What Strokes to Modify in the Painting? Code Changes Prediction for Object-Oriented Software

Dinan Zhang[1] Shizhan Chen[1] Qiang He[2,*] Zhiyong Feng[1]
Keman Huang[4]

1 Division of Intelligence and Computing, Tianjin University, China

2 School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn 3783, Victoria Australia

3 Sloan School of Management, MIT, Cambridge 02142, MA, USA

dnzhang@tju.edu.cn

November 23rd, 2018

Contents

Contents

Motivation

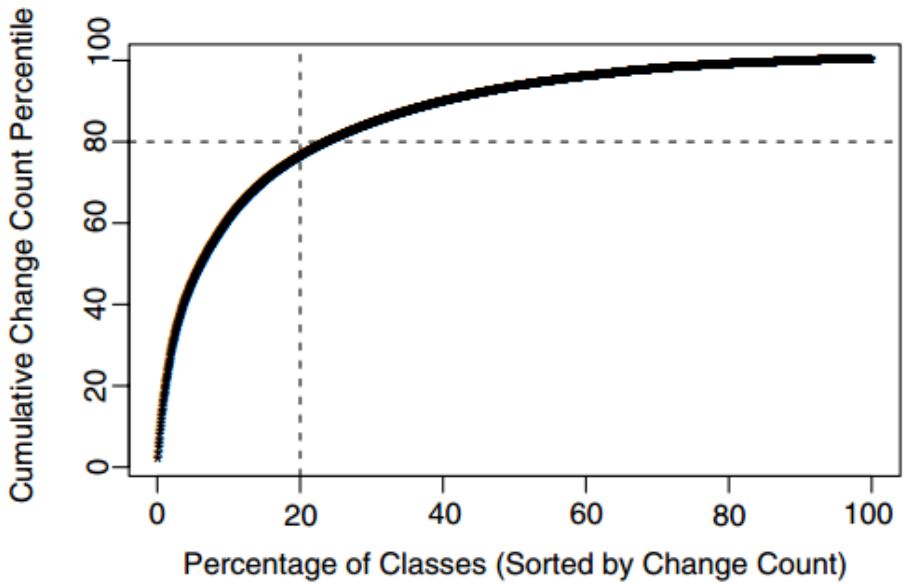
Software evolution: Software systems shall evolve to fulfill users' increasingly various and sophisticated needs.

Key Words:

1. Inevitable process
2. Software degradation
3. Evolutionary Characteristics
4. Software development efficiency
5. Software change
   - Software Maintenance
   - Software Reengineering

An Example

1. The software maintenance cost is high
2. A great majority (around 80%) of code changes are rooted in a small proportion (around 20%) of classes



Related Work

The related work of the study and involves three sections:

1. Use several structural metrics to describe the objective-oriented characteristics:

Related Work

The related work of the study and involves three sections:

1. Use several structural metrics to describe the objective-oriented characteristics:

- C&K metrics suite [**Chidamber S and Kemerer C, 1991**]
- QMOOD metrics [**Bansiya J and Davis C,2002**]
- Afferent coupling (Ca) and Efferent coupling (Ce) metrics[**Martin,2002**]
- OO metrics on change-proneness should also consider class size as a cofounding variable [**Zhou,2009**]
- size metrics is better than the coupling and cohesion [**Lu,2012**]

2. Some studies attempted to summarize the change history of a software in order to identify change-prone classes:

2. Some studies attempted to summarize the change history of a software in order to identify change-prone classes:

- Yesterday's Weather [**Girba, 2004**]
- evaluated the probability that each class of the system will be affected [**Tsantalis,2006**]
- build release-by-release prediction models [**Elish and AlKhiaty,2013**]
- statistical techniques,machine learning techniques used to construct prediction model

3. statistical techniques,machine learning techniques used to construct prediction model:

3. statistical techniques,machine learning techniques used to construct prediction model:

