to build a query for information retrieval. The result is depicted in Table 5.

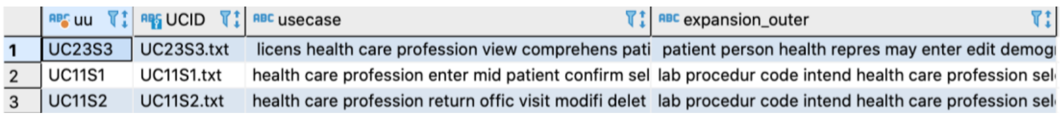
The amount of items retrieved was reduced to 479 of 541 documents, and the amount of items relevant and retrieved was also decreased to 55 documents of 102 documents relevant. The average recall was 60.7%, which means that the baseline approach could provide 60 documents out of 100 correct documents. The average number of precision was approximately the same at 15.4%, which means it could recommend 15 documents of 100 documents.

The best recall was 100%, which comes from several use case scenarios (UC9S2, UC10E1, UC23S1, UC26S1, UC26S4). The reason was about the same, which those files used many words which quite technical, e.g., "immunization, diagnoses, and office visit" which could also be found on the source codes as implemented in the field of persistent files such as databases/tables. The advantage of the outer layer had an impact on UC23S3, UC11S1, and UC11S2 as shown in Figure 16. The precision of both UC11S1 and UC11S2 increased to 19% and 27.8%, respectively, with a baseline precision for UC11S1 of only 10% and UC11S2 of 23.8%. Meanwhile, the precision of UC23S3 also increased to 8% from 4%.

The worst recall was also the same as 0, which appeared on UC10S2 and UC10E2. The reason was about the same, which it implemented in few files of source code, and as a result, the item relevance becomes limited (1 and 2 documents only). The words on the use case scenario UC10S2 were "HCP choose height weight graph. presented chart chosen measurements patient spanning 3 calendar years data, averaged quarters (January-March, April-June, July September, October-December)". It is quite specific and directed and might not be shared among use case scenarios.

Table 5. Experiment results using outer relationships as query expansion

Query	Items Retrieved	Items relevant and items retrieved	Items relevant	Recall	Precision
UC1S1	4	2	5	40.0%	50.0%
UC9S1	14	2	8	25.0%	14.3%
UC9S2	12	3	3	100.0%	25.0%
UC10S1	46	12	15	80.0%	26.1%
UC10S2	18	0	1	0.0%	0.0%
UC10E1	31	2	2	100.0%	6.5%
UC10E2	37	0	2	0.0%	0.0%
UC11S1	26	5	6	83.3%	19.2%
UC11S2	18	5	6	83.3%	27.8%
UC16	12	6	8	75.0%	50.0%
UC23S1	25	2	2	100.0%	8.0%
UC23S3	30	6	26	23.1%	20.0%
UC23S4	28	1	2	50.0%	3.6%
UC26S1	34	2	2	100.0%	5.9%
UC26S2	51	3	6	50.0%	5.9%
UC26S3	44	2	4	50.0%	4.5%
UC26S4	37	1	1	100.0%	2.7%
UC28	12	1	3	33.3%	8.3%
	(Sum) 479	(Sum) 55	(Sum) 102	(Avg) 60.7%	(Avg) 15.4%



	uu	UCID	usecase	expansion_outer
1	UC23S3	UC23S3.txt	licens health care profession view comprehens pati	patient person health repres may enter edit demogı
2	UC11S1	UC11S1.txt	health care profession enter mid patient confirm sel	lab procedur code intend health care profession sel
3	UC11S2	UC11S2.txt	health care profession return offic visit modifi delet	lab procedur code intend health care profession sel

Figure 16. Data of outer relationship

The best precision on this phased was 50%, which came from UC16 and UC1S1. After we inspect the data, we found that the UC16 contains balanced words on both abstract and detail (technical) topics. This results about the same with the previous phase (non query expansion and inner use case relationship). The outer use case precision rate mostly underperforms against inner relationship, except on UC11S2 which 4% better. The reason was the token related with it could extent the search result.

