The Rules of the Intelligent Software Analysis Association

Yousif Hardan Sulaiman
Al-Maaref University College, Iraq-Al-Ramadi

Abstract

This paper presents the concept of association rules through the intellectual analysis with the aim of bug fixing. Finding the association rules allows determining the relations or connection between the specified values of categorical variables in large databases. This problem is often met in many projects on Data mining (knowledge discovery in databases process). Such research methods have many applications in many fields of business and research – starting from the consumer demand analysis or human capital management and up to the history of language. The association rules definition method is based on three statistic indicators, being calculated for pairs of objects (events, which happen simultaneously) in the data, named "Cause" and "Sequence"- Support (how often "Cause" and "Sequence" objects in the data are met, Trust (probability of the fact how often both objects "Cause" and "Sequence" among the data are met together) and Correlation (support for "Cause" and "Sequence", divided into square root of the support result for "Case" and support for 'Sequence"). The aim of the method is finding the type of the links "If "Cause", then "Sequence". The main problem of the conformity detection methods is the enumeration of possibilities within the acceptable time. The known methods either artificially restrict such enumeration or build the decision trees, having principal search efficiency restrictions of "If-then" rule. Other problems are connected with the fact, that known association rules searching technique do not support the generalization function of the found rules and the optimal composition search function for such rules. A successful solution to the indicated problems can make a subject of new competitive developments.

Keywords: *data mining, association rules, software, Apriori algorithm, evolutionary software.*

Introduction

The most popular application for the development of association rules is the analysis of market basket. On the basis of sales transactions often templates are usually viewed and come back as unification precepts. One of the prevalent examples the shoes with socks are frequently for sale along with. Such data is the price for cross-selling, which increases the total volume of company sales. Tool ROSE (Reengineering Software Evolution) makes the same for software developers (SW). He is searching templates in the story of versions and represents related objects in representation next to the original code. However, ROSE's goals are not to increase the total number of changes by developers, but simplifying navigation on the original code and avoiding of errors because of the lack of updates [1-5].

In this article methods, used by ROSE for confirming of its functionality are analyzed. We start from the construct of unification precepts and let us display how ROSE utilizes them. Show general method of intellectual analysis, named Priory algorithm, and then present improved algorithm of intellectual analysis for ROSE.

1. Problem area analysis

1.1. Pre-processing of the data.

Extraction of data from CSV has lit very well and many tools are available for free. In [6] the tool Soft Change, which extracts and pools the data from CVS databases and monitors errors is presented [7]. In [8] BLOOF system, which extracts data from CVS journal file into the database and visualizes the evolution of software with the use of indicators is developed [9].

In [10] it is indicated how a database of release story, connecting data from CVS and BUGZILLA should be filed, the algorithm of mergers version detection is proposed. In [11] the same authors connected their approach with specialties of HIPICAT, which considers few data sources. They combine information from the lists of CVS, BUGZILLA scatter and developers, using text similarity.

A transaction lifting was used by many approaches, but it was not described in details anywhere. In [13] fixed time frames were used for this. Up to now only a few approaches [13] considered minor changes In [14] relations between classes were analyzed In [15] it is shown how analysis of the origin for determination of the merge and functions division should be used.

Nevertheless, purification of data is often ignored and still many possibilities for improvement remain.

1.2. Development of the software.

In [16] Ying tool, which uses the rule of rules integration in archives of CVS versions is developed. In contrast to ROSE, Ying can analyze only the level of files, but not finite main points and does not support analysis on-the-spot. In [17] data mining in the initial code of programming libraries is used for determination of reuse pattern in the form of association.

To guide programmers the number of tools used log messages text similarity [18] or program code [19]. In [12] these results are improved at the expense of usage of such sources as post archives and online documentation.

Unlike ROSE, all these tools are focused on high recall, but not on a high accuracy, as well as on relation between files or cells but not between indistinct objects.

2. Association rules

Aassociation rules present template. Let us say, the succeeding rule presents some template {f Keys[], init Defaults(). Plug in. properties} in ECLIPSE story versions:

Shift (f Keys[])⇒shift(init Defaults())∧
∧shift(plug in. properties) [0.888]

For this rule two different interpretations exist:

1. Descriptive interpretation goes back to the past: every time, when the client shifted the field {f Keys[], it also shifted in it Defaults()method and file plug-in properties with certainty in 88,8%.