- combined each ordered list of metric into a new single list [**Eski and Buzluca, 2011**]
- reuse the generated prediction model of one project and validate it on another project [**Malhotra and Khanna,2013**]
- nalyzed and compared the predictive performance of different machine learning techniques and a statistical technique [**Malhotra and Khanna ,2017**]
- evaluated the developmental models with the use of wellknown statistical tests [**Malhotra and Khanna ,2017**]

Contributions

Our main contributions include:

Contributions

Our main contributions include:

1. **Two new metrics** are proposed for describing the code changes in classes between successive software releases, and the corresponding change sizes.
2. An approach is proposed for predicting the **change-size** of change-prone classes.
3. Extensive experiments based on 17 real-world OO software systems were conducted to evaluate the prediction accuracy achieved by our approach.

Contents

CCM & CSM Definition

Definition (CCM & CSM)

Class Change Metric (CCM). This metric describes whether a class is added, deleted or modified in $version_i$ in comparison to $version_{i+1}$. It has three optional values: Added, Deleted, Modified, formally defined below:

CCM & CSM Definition

Definition (CCM & CSM)

Class Change Metric (CCM). This metric describes whether a class is added, deleted or modified in $version_i$ in comparison to $version_{i+1}$. It has three optional values: Added, Deleted, Modified, formally defined below:

1. ***Added.*** It is a Boolean value that indicates whether the class has been newly added in $version_{i+1}$.
2. ***Deleted.*** It is a Boolean value that indicates whether the class has been deleted from in $version_{i+1}$
3. ***Modified.*** It is a Boolean value that indicates whether the class has been changed $version_{i+1}$

CCM & CSM Definition

Definition (CCM & CSM)

Change Size Metric (CSM). This metric decsribes the size of a change in the class between two successive releases. This way, the sizes of class changes are quantified. The value of this metric is calculated in two cases.

CCM & CSM Definition

Definition (CCM & CSM)

Change Size Metric (CSM). This metric decsribes the size of a change in the class between two successive releases. This way, the sizes of class changes are quantified. The value of this metric is calculated in two cases.

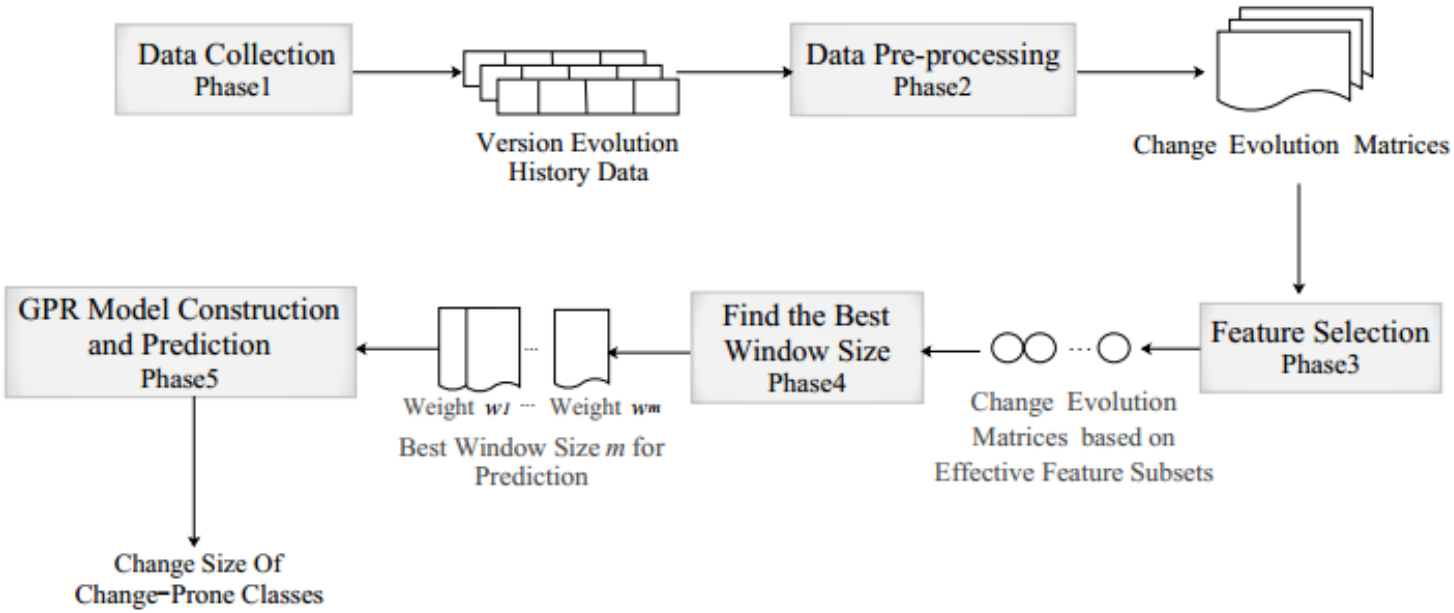
1. ***Common Classes.*** The source code of $class_j$ in $version_i$ is $source\,code(i)_{class_j}$, the source code of $class_j$ in $version_{i+1}$ is $source\,code(i+1)_{class_j}$. The similarity between them is calculated, $similarity(i+1)_{class_j}$. The change size is:

$$\text{change size}(i+1)_{class_j} = 1 - \text{similarity}(i+1)_{class_j}$$

2. ***Added or Deleted Classes.*** If a class is added or deleted in the next release, its change size is assigned a fixed value of 1.

Contents

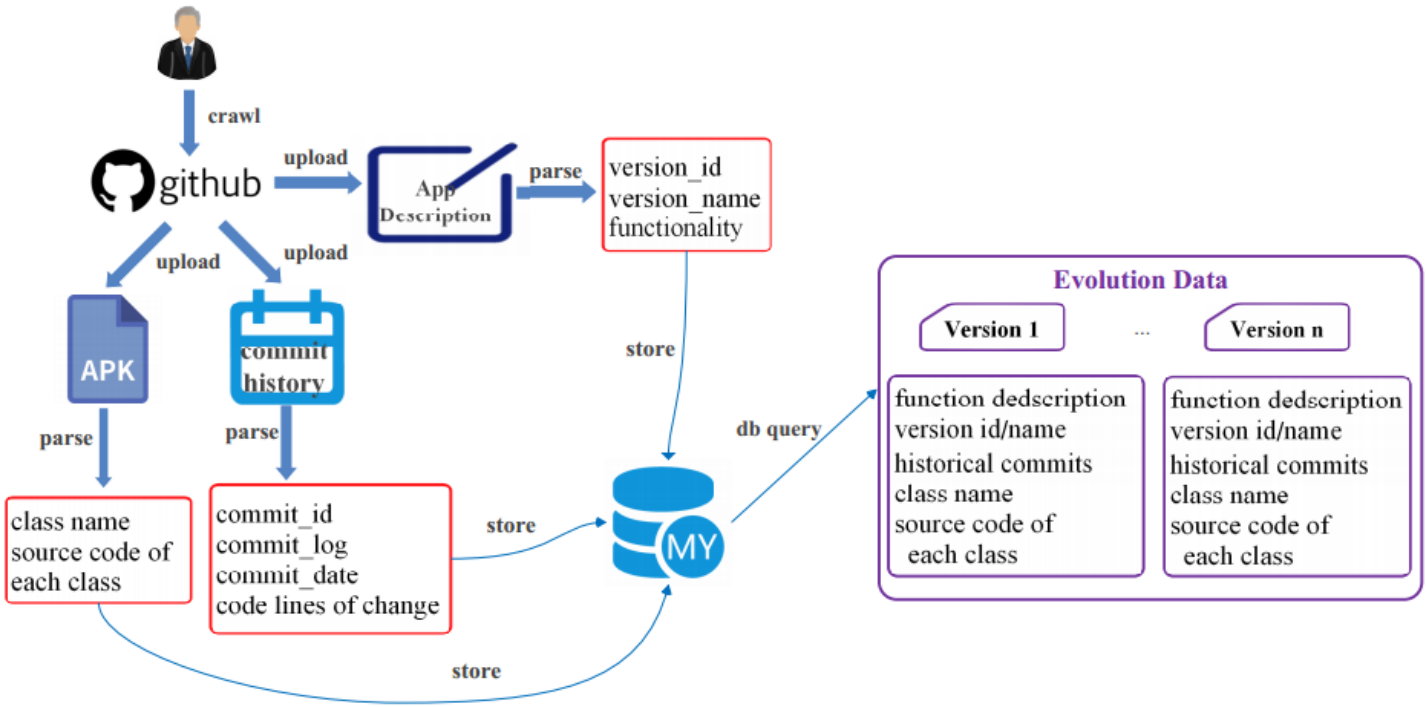
Framework Overview



In order to identify and rank change-prone classes in an OO software, we need to:

1. Collect effective historical changes.
2. Construct a regression model to predict the change sizes of classes for the new software release.

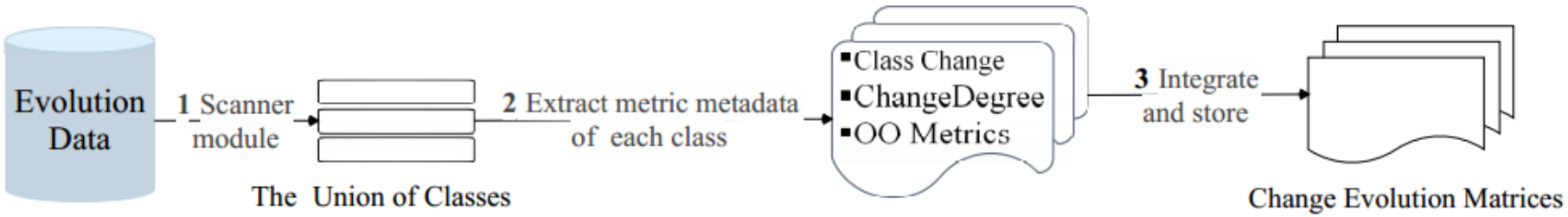
Data Collection



The historic version evolution data need to be collected, include:

1. software description
2. commit history
3. the source code of all releases

Data Pre-processing



The procedure for pre-processing the collected data about the software sysytem involve three steps:

1. Scanner Module
2. Metric Extraction
3. Integrate and Store

Data Pre-processing

1. Scanner Module: All the classes that appear since the first version until the last release are extracted and combined. Assume that there is a $version_i$ in an app, the collection of classes is represented as:

$$Ci = \{\text{class}|\text{class} \in version_i\}(0 < i \leq v)$$