4.4. Experiment using query expansion based intratoken use case relationship

The fourth experiment was a query using tokens of use case scenario with expansion from the intratoken use case relationship. It was preprocessed using tokenize, stop word elimination, and stemming using the porter algorithm. The preprocessed tokens originating from the use case scenario were concatenated with additional tokens from the intratoken use case scenario to build a query for information retrieval. The result is depicted in Table 6.

The amount of items retrieved was reduced to 475 of 541 documents, and the amount of items relevant and retrieved was also decreased to 53 documents of 102 documents relevant. The average recall was 59.3%, which means that the baseline approach could provide 59 documents out of 100 correct documents. The average number of precision was approximately the same at 15,5%, which means it could recommend 15 documents of 100 documents.

The best recall was 100%, which comes from several use case scenarios (UC9S2, UC10E1, UC23S1, UC26S1). The reason was about the same, which those files used many words which quite technical, e.g., “immunization, diagnoses, and office visit.”, which could also be found on the source codes as implemented in the field of persistent files such as databases/tables.

The advantage of the intratoken had an impact on UC10S2 and UC10E1. The precision of both UC10S2 and UC10E1 increased to 4.8% and 6.7%, respectively, and the baseline precision for both UC10S2 and UC10E1 was 0%. The precision of UC23S1 also increases to 8% from previous precision (4.7%).

The worst recall was also the same as 0, which appeared on UC10S2 and UC10E2. The reason was about the same, which it implemented in few files of source code, and as a result, the item relevance becomes limited (1 & 2 documents only). The words on the use case scenario UC10S2 were “HCP choose height weight graph. presented chart chosen measurements patient spanning 3 calendar years data, averaged quarters (January-March, April-June, July September, October-December)”. It is quite specific and directed and might not be shared among use case scenarios.

The best precision was 50%, which came from UC16 and UC1S1. The reason for UC16 is still the same, which contains balanced words on both abstract and detail (technical) topics. This results about the same with the previous phase (non query expansion and inner use case relationship). The outer use case precision rate mostly underperforms against inner relationship, except on UC11S2 which 4% better. The reason was the token related with it could extent the search result.

Table 6. Experiment results using intratoken relationships as query expansion

Query	Items Retrieved	Items relevant & items retrieved	Items relevant	Recall	Precision
UC1S1	4	2	5	40.0%	50.0%
UC9S1	23	3	8	37.5%	13.0%
UC9S2	18	3	3	100.0%	16.7%
UC10S1	35	11	15	73.3%	31.4%
UC10S2	21	1	1	100.0%	4.8%
UC10E1	30	2	2	100.0%	6.7%
UC10E2	30	0	2	0.0%	0.0%
UC11S1	17	3	6	50.0%	17.6%
UC11S2	16	3	6	50.0%	18.8%
UC16	12	6	8	75.0%	50.0%
UC23S1	25	2	2	100.0%	8.0%
UC23S3	32	11	26	42.3%	34.4%
UC23S4	29	1	2	50.0%	3.4%
UC26S1	45	2	2	100.0%	4.4%
UC26S2	56	4	6	66.7%	7.1%
UC26S3	44	2	4	50.0%	4.5%
UC26S4	26	0	1	0.0%	0.0%
UC28	12	1	3	33.3%	8.3%
	(Sum) 475	(Sum) 57	(Sum) 102	(Avg) 59.3%	(Avg) 15.5%

4.5. Experiment using query expansion-based compound of all elements

The fifth experiment was query using tokens of use case scenario with expansion from the compound of all elements from the inner, outer and intratoken use case relationships. It was preprocessed using tokenize, stop word elimination, and stemming using the porter algorithm. The preprocessed tokens originating from the use case scenario were concatenated with additional tokens from the compound of all element token use case scenarios to build a query for information retrieval. The result is depicted in Table 7.

The amount of items retrieved increased to 604 of 541 documents, and the amount of items relevant and retrieved also decreased to 62 documents of 102 documents relevant. The average recall increase was 68.3%, which means that the baseline approach could provide 68 documents out of 100 correct documents. Unfortunately, the average precision dropped to 10%, which means that it could recommend 10 documents of 100 documents.