2. On the contrary, intellectual interpretation (as ROSE is used) is guided to after time : presently this principle means that every time when the client change field f Keys[], it as well must edit the mode in it Defaults() and data set plug-in internals. Here «must» indicates that the precept is based on the experiment (with credibility in 88,8%) and it is not an absolute truth; Symbol «⇒», in such a way should not be read as logical meaning, which is always performed.

As it was adverted above, the precepts have probability exegesis, founded on the quantity of proofs in activities from where they were obtained. This quantity of proofs is defined by three gauges:

Quantity. Quantity (or frequency) define the quantity of transactions, from which the rule was obtained. Let's assume, that f Keys[] was changed in 9 transactions. From these 9 transactions 8 too comprised the change as a method in it Defaults(), as well as file plug-in properties. The quantity for the aforementioned rule is equal to 8.

Support. Support connects the rule frequency with a common quantity of transactions. Because of the fact that ECLIPSE has 44786 transactions, support composes 8/44786 = 0,00018.

Confidence. Determines frequency in sequences, if left part of the rule is met. In the aforementioned rule, the sequence of initial Defaults() change is applied in 8 and 9 transactions with fKeys[]. Consequently, the statement for abovementioned rule is 8/9 = 0,888.

formally determination of unification precepts we accept:

Let's $I=\{i_1,...,i_n\}$ - is a set of all elements, recognized during preprocessing. Review, which is an element of the essence e, joint, for example, with action, changes the object e. Let's D - data, related to the problem, i.e. set of all transactions, where each transaction T represents the set of elements, such, that $T\subseteq I$.

The rule of association is a magnification of the shape $A\Rightarrow B$, where $A\subset I$, $B\subset I$ and $A\cap B=\varnothing$. A previous precept is A, and investigation B. Ordinarily A and B are connections, but can use any proposal. But, we will concentrate on alliances. Pro forma we determine the frequency, aid, and credibility of the unification precept in the following way:

- a frequency of setup X in pertinent for data transmission D is determined as

$$frequency_D(X)=|\{T|T\in D,\ X\subseteq T\}|.$$

A frequency of unification rules $A\Rightarrow B$ in relevant for data transmission D is determined as:

$$frequencyD(A\Rightarrow B)=frequency(A\cup B).$$

- support of the kit X correspondent problems D is determined as:

$$support_D(X)=\frac{frequency_D(X)}{|D|}=P(X).$$

The aid of the association precepts $A\Rightarrow B$ in data D, correspondent to problems is determined as:

- satisfaction of unification precept $A \Rightarrow B$ in data D **,** correspondent to problems is determined as:

$$confidence_D(A \Rightarrow B) = \frac{frequency_D(A \cup B)}{frequency_D(A)} = P(B|A).$$

We omit relevant data for the problem D, if they are known in the context or not actual. The abbreviated designation $r[s;c]$ indicates a precept r c $s = support(r)$ and $c = confidence(r)$.

For the kit of elements I, there is more probable precepts exit: every of $2^{|I|}$ templates promotes to one or more rules. In such a way, for reducing the general rules number the threshold values are used $support(min\_supp)$ and $confidence(min\_conf)$. The rule r is named powerful then and only then, when $support(r) \geq min\_supp$ and $confidence(r) \geq min\_conf$.

It is obvious that the reference riffle can take the place of a frequency riffle:

$$min\_freq = dmin\_supp \cdot |D| \neg.$$

ROSE utilizes frequency vice of aid for two causes:

1. The frequency for the developers is understood. Meanings of support, such as 0.00018, don't give an opportunity to understand if this meaning high or low. On the contrary, correspondent frequency 8 explicitly expresses the importance of the rule.

2. Frequency permit to compare miscellaneous projects. It is impossible to reuse reference riffle for other projects. Descry ECLIPSE with the sum of 44786 deals and JEDIT – totally 1905 deals. Utilization min_supp=0.0001, we conduct in ECLIPSE with min_freq=10, but in JEDIT alone with min_freq=1. Usage of such low-frequency riffles is not sensible.

Reliability is an indicator for distinctness. Confirmation of the rule may be a fallacy, as shown below:

	B	$\bar{B}$	
A	15%	10%	25%
$\bar{A}$	65%	10%	75%
	80%	20%	100%

Let's consider a rule $A \Rightarrow B$ with high probability degree

$$p(B/A) = \frac{p(A \cap B)}{p(A)} = \frac{0,15}{0,25} = 0,60.$$

This rule is misleading because has a high probability, but appearance A actually reduces the probability B of $p(B)$=0,80 to $p(B/A)$ =0,60. This supervision led to the development of unification rules called correlation analysis. It is not important for ROSE if the rule is false or not. In given below example B changed on 60% after the change A and that is why remains important for developers.