$$C = C_1 \cup C_2 \ldots \cup C_{v-1} \cup C_v$$

2. Metric Extraction: TEach class is marked with 22 metrics,The CKJM tool is employed to calculate the 18 OO metrics for each class, include:

1. C&K
2. QMOOD
3. Ca Ce
4. other metrics proposed by Henderson-Sellers

Data Pre-processing

No.	Metrics	Description	Characteristic	Metric Suite
1	WMC	Weighted methods per class	Size	C&K
2	DIT	Depth of Inheritance Tree	Inheritance	C&K
3	NOC	Number of Children	Inheritance	C&K
4	CBO	Coupling between object classes	Coupling	C&K
5	RFC	Response for a Class	Coupling	C&K
6	LCOM	Lack of cohesion in methods	Cohesion	C&K
7	MOA	Measure of Aggregation	Composition	QMOOD
8	DAM	Data Access Metric	Encapsulation	QMOOD
9	MFA	Measure of Functional Abstraction	Inheritance	QMOOD
10	NPM	Number of Public Methods for a class	Size	QMOOD
11	CAM	Cohesion Among Methods of Class	Cohesion	QMOOD
12	Ca	Afferent coupling	Coupling	other
13	Ce	Efferent coupling	Coupling	other
14	AMC	Average Method Complexity	Size	other
15	LOC	Lines of Code	Size	other
16	LCOM3	Lack of cohesion in methods Henderson-Sellers version	Cohesion	other
17	IC	Inheritance Coupling	Coupling	other
18	CBM	Coupling Between Methods	Coupling	other