The best recall was 100%, which comes from several use case scenarios (UC9S2, UC10S2, UC10E1, UC11S1, UC11S2, UC23S1, UC26S1). The reason was the increase in words due to the impact of all association tokens on the query. The worst recall was also the same as 0, which appeared on UC10E2. The reason was about the same, which it implemented in few files of source code, and as a result, the item relevance becomes limited (1 & 2 documents only).

The best precision was 33%, which came from UC1S1. The reason for this is that UC1S1 contains balanced words on both abstract and detail (technical) topics. UC16 no longer has the best precision since it has many additional words for query expansion. The advantage for compounds of all elements was the increase in recall. Most documents were returned, and only UC10E2 did not return correct recommendations (recall 0), while the precision was reduced since the research produced many documents.

Table 7. Experiment using the compound of all element token relationships as query expansion

Query	Items Retrieved	Items relevant and items retrieved	Items relevant	Recall	Precision
UC1S1	21	3	5	60.0%	14.3%
UC9S1	26	4	8	50.0%	15.4%
UC9S2	19	3	3	100.0%	15.8%
UC10S1	36	12	15	80.0%	33.3%
UC10S2	21	1	1	100.0%	4.8%
UC10E1	30	2	2	100.0%	6.7%
UC10E2	30	0	2	0.0%	0.0%
UC11S1	57	6	6	100.0%	10.5%
UC11S2	57	6	6	100.0%	10.5%
UC16	24	7	8	87.5%	29.2%
UC23S1	23	2	2	100.0%	8.7%
UC23S3	28	5	26	19.2%	17.9%
UC23S4	29	1	2	50.0%	3.4%
UC26S1	40	2	2	100.0%	5.0%
UC26S2	56	4	6	66.7%	7.1%
UC26S3	44	2	4	50.0%	4.5%
UC26S4	26	0	1	0.0%	0.0%
UC28	37	2	3	66.7%	5.4%
	(Sum) 604	(Sum) 62	(Sum) 102	(Avg) 68.3%	(Avg) 10.7%

4.6. Result analysis and comparison

The sets of experiments have been performed and produce several results. Each of the results was compared to each other to measure how effective our methods were at finding feature locations. The chart is shown in Figure 17. Based on the chart, the best recall was the experiments using compound off all relationships (inner, outer, and intratoken). The main reason was that the number of tokens was huge since all use case relationships are included here. The compound produces less precision since it takes all tokens, which impacts the increasing number of documents as a dividing factor in precision measurement.

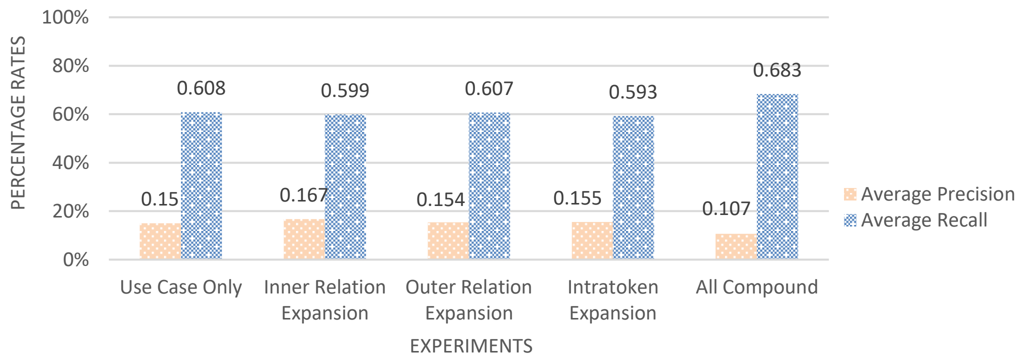


Figure 17. Comparison of average precision and recall

The best precision was the experiment using the inner relation. The worst recall was the experiment using intratokens, and the worst precision was all compounds. The Inner performs with the best precision because many use cases have inner relations among them, and the total tokens related were higher than the outer and intratokens.

5. CONCLUSION

This research introduces the novel concept of the use case relationship. It includes inner association, outer association, and intratoken association. The use case relationship was implemented and tested on the topic modeled source using the LDA algorithm. Query expansion based on inner, outer and intratoken associations was tested to find feature locations. The precision and recall rates were used to measure the success of the approach.