Meaning of precept support is a gauge of its statistic relevance. If the as follows conditions are met: A and B, probably, are independent, and their joint appearance in transactions happens by chance:

$$support(A \Rightarrow B) \approx support(A) \cdot support(B).$$

However, it is not related to the as follows circs:

$$support(A \Rightarrow B) ? \ support(A) \cdot support(B). \quad (1)$$

This term can be transferred into the equation, based on the assurance in the rule:

$$support(A \Rightarrow B) ? \ support(A) \cdot support(B).$$

$$\frac{support(A \Rightarrow B)}{support(A)} ? \ \cdot support(B).$$

$$\frac{frequency(A \Rightarrow B) \cdot |D|}{frequency(A) \cdot |D|} = support(B).$$

$$confidence(A \Rightarrow B)? \ \ support(B). \qquad (2)$$

However, we can define the higher boundary for $support(B)$. Obviously, that support is maximal for the singleton B.

In most cases, this estimation is significantly lower than 10%. In such a way, using probability threshold not less than 10%, ROSE is located in the safe zone and can delay significance verification. For projects rated higher than 10% ROSE, most likely, will recommend the objects, which are not statistical considerable, but still are acquitted (for example, TODO.txt for JEDIT).

As soon as programmer starts inserting changes, ROSE client proposes possible further changes. This is performed in the way of correspondence rules application. generally, two concepts of correspondence matching rules:

Low correspondence. The rule $A \Rightarrow B$ corresponds to the set of elements Σ (for example, changed objects), if the antecedent is a subset Σ, then $A \subseteq \Sigma$.

Strong correspondence. The rule corresponds to the set of elements Σ if this totality is equaled bygone of the rule, i.e. this precept $\Sigma \Rightarrow B$.

Either concepts antecedent of the precept is satisfied, but alone for powerful correspondence, it is executed accurately. We consider the kit of elements Σ as context, where ROSE gives recommendations. Remind, that element is a tactic, for example, change or essence.

Fig. 1. Example for preference ECLIPSE

Accounting of weak correspondence rules is not reasonable because it passes by the cutoffs of support and probability. Let's assume that we have three functions $f()$, $g()$ and $h()$. Functions $g()$ and $h()$ excludes each other. In such a way, the powerful rule $f() \wedge g() \Rightarrow h()$ does not exist, because it does not have aid. The client shift $f()$ and $g()$. utilization low correspondence, we consider the rule $f() \Rightarrow h()$ and spuriously advise $h()$, expelled by the appearance $g()$. In such a way, ROSE utilizes alone powerful rules and powerful correspondence.

How does ROSE calculate assertions? The kit of assertions for context Σ and kit of precepts R is determined as the integration of the sequences of all correspondence precepts:

$$apply_R(\Sigma) = \bigcup_{(\Sigma \Rightarrow B) \in R} B.$$

Let's suppose, that the problem of a coder is to deepen ECLIPSE with a recent precedence. Commonly precedence composed GUI details, Value for reduction and description (Fig. 1). Let programmer deepened set of lattice points *fKeys*[] in a file. ComparePreferencePage.java. In such a way, a situation Σ_1 is the following:

$$\Sigma_1 = \{change(fKeys\ [])\}.$$

ROSE contains many appropriate precepts for this context; One of them is r:

shift (fKeys[]) => shift(initDefaults ())^
^ shift (plugin.properties)[8;0,888].

Utilization the precept r in this context Σ_1, ROSE proposes investigation r:

$$apply_{\{r\}}(\Sigma_1) = \{change(initDefaults()), change(plugin.properties)\}.$$

However, all kit R of really found precepts contains additional contexts. In fact, ROSE uses powerful rules for recommendations.

Suppose that the client determines to abide by the first guidance for in it Defaults() (off assurance 1.0). Obviously, a new preference should get the default value. That is why it shifts the as it Defaults(). ROSE more shifts, which in

this instance are also as earlier, excluding the fact that now in it Defaults() is absent. Now situation additionally contains in it Defaults():

$$\Sigma_2 = \{change(fKeys[]),\ change(initDefaults())\}.$$

The client examines the methods makes General Page() and create Text Compare Page(), because they are located in the same file where f Keys[] does and initDefaults(). Each of these two techniques makes a page, where the setting could be customized. (on the Fig. 1 the page General is opened). Now it deepens method create General Page(), in the result of which arises a new situation Σ_3:

$$\Sigma_3 = \{change(fKeys[]),\ change(initDefaults()),\ change(createGeneralPage())\}.$$