The difference between two successive versions in each metric is measured. The corresponding change in $class_j$ is calculated as follows:

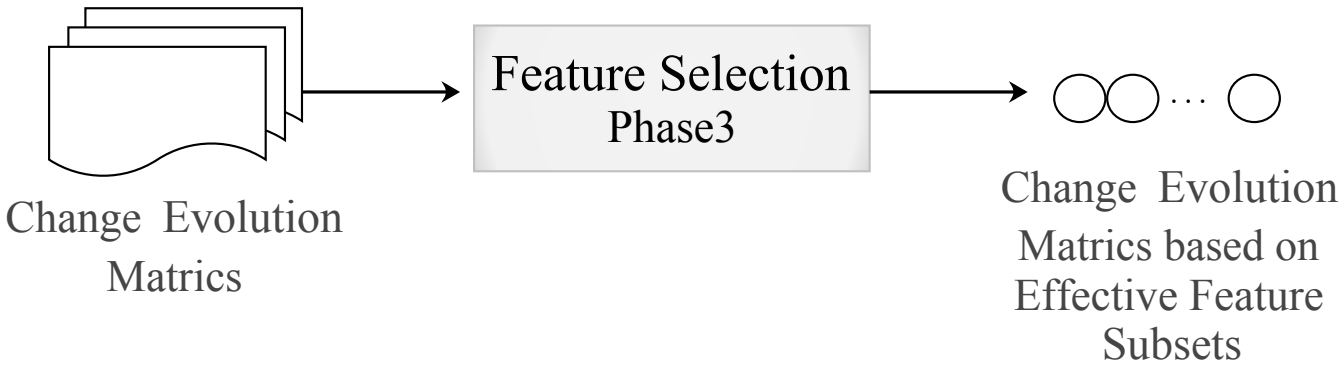
$$\Delta wmc(i+1)_{class_j} = wmc(i+1)_{class_j} - wmc(i)_{class_j}$$

3. Integrate and Store: we build a change evolution matrix based on every two successive releases. For an OO software with m releases and n classes as an example, a total of $m-1$ matrices are built.

Metrics / Class	*Added*	*Deleted*	*Modified*	*ΔOO Metrics*	*Change Size*
Class 1	0/1	0/1	0/1	Δ Metric$_1$ ... Δ Metric$_{18}$	0~1
Class 2	0/1	0/1	0/1	Δ Metric$_1$ ... Δ Metric$_{18}$	0~1
Class n	0/1	0/1	0/1	Δ Metric$_1$ ... Δ Metric$_{18}$	0~1

Feature Selection

Some of 22 metrics measure the same properties of the class. Thus, it is not necessary to include all these 22 metrics.