The best precision rate was 50% found in UC16, which contained tokens that were balanced on both the abstract side and technical side. The best recall was 100%, which was found in several use case scenarios implemented in a few files. The best average precision rate was 16.7%, which was found in inner association experiments. The inner association could help attract more tokens among other associations (outer, intratoken), which made average precision better than baseline (use case without expansion). The best average recall rate was 68.3% on all compound experiments since it contains all expansion tokens.

In the future, we plan to extend the methodology on the source code processing side by applying some structural exploration. The association among identifiers, methods, and comments could also be arranged on some data modeling to gain better precision and recall in feature location. The additional weighting of source code elements might also beneficial as an alternative methodology.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Education, Culture, Research, and Technology of Indonesia through Doctoral Dissertation Research (*Penelitian Disertasi Doktor*) under grant no. 1417/PKS/ITS/2022.

REFERENCES

[1] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "Measuring program comprehension: a large-scale field study with professionals," *IEEE Transactions on Software Engineering*, vol. 44, no. 10, pp. 951–976, 2018, doi: 10.1109/TSE.2017.2734091.

[2] A. Razzaq, A. Wasala, C. Exton, and J. Buckley, "The state of empirical evaluation in static feature location," *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 1, pp. 1–58, Jan. 2019, doi: 10.1145/3280988.

[3] X. Gu, H. Zhang, and S. Kim, "Deep code search," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 933–944. doi: 10.1145/3180155.3180167.

[4] A. Razzaq, A. Le Gear, C. Exton, and J. Buckley, "An empirical assessment of baseline feature location techniques," *Empirical Software Engineering*, vol. 25, no. 1, pp. 266–321, Jan. 2020, doi: 10.1007/s10664-019-09734-5.

[5] S. Sachdev, H. Li, S. Luan, S. Kim, K. Sen, and S. Chandra, "Retrieval on source code: a neural code search," in *Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, Jun. 2018, pp. 31–41. doi: 10.1145/3211346.3211353.

[6] R. Gharibi, A. H. Rasekh, M. H. Sadreddini, and S. M. Fakhrahmad, "Leveraging textual properties of bug reports to localize relevant source files," *Information Processing & Management*, vol. 54, no. 6, pp. 1058–1076, Nov. 2018, doi: 10.1016/j.ipm.2018.07.004.

[7] B. P. Eddy, N. A. Kraft, and J. Gray, "Impact of structural weighting on a latent dirichlet allocation-based feature location technique," *Journal of Software: Evolution and Process*, vol. 30, no. 1, Jan. 2018, doi: 10.1002/smr.1892.

[8] F. Perez, T. Ziadi, and C. Cetina, "Utilizing automatic query reformulations as genetic operations to improve feature location in software models," *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 713–731, Feb. 2022, doi: 10.1109/TSE.2020.3000520.

[9] R. K. Saha, M. Lease, S. Khurshid, and D. E. Perry, "Improving bug localization using structured information retrieval," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. 2013, pp. 345–355. doi: 10.1109/ASE.2013.6693093.

[10] A. Panichella *et al.*, "When and how using structural information to improve IR-based traceability recovery," in *2013 17th European Conference on Software Maintenance and Reengineering*, Mar. 2013, pp. 199–208. doi: 10.1109/CSMR.2013.29.

[11] J. Krüger, M. Mukelabai, W. Gu, H. Shen, R. Hebig, and T. Berger, "Where is my feature and what is it about? a case study on recovering feature facets," *Journal of Systems and Software*, vol. 152, pp. 239–253, Jun. 2019, doi: 10.1016/j.jss.2019.01.057.

[12] J. Echeverría, J. Font, F. Pérez, and C. Cetina, "Comparison of search strategies for feature location in software models," *Journal of Systems and Software*, vol. 181, 2021, doi: 10.1016/j.jss.2021.111037.

[13] A. Razzaq, A. Ventresque, R. Koschke, A. De Lucia, and J. Buckley, "The effect of feature characteristics on the performance of feature location techniques," *IEEE Transactions on Software Engineering*, vol. 48, no. 6, pp. 2066–2085, Jun. 2022, doi: 10.1109/TSE.2021.3049735.