Present this context, minimal support 3 and minimal probability 0,5, ROSE calculates the next precepts :

$$\Sigma_3 \Rightarrow \{change(plugin.properties)\}\ [5;1.0].$$

$$\Sigma_3 \Rightarrow \{change(TextMergeViewer())\}\ [3;0.6].$$

$$\Sigma_3 \Rightarrow \{change(propertyChange())\}\ [3;0.6].$$

$$\Sigma_3 \Rightarrow \{change(build.html)\}\ [3;0.6].$$

Applying aforementioned rules, we obtain sequence integration of all precepts, as they have the idem bygone and correspond to the context Σ_3. ROSE estimates an essence according to their assurance, having proposed to the client to shift the data set of plug-in. This data set contains specifications, which are utilized for the marks of precedence (for example, “Automatic comparison of opened structure” on the Fig.1).

In the two following sections, methods of intellectual analysis are represented: algorithm Apriori finds all powerful rules; ROSE touch finds only powerful and coincident rules. Correspond rules or not, depends on the situation Σ, where ROSE caused.

3. Priori approach to the intellectual analysis association precepts

One of the many well-known touches for detection of all-powerful unification precepts is A priori algorithm. It uses threshold value *min_supp* and *min_conf*, as well as corresponding data to the problem D as initial.

Inside A priori algorithm represent templates with the set of elements. $k-itemset$ – is a set of elements os the size k. Set of elements is named frequently if it responds the threshold of frequency (or support). The kit of all frequent sets k-objects is designated as L_k.

A priori attribute assists to shorten the area of scan for frequent sets of subjects: All not blank sub kits of frequent sets of objects should too be common.

It is clear because support is increasing if elements X are removed from the set of items $I: P(I) \le P(I-X)$. In such a way, i I t was common, $min\_supp \le P(I)$ then $I-X$ will be also common:

$$min\_supp \le P(I) \le P(I-X).$$

A priori algorithm composed of two stages:

1. A search of all private object sets.

Frequents sets of elements are generating at the level: at first L_1 is calculated, then L_1 is used for determination L_2, which is used for calculation L_3, and so on. This phase is completed, if more frequent k-elements are not found. Every layer, i.e. creating a set L_k, consists of four steps:

a. Binding step:

The applicant $k-itemset$ C_k is created by integration L_{k-1} with itself. The term of integration is the fact that first $k-1$ elements of two sets l_1 and l_2 are equal, and only the latest details are different $l_1[k] < l_2[k]$.

b. Trimming step:

Deleting of object sets from C_k, some cannot be frequent with the help of Apriority property. Verification if the subsets are private or not can be performed quickly, basing on the hash-tree of all frequent object sets.

c. Scanning or calculation step:

Scanning of databank D and calculation the frequencies of every remained applicant to C_k.

d. Creation step:

By the frequent sets k of elements L_k are those subsets in C_k, which satisfy the threshold of the frequency value.

A search of frequent element sets – the most labor-intensive part of A priori algorithm. For each layer the complete scanning of a database is required. In such a way, the support frequency (or threshold) has a great influence on the working time. . *2.Creating of association rules of frequent element sets.*

For each element set l all non-empty sub kits s are constructed. Such sub kit leads to the precept $s \Rightarrow l-s$ then and alone then, when:

$$confidence(s \Rightarrow l-s) = P(l-s|s) = \\ = \frac{frequency(l-s)}{frequency(s)} \geq min\_conf.$$

Check on the threshold of frequency (or support) can be ignored, as the precepts are constructed of frequent elements sets, and so the next check is ever faithful:

$$support(s \Rightarrow l-s) = P(l-s \cup s) = P(l) \geq min\_supp.$$

Work on A priori algorithm.

Candidate 1-element C_1 matches to the kit of all elements I. The step of calculation shows, that elements $\{E\}$ are not private. Then applicant of the 2-nd set C_2 generates in the way of binding of L_1 c with itself (C_2 is always cross-product of L_1). Along with $k=2$ any elements cannot be deleted, because all the subsets are single-point and are always contained in L_1. Calculation step identifies $\{B, D\}$ and $\{C, D\}$ as not common. Then applicant of 3-set C_3 generates of L_2, using the condition of binding $l_1[1]=l_2[1] \wedge l_1[2]>l_2[2]$. This repays three element sets. Two of them are not private according to the Apriori attribute and are cut: for $\{A, B, D\}$ sub kit {B, D} is not private, and subset $\{A, C, D\}$ of the points $\{A, C, D\}$ is not common. For the third applicant $\{A, B, C\}$ verifying of the databank confirmed that it is common. When all frequently used elements were detected, each element sets in L_2 and L_3 is used for precepts creation. Assurance is calculated for every precept and alone powerful precepts are returned.