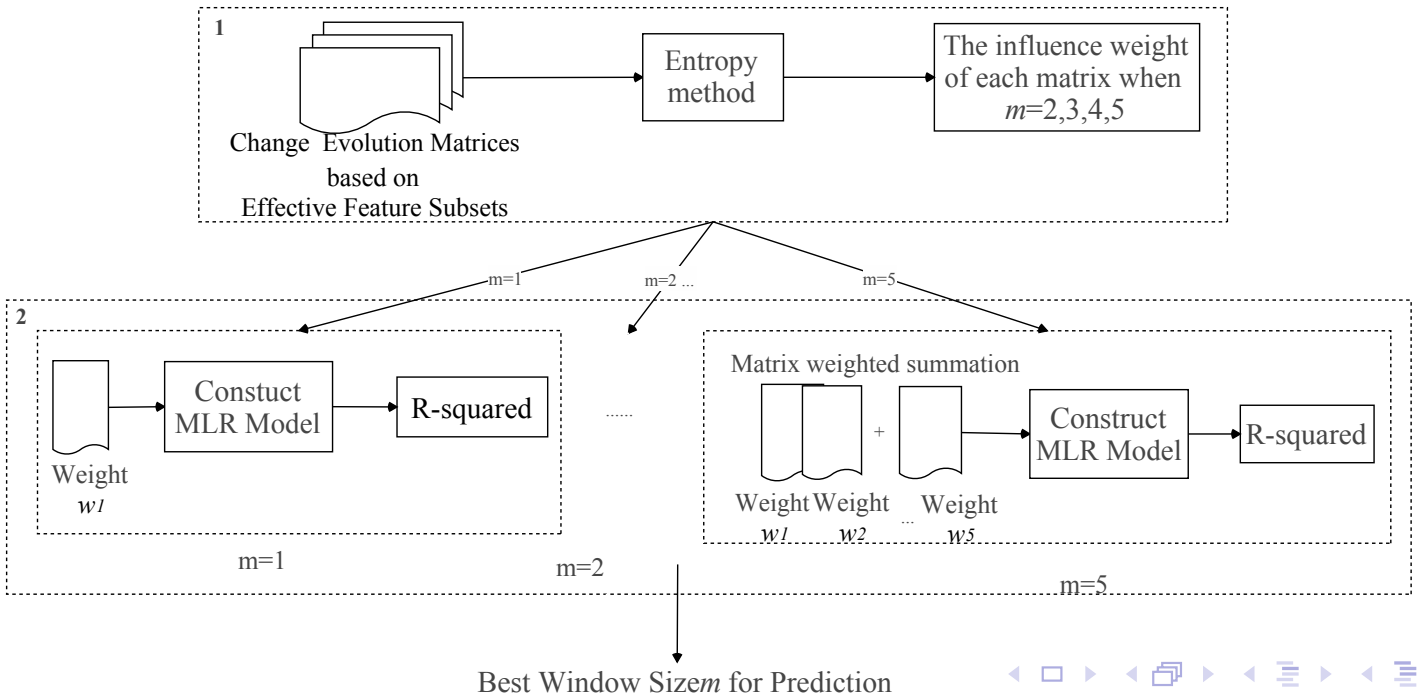


Find the Best Window Size

Definition (the window size for prediction)

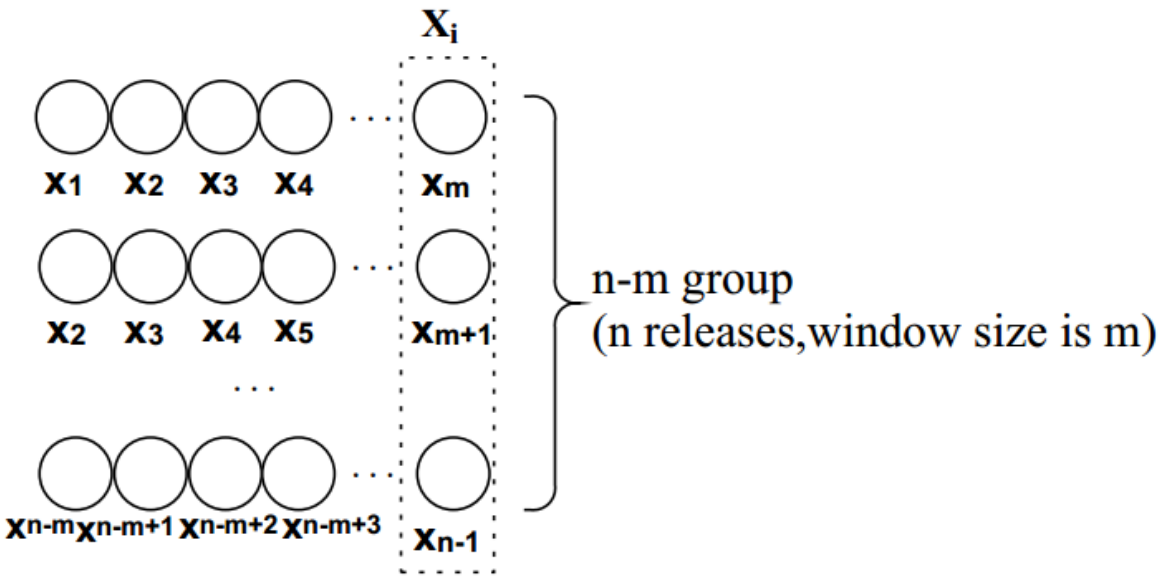
The number of previous releases used to predict class change-proneness.

1. Calculate the Weight of Matrix : Different previous releases may have different impacts on the change-proneness prediction for the next release. According to the information entropy theory,the weight is determined by the size of the index variability.



Find the Best Window Size

Following three steps, we can obtain the weight of each matrix when the window size m is different.



- *Step 1 Data-processing.* m indicators, i.e., $X_1, X_2, \ldots X_m$, where $X_i = \{x_1, x_2, \ldots, x_{n-m}\}$.

$$Y_{ij} = \frac{x_{ij} - min(X_i)}{max(X_i) - min(X_i)} \tag{1}$$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n-m$.

Find the Best Window Size

- *Step 2 Information entropy calculation.* The information entropy of a set of data, i.e., E_i, $i = 1, 2, \ldots, m$, is determined as follows:

$$E_i = -\ln(n-m)^{-1} \sum_{j=1}^{n-m} p_{ij} \ln p_{ij} \tag{2}$$

$$p_{ij} = \frac{Y_{ij}}{\sum_{j=1}^{n-m} Y_{ij}} \tag{3}$$

If $p_{ij} = 0$,

$$\lim_{p_{ij}=0} p_{ij} \ln p_{ij} = 0 \tag{4}$$

- *Step 3 Determine the weight of each matrix.*The weight of the m matrices are calculated through information entropy, formally

$$w_i = \frac{1 - E_i}{k - \sum E_i} (i = 1, 2, \ldots, m) \tag{5}$$

Find the Best Window Size

2. Determination of optimal window Size : The optimal window size m for prediction is calculated based on regression model.The fitting effect of these models determines the window size for prediction.

The independent variable. According to the weights of different matrices in different window sizes, $w_1, w_2, w_3 \ldots, w_n$, there is $w_1 + w_2 + w_3 \ldots + w_n = 1$. The n matrices are combined as follows:

$$w_1 Matrix_1 + w_2 Matrix_2 + \ldots + w_n Matrix_n \rightarrow Matrix \tag{6}$$

The dependent variable. It is a dichotomous variable that indicates whether a class was changed from the previous release to the current release. A class is considered changed when there is code addition, deletion or modification.

The Gauss process can handle small samples with high dimensionality. Therefore, we construct our prediction model using the Gauss Process Regression (GPR) algorithm.

Contents

Data Sources

The criteria for selecting the OO software sysytems are:

- A number of releases.
- Different sizes and contain reasonably large number of classes.
- Different lifespans and they are long-lived.
- Open-sourced.