[14] M. B. Silva, "Summary for policymakers," in *Climate Change 2013 – The Physical Science Basis*, vol. 1, no. 9, Cambridge University Press, 2014, pp. 1–30. doi: 10.1017/CBO9781107415324.004.

[15] A. Arwan, S. Rochimah, and C. Fatichah, "A comparison study : the effect of nouns and verbs in finding feature location," in *7th International Conference on Sustainable Information Engineering and Technology 2022*, Nov. 2022, vol. 1, no. 1, pp. 310–315. doi: 10.1145/3568231.3568282.

[16] A. Arwan, S. Rochimah, and C. Fatichah, "Query expansion based on user requirements clustering for finding feature location," in *2022 6th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 2022, pp. 1–5. doi: 10.1109/ICITISEE57756.2022.10057893.

[17] A. Meneely, B. Smith, and L. Williams, "iTrust electronic health care system : a case study," *Software and Systems Traceability*, pp. 1–16, 2009.

[18] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.

[19] A. Qusef, G. Bavota, R. Oliveto, A. De Lucia, and D. Binkley, "Recovering test-to-code traceability using slicing and textual analysis," *Journal of Systems and Software*, vol. 88, no. 1, pp. 147–168, Feb. 2014, doi: 10.1016/j.jss.2013.10.019.

[20] D. M. Blei, A. Y. Ng, and M. T. Jordan, "Latent dirichlet allocation," *Advances in Neural Information Processing Systems*, vol. 3, pp. 993–1022, 2002.

[21] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *ACM Computing Surveys*, vol. 44, no. 1, pp. 1–50, Jan. 2012, doi: 10.1145/2071389.2071390.

[22] A. Mahmoud and N. Niu, "Source code indexing for automated tracing," in *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, 2011, pp. 3–9. doi: 10.1145/1987856.1987859.

[23] N. Ali, Y.-G. Gueheneuc, and G. Antoniol, "Requirements traceability for object oriented systems by partitioning source code," in *2011 18th Working Conference on Reverse Engineering*, 2011, pp. 45–54. doi: 10.1109/WCRE.2011.16.

[24] M. F. Porter, R. Boulton, and A. Macfarlane, "The English (porter2) stemming algorithm," *Retrieved*, 2002. http://snowball.tartarus.org/algorithms/english/stemmer.html (accessed Jan. 12, 2022).

[25] A. K. McCallum, "MALLET: a machine learning for language toolkit." 2002. http://www.cs.umass.edu/~mccallum/mallet (accessed Dec 20, 2022).

[26] K. Toutanova and C. D. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics* -, 2000, vol. 13, pp. 63–70. doi: 10.3115/1117794.1117802.

[27] "Apache OpenNLP: Toolkit for the processing of natural language text," *Apache*. https://opennlp.apache.org (accessed Jan. 01, 2022).

[28] S. Zamani, S. P. Lee, R. Shokripour, and J. Anvik, "A noun-based approach to feature location using time-aware term-weighting," *Information and Software Technology*, vol. 56, no. 8, pp. 991–1011, 2014, doi: 10.1016/j.infsof.2014.03.007.

[29] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* -, 1994, pp. 133–138. doi: 10.3115/981732.981751.

BIOGRAPHIES OF AUTHORS

Achmad Arwan received an M.S. degree in informatics engineering from Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2015. He is currently pursuing a doctoral degree in informatics engineering Institut Teknologi Sepuluh Nopember. His research interest includes software mining repository, database, and software development. He can be contacted at email arwan@ub.ac.id.

Siti Rochimah received her Ph.D. degree in software engineering from Universiti Teknologi Malaysia in 2010. Currently, she is the head of the Software Engineering laboratory with the Department of Informatics, Institut Teknologi Sepuluh Nopember. She authored and coauthored more than 50 articles related to software engineering. Her research interests include software quality, software traceability and software testing. She can be contacted at email siti@if.its.ac.id.

Chastine Fatichah received her Ph.D. from the Tokyo Institute of Technology, Japan, in 2012. She is currently full professor at the Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Her research interests focus on artificial intelligence, data mining, and image processing. She can be contacted at email chastine@if.its.ac.id.