Apriori property can only say that the set of objects is not frequent. Verification of element sets should always check database.

Apriori algorithm has a few disadvantages: D database is frequently scanned for each layer of frequent element sets creation. Besides, the formation of applicant kits is expensive. If there are 10^4 frequent 1-element sets, around 10^8 candidates of 2-element sets are generated. Furthermore, in order to detect a template of 100 sizes, Apriori algorithm should make more than 2^{100} applicants.

It is might find unification precepts on the basis of frequent pattern-growth (FPG) algorithm, which allows not only avoid costly procedure of candidates generation but also reduce the necessary number of DS approaches to two.

4. ROSE Approach to the search for association rules

The classic usage of A priori algorism is to calculate all the precepts more than minimal aid and assurance. However calculation of all concepts it is advisable for the scan of general templates, but not for providing recommendations.

A priori covering is very low. Sweep is linearly proportional to the quantity of separate bygones in the set of concepts R. High sweep is advise, so in this case ROSE can give prescriptions in most of occasions. Low sweep indicates that ROSE is frequently not effective.

Sweep may be enlarged in the way of R set deepen, for example, in the way of reduction of assurance and especially reference threshold. However for very low thresholds Apriori support can take several months. A narrow place is not Apriori, but those circumstances, that R becomes very big – bigger than $2^{|I|}$ in the worst occasion.

Certainly, very low steps of support have bad effect on recommendation character. Nonetheless, the developer should have possibility to determine thresholds aid meanings, but not only industrial borders, constructed by A priori algorism.

Scan of expensive compliance precepts. How it was mentioned before , R becomes highly big – for most projects production of numbers of transactions. In such a way, search of conformity precepts is costly, if R does not store in memory and there are no appropriate subject-heading frameworks.

In such a way, ROSE uses his own algorithm, which finds alone needed precepts on-the-spot. This algorism is based at two optimizations:

Analysis with limited bygones. In our concrete occasion bygone is even to the situation. Consequently, we have alone precepts, found on-the-spot, which correspond to the context Σ , i.e. rules, for which $\Sigma \Rightarrow B$. Search of the rules with such limited antecedents occupies just a little seconds. Complementary edges of such treatment is that additional in that meaning that permits to add new transaction to D between two contexts. In such a way, recommendations are always actual.

Analysis of only sequences. In order to speed up a process of search even more, we calculate only the precepts with one element in their result. So, for the context Σ the precepts have the kind of $\Sigma \Rightarrow \{i\}$. For ROSE such precepts are enough, so ROSE in any case calculates sequences association. In such a way, review of non single-element sequences is excessive: for each element $i \in B$ rules $\Sigma \Rightarrow B[s;c]$ single-element rule exists $\Sigma \Rightarrow \{i\}[si;ci]$ with a higher or equal support and certainty meaning $s_i \geq s$ and $c_i \geq c$, i.e. $frequency(\Sigma \cup \{i\}) \geq frequency(\Sigma \cup B)$.

Algorithm of search ROSE consists of three stages:

Transactions search. Search of all deals T , containing all elements of Σ context, i.e. $\Sigma \subseteq T$. Let's denote these transactions as

$$\sigma_{TransactionID}(Lineitems \div \Sigma).$$

Grouping and sorting. Grouping of elements

$$Lineitems \triangleright\!\triangleleft \sigma_{TransactionID}(Lineitems \div \Sigma)$$

Of these transactions according to *EntityID* and their descending sorting.

Creation of rules. One single-element rule corresponds to each group.

- frequency Σ is a maximal number of groups (that is possible to element groups $i \in \Sigma$ and always right for the first returned group).

-account for element group i is a frequency of the rule $\Sigma \Rightarrow \{i\}$.

- Certainty of the rule $\Sigma \Rightarrow \{i\}$:

$$\frac{frequency(\Sigma \Rightarrow \{i\})}{frequency(\Sigma)}.$$

- to disregard trivial rules $\Sigma \Rightarrow \{i\}$, $i \in \Sigma$.

Only the rules, satisfying the support threshold and certainty are returned.