Software	Time Frame	Versions	No. of classes	No. of changed classes
WordPress	2011-4-13 ~2017-11-14	254	7718	1282
V2ex-android	2015-07-06 ~2017-08-18	6	3443	144
Qksms	2015-09-01 ~2016-09-05	24	3751	700
TweetLanes	2013-02-23 ~2014-10-08	22	1986	829
Douya	2016-02-27 ~2017-07-15	8	6630	2858
POT-Droid	2012-01-30 ~2017-05-01	30	9768	3086
FaceSlim	2015-01-18 ~2017-09-27	44	3958	1911
MLManager	2015-05-26 ~2016-03-29	13	4066	814
Transdrone	2013-06-28 ~2017-03-17	24	3608	1332
AnySoft Keyboard	2012-4-25 ~2017-7-27	62	3841	2707
Zxing	2014-01-19 ~2017-10-25	23	448	299
Googlesamples	2015-07-06 ~2017-08-18	6	3889	1097
AppOpsx	2017-01-11 ~2017-09-30	12	7105	2288
Rpicheck	2014-8-29 ~2017-08-27	17	9560	3129
Reddit is fun	2011-11-27 ~2012-01-15	10	818	131
Financius	2014-10-18 ~2015-01-31	35	9233	2476
Openfoodfacts	2015-05-12 ~2017-03-24	25	9740	4320

Metric Suite Selection

- Seven systems select CAM, DMP, which are QMOOD metrics.
- Four metrics have never been selected by any sysytems, including NOC, DAM, IC and CBM.
- The Class Change Metric and Change Size Metric are selected by 12 out of systems, the most of all metrics.
- Some metrics have even never been selected.

Window Size Selection

The statistical results are shown in the following Tables. It shows that the matrix has different weight when the window size is different.

Window Size	Weight On Average
m=2	0.471
	0.529
m=3	0.291
	0.323
	0.386
m=4	0.219
	0.213
	0.246
	0.322
m=5	0.188
	0.186
	0.180
	0.214
	0.232

(a) THE WEIGHT OF EACH MATRIX WHEN m IS DIFFERENT

Window Size	R-quared On Average
m=1	0.79062
m=2	**0.83174**
m=3	0.78409
m=4	0.77579
m=5	0.72828

(b) R-SQUARED OF DIFFERENT WINDOW SIZE m

Contrast Model

- ***CEM:*** Our model uses the Change Evolution Matrices (based on the selected metric suite) as independent variables.
- ***PSM :*** This probability estimation model uses the internal class probability of changes as independent variable.
- ***QHC :*** This model uses the evolution-based metrics as independent variables based on the quantifying historical changes model
- ***C&K :*** This model uses the C&K metrics as independent variables.

Prediction Results

The highest performance in R-squared is obtained on the TweetLanes with 0.932, and the average value in R-squared and RSS are 0.871 and 0.113 respectively.

	R-squared	Residual sum of squares	Classification accuracy	Average relative error
Average	0.871	0.113	87.9%	0.048

The Robustness of Our Method We use the area under the receiver operating characteristic (AOC) curve statistic (AUC), the median precision and recall values to evaluate these classification models.

	CEM	PSM	QHC	C&K
Median AUC	**0.89**	0.71	0.79	0.81
Median Precision	**0.88**	0.68	0.81	0.76
Median Recall	**0.85**	0.76	0.81	0.83

The Change Size Ranking List Prediction

The following ranking evaluation metrics are calculated to measure the performance of the ranking list prediction.

1. *Mean Average Precision (MAP)*:

$$MAP = \frac{\sum_{r=1}^{N}(\frac{r}{N_r} I_r)}{N_{used}} \tag{7}$$

2. *Normalized Discounted Cumulative Gain (NDCG)*:

$$NDCG = \frac{1}{S_N} \sum_{j=1}^{N} \frac{(2^{r(j)} - 1)}{log_2^{(1+j)}} \tag{8}$$

The Table presents the MAP and NDCG for the four approaches. It shows that ***our model outperforms*** other models in both MAP and NDGC.

Index	CEM	PSM	QHC	C&K
MAP	**83.3%**	70.6%	75.8%	79.2%
NDCG	**88.2%**	75.1%	79.6%	82.5%

Contents

Conclusion

In our paper, we present

- We propose Class Change Metric & Change Size Metric for describing the changes made in classes between different software releases.
- Our model is the first attempt to predict the change size ranking list of change-prone classes.
- Our approach is experimentally evaluated on 17 real-world data sets collected from the Github. The results show that our model significantly outperforms three existing models.

THANK YOU!