AutoODC: Automated Generation of Orthogonal Defect Classifications

LiGuo Huang[1] *Vincent Ng*[2] *Isaac Persing*[2] *Ruili Geng*[1] *Xu Bai*[1] *Jeff Tian*[1]

Department of Computer Science and Engineering, Southern Methodist University[1]
Human Language Technology Research Institute, University of Texas at Dallas[2]
Dallas, TX, USA

INTRODUCTION

- The systematic classification and analysis of defect data bridge the gap between causal analysis and statistical quality control, provide valuable in-process feedback to the system development or maintenance, as well as improve system and software quality.
- The analysis results based on systematic defect classifications enable us to understand the major impact types of system defects and to pinpoint specific problematic areas for focused problem resolution and quality improvement.
- Orthogonal Defect Classification (ODC) is the most influential among general frameworks for software defect classification and analysis.

Issues with Manual ODC Generation

- Manual ODC defect classification based on assimilation of existing defect repositories (e.g., defect/bug reports) is extremely effort consuming esp. for novices.
  - In the GSM Base Station Systems (GSMBSS) software development group at Motorola GSM Products Division, 5 experienced development engineers who are familiar with both project development and ODC were assigned to classify all post-release defects since 1996 by defect "impact" attribute. It took them approximately 6 minutes to classify each defect. Considering a 1000 defect data set, it would take approximately 2.5 weeks for 5 experienced engineers to complete the classification only for one single ODC attribute.
- The types of follow-on ODC-based defect analyses are often restricted by the limited defect classification results. Multi-way defect analyses may not be supported.

Major Contribution of *AutoODC*

- From a **software engineering** perspective, we semi-automate ODC defect classification and evaluate our approach on the ODC "***Impact***" attribute.
- From an **AI** perspective, we **propose the annotation relevance framework,** which aims to improve automated ODC classification by enabling a machine learning algorithm to exploit additional experts' domain knowledge expressed in the form of relevant annotations.
  - **Traditionally,** an ODC expert analyst simply provides the correct classification of a defect record.
  - In the **annotation relevance** framework, the analyst additionally provides for each defect record ***relevant annotations***, each of which is a text segment (typically a word or phrase) that she believes can help her determine the ODC classification the record. These annotations
    - help a learning algorithm focus on the portions of the report relevant for classification
    - need to be provided only for the training records, not for the test records

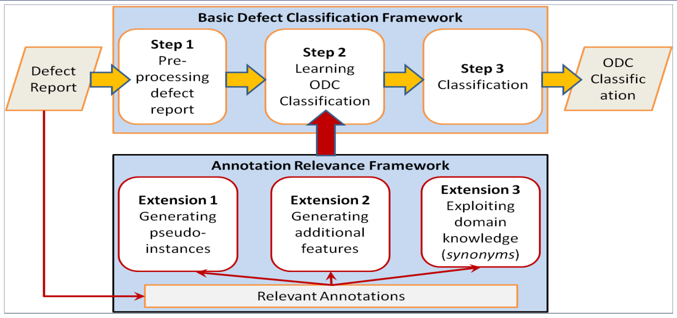
Why Relevant Annotations are Needed: Two Motivating Examples

ID	Summary	Description	Relevant words/Phrases
1	Prevent (last) Admin User from self-destruction	My client somehow managed to delete himself, meaning that no admin users continued to exist, so no new accounts - admin or otherwise - could be created, and no-one could log into the site! Fortunatley the site was in its infancy, so a simple database restore got it up and running again. But imagine the situtaion if a larger/active site had not been backed up and all the admins had been deleted :-(In principle this shouldn't be possible, but is in fact permitted by the current Elgg core implementation, which provides no "guard code" to ensure that an admin user cannot self-delete, self-ban or self-remove admin. The attached code diffs provide that necessary functionality, and it would be very useful to see these appear in the forthcoming Elgg 1.7 release, so that the update won't overwrite the code changes I have made to my client sites.	provide, necessary functionality, would be useful
2		getting 404 after "system settings" this is a fresh install of elgg and after I hit the "save" button on the "system settings" page I get 404 with the message "Oops! This link appears broken." The link in the address bar points to "http://cricmate.com/action/systemsettings/install" Can someone please help me out there? Thanks.	getting 404, link appears broken

Example 1: Weakened Knowledge. Long defect description lessens the impact of the important signature, and can potentially cause this report to be misclassified. The phrase "would be" in the last sentence is a strong indicator of the ***Requirements*** category (e.g., "would be useful", "would be great", "would be cool"), which implies a desired requirement currently missing from the system. Employing "would be" as a relevance annotation highlights its importance, allowing the learning algorithm to focus on this piece of evidence in spite of the long report and correctly classify this example.

Example 2: Confusing Information. Though the presence of the word "install" is evidence that it may belong to the ***Installability*** category, it is actually an example of ***Reliability***. Exploiting relevance annotations for automated classification could avoid this mistake: the phrase "link appears broken" was marked as evidence that it is an example of ***Reliability***.

APPROACH: *AutoODC* Overview


Basic Defect Classification Framework
Defect Report
Step 1 Pre-processing defect report
Step 2 Learning ODC Classification
Step 3 Classification
ODC Classification
Annotation Relevance Framework
Extension 1 Generating pseudo-instances
Extension 2 Generating additional features
Extension 3 Exploiting domain knowledge (synonyms)
Relevant Annotations

Basic Defect Classification Framework

Step 1: Pre-processing defect reports

- Tokenize and stem with the WordNet lexical knowledge base
- Use unigrams as features and represent each defect record as a binary-valued vector
- Normalize each vector to unit length

Step 2: Learning ODC classification

- Use Support Vector Machine (SVM) for classifier training
- One-versus-others training scheme to train one SVM classifier for predicting each class

Step 3: Classification

- Apply each trained classifier separately to classify an instance.
- Each classifier returns a confidence value. We assign to a defect record the class whose classifier returns the highest value among the set of values returned by all the classifiers.

Annotation Relevance Framework

Extension 