On the Fig. 2 the example of search algorithm ROSE is shown. Let's assume, that situation $\Sigma = \{A, B\}$. Firstly, ROSE performs the search according to all transactions: 2, 100, 300 and 700. Then it groups namely these transactions according to the elements and sorts them in descending order. The biggest account for element A , that is why frequency for Σ is equal to 3. The precepts for A and B are toil (because both of them are located in the situation), that is why they are ignored.

Situation $\Sigma = \{A, B\}$ and $k = |\Sigma| = 2$.

Creation of repeated k-element and k+1-element sets, which containing Σ .

TxID	List of items
100	A, B, C
200	A, D
300	A, B, C
400	B, D

find →

TxID	List of items
100	A, B, C
300	A, B, C
700	A, B

group&sort →

500	A, D
600	B, E
700	A, B

group&sort →

Item	Frequency
A B C	3 3 2

⇒ {A, B}
⇒{A, B}
⇒{A, B, C}

Creation of single-element rules with antecedent Σ.

article	periodicity
A B C	*periodicity*(Σ) =3
	3
	2

{A,B} ⇒ {A} is toil
{A,B} ⇒ {B} is toil
{A,B} ⇒ {C} has *periodicity* =2, *assurance* =2/3 and is powerful

Fig. 2. Example for ROSE algorithm
(min_freq=2; min_conf=0,5)

For C the precept $\Sigma=\{A, B\}\Rightarrow\{C\}$ is powerful, so threshold meanings for *min_freq* и *min_conf* are contented.

Aforementioned optimization makes algorithm of search much more effective: average duration of making a request is around 0,5sec for huge versions as GCC.

ROSE provides one more search algorithm for individual antecedent 1-element precepts $\{a\}\Rightarrow\{b\}$. Such precepts are less accurate for a prescription, but the prices for measuring and visualization of the connection among the objects. Algorithm is exactly like A priori algorithm, represented in (p. 3), except that only two frequently used element sets are generated and used for the rules creation.

5. Evaluation of results

For our estimation, we have analyzed chanceries of eight huge plans with opened initial code, table 1. For each archive we have chosen several full months, containing latest 1000 deals, but not above than 50% of all deals as our estimation period. In this period we test every deal T, if its elements may be foretold from above anecdote:

1. Creating a check of occasion $q=(Q, E)$, consisting of a request $Q\subseteq T$ and the anticipated result $E=T-Q$.

2. Accept all the deals T_i, finished until time (T), as a teaching kit, and comply with a set of rules R from these deals.

3. In order to get away the usage of infinite list of offers, ROSE shows only ten single precepts $R_{10}\subseteq R$, estimated according to the level of certainty. Apply R_{10}, in order to obtain the request result $A_q=apply_{R_{10}}(Q)$. The scale A_q will ever be under or equal to ten.

Table 1. Analyzing projects (Txn=transaction, Ety=essence)

Draft (in CVS with) *Depiction*	**History** (Training)			**Evaluation**
	#Txns1	#Txns/Day	#Etys/Txn	#Txns
ECLIPSE *integrated environment*	46,843	56.0	3.17	2,965
GCC *compiler collection*	47,424	22.4	3.90	1,083
GIMP *image manipulation tool*	9,796	4.1	4.54	1,305
JBOSS *application server*	10,843	9.0	3.49	1,320
JEDIT *text editor*	2,024	2.9	4.54	577
KOFFICE *office suite*	20,903	11.2	4.25	1,385
POSTGRESQL *database system*	13,477	5.4	3.27	925
PYTHON *language + library*	29,588	6.2	2.62	1,201

4. The product A_q of the check example q compose of two parts:

- $A_q\cap E_q$ - elements, which correspond to the expected result are considered to be correct predictions;

- A_q-E_q - unexpected recommendations, which are considered to be false predictions and named as malfunctioning.

Besides, ROSE can have missed objects:

- $E_q - A_q$ - predictions are absent and are named as false negatives.

Sets A_q и E_q indicated on the Fig. 3.

For the result estimation A_q we use two dimensions from obtained information:

- accuracy P_q - which part of the returned elements was virtually right, i.e. anticipated by the client – the higher the accuracy, the less dummy inclusions;

- review R_q - percent of accurate predictions. The upper response, the less pseudo inclusions.

$$P_q = \frac{|A_q - E_q|}{|A_q|} \text{ and } R_q = \frac{|A_q \cap E_q|}{|E_q|}$$

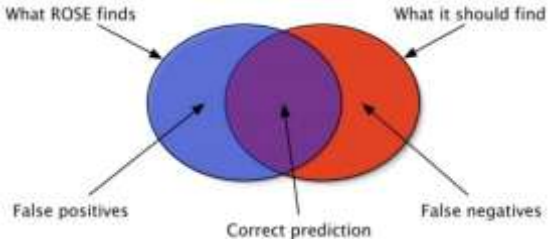

Fig. 3. Accuracy and review

In case, when low essences do not return (A_q blank), we determine accuracy as $P_q = 1$, and in an occasion, when any essences are not anticipated, we determine the review as $R_q = 1$.

Our purpose is to seek tall accuracy, as well as high review meanings, this means to recommend everything (review 1) and only expected objects (accuracy 1). In practice, however, review and accuracy negatively correlate with each other. For high review, it is possible to return many or even all elements, which leads to low accuracy. On the other hand, it is possible to obtain high accuracy but low review only recommending several certain elements.

For each request q_i we obtain a pair with precision review (P_q, R_q). We bind these pairs into one pair, applying two various methods of average from data search.

Macro-evaluation just accept average meaning of pairs of precision review:

$$P_M = \frac{1}{N}\sum_{i=1}^{N} P_q \text{ и } R_\mu = \frac{\sum_{i=1}^{N} |A_q \cap E_q|}{\sum_{i=1}^{N} |E_q|}$$

This treatment defines prognostic power for each request, using meaning of accuracy and review for each request.

Macro-evaluation builds average pair of accuracy-review on the basis of elements. It uses not meanings of accuracy and review of single requests, but amounts of returned, accurate and desired elements.

$$P_\mu = \frac{\sum_{i=1}^{N} |A_{q_i} \cap E_{q_i}|}{\sum_{i=1}^{N} |A_{q_i}|} \text{ и } R_\mu = \frac{\sum_{i=1}^{N} |A_{q_i} \cap E_{q_i}|}{\sum_{i=1}^{N} |E_{q_i}|}$$

Distinction betwixt macro-evaluation and micro-evaluation is significant. Micro-evaluation calculates mean accuracy and review on each element, but not on the request. This is pictorial in the incoming sample:

Lets' assume, we have two talks:

- talk A from 200 pupils, 50 of each wear spectacles. In such a way, correlation for this lecture is 25%.
- talk B with 40 pupils, 30 of which wear glasses. In the result of which correlation is 75%.

Averaging these indicators with macro-and micro-evaluation we obtain:

- Macro-estimation accepts average meaning of either coefficients:

$$\frac{25\% + 75\%}{2} = 50\%$$

Mean value is calculates on the layer of lecture. This means, that 50% of students which wear glasses are participating in lecture in average.

- Micro-evaluation, on the contrary, calculates average meaning on the level of a student:

$$\frac{200.25\% + 40.75\%}{200 + 40} = \frac{80}{240} = 33{,}3\%$$

If students A and B do not intercross, this denotes, that each third pupil of this two talks wears spectacles.

This sample indicates that it is important carefully to use and interpret average meaning.

Conclusion

Numeral evaluation of the results. ROSE can be useful tool for requests production about further changes and warnings upon lacking shifts. Nevertheless, the larger it is possible to obtain lessons from the bygone, the better and more the offers maybe done:

1. For stabile schemes, such as GCC, ROSE produces a set of accurate proposals: 44% connected files and 28% of connected objects can be predicted with accuracy around 40% for each separate proposal and probability more than 90% for three best proposals.

2. For fast developing systems, such as KOFFICE or JEDIT, the most useful offers of ROSE are located on the level. In general, it is unsurprisingly stable, because ROSE will should forecast renewed functions, whether, likely, are not available for every treatment.

3. In 4-7% of all transaction errors ROSE accurately detects a lack change. If such warnings appears, it most be take it earnestly, because alone 2% of all deals induce dummy alerts.

Prospects of the system improving

1. Aspect identification. If the parts of the program were changed together several times, common abstractions of separate changes can be candidates for facets .Then connection development will be transformed into a single syntactic essence, so further changes may be performed only in one site.

2. Rule representation. Precepts, discovered by ROSE, depict the process of developmentally s/w, which can be or not be assumed operation. In order to do these rules clear, it is possible to use visual analysis for detection of conformities in logically connected elements.

3. Taxonomies. Each change in the method presumes the change of incoming class, which also presumes changes in attached files or packet. It is possible to use taxonomies for detection of such models as “this change presumes the change in this packet” (and not “in this method”). They can be under accurate in position, but will ensure more high certainty.

4. Consistency rules. Just now we bind only changes, which are happening in one transaction. Further we will also detect the rules for several transactions: “System is always tested before issue” (as stated by the correspondent changes).

5. Additional sources of the data. Archived shifts comprise larger than only a creator, date and position. It was possible to scan journal messages (including one of changes, which is necessary to be performed), in order to define, which problem does the change relate to.

6. Program analysis. One more not using source of the data is a program analysis. Albeit our treatment can find connection between elements, which are not arise programs, knowing in programs semantic also can help to separate connected changes with probable and improbable.

Reference list

1. “Evaluating and improving fault localization” by Spencer Pearson, José Campos, René Just, Gordon Fraser, Rui Abreu, Michael D. Ernst, Deric Pang, and Benjamin Keller. In ICSE 2017, Proceedings of the 39th International Conference on Software Engineering, (Buenos Aires, Argentina), May 2017.

2. S.L. Pfleeger, J.M. Atlee. Software Engineering: Theory and Practice, 4th edition, Pearson Education, 2010.

3. Sommerville. Software Engineering, 9th edition, Pearson Education, 2011.

4. “Verifying Invariants of Lock-free Data Structures with Rely-Guarantee and Refinement Types” by Colin S. Gordon, Michael D. Ernst, Dan Grossman, and Matthew Parkinson. ACM Transactions on Programming Languages and Systems, *vol. 39, no. 3, May 2017, pp. 11:1-11:54.*

5. T. Mens, S. Demeyer (Eds.), Software Evolution, Springer, 2008. International Conference on Software Maintenance, IEEE International Conference on Program Comprehension, IEEE.

6. *SoftChange. Project homepage. http://sourcechange.sourceforge.net/, June 2004.*

7. *Daniel German and Audris Mockus. Automating the measurement of opensource projects. In Proceedings of ICSE '03 Workshop on Open Source Soft-ware Engineering, Portland, Oregon, USA, May 2003.*

8. *Bloof. Project homepage. http://bloof.sourceforge.net/, June 2004.*

9. *Dirk Draheim and Lukasz Pekacki. Process-centric analytical processing of version control data. In Proc. International Workshop on Principles of Software Evolution (IWPSE 2003), Helsinki, Finland, September 2003. IEEE Press.*

10. *Michael Fischer, Martin Pinzger, and Harald Gall. Populating a release history database from version control and bug tracking systems. In Proc. International Conference on Software Maintenance (ICSM 2003), Amsterdam, Netherlands, September 2003. IEEE.*

11. *Michael Fischer, Martin Pinzger, and Harald Gall. Analyzing and relating bug report data for feature tracking. In Proc. 10th Working Conference on Reverse Engineering (WCRE 2003), Victoria, British Columbia, Canada, November 2003. IEEE.*

12. *Davor ˇCubrani´c and Gail C. Murphy. Hipikat: Recommending pertinent software development artifacts. In Proc. 25th International Conference on Software Engineering (ICSE), pages 408–418, Portland, Oregon, May 2003.*

13. *Harald Gall, Mehdi Jazayeri, and Jacek Krajewski. CVS release history data for detecting logical couplings. In Proc. International Workshop on Principles of Software Evolution (IWPSE 2003), pages 13–23, Helsinki, Finland, September 2003. IEEE Press.*

14. *James M. Bieman, Anneliese A. Andrews, and Helen J. Yang. Understanding change-proneness in OO software through visualization. In Proc. 11th International Workshop on Program Comprehension, pages 44–53, Portland, Oregon, May 2003.*

15. *Lijie Zou and Michael W. Godfrey. Detecting merging and splitting using origin analysis. In Proc. 10th Working Conference on Reverse Engineering (WCRE 2003), Victoria, British Columbia, Canada, November 2003. IEEE.*

16. *Annie Tsui Tsui Ying. Predicting source code changes by mining revision history. Master's thesis, University of British Columbia, Canada, October 2003.*

17. *Amir Michail. Data mining library reuse patterns in user-selected applications. In Proc. 14th International Conference on Automated Software Engineering (ASE'99), pages 24–33, Cocoa Beach, Florida, USA, October 1999. IEEE Press.*

18. *Annie Chen, Eric Chou, Joshua Wong, Andrew Y. Yao, Qing Zhang, Shao Zhang, and Amir Michail. CVSSearch: searching through source code using CVS comments. In Proc. International Conference on Software Maintenance (ICSM 2001), pages 364–374, Florence, Italy, November 2001. IEEE.*

19. *David L. Atkins. Version sensitive editing: Change history as a programming tool. In B. Magnusson, editor, Proceedings of System Configuration Management SCM'98, volume 1439 of LNCS, pages 146–157. Springer-Verlag, 1998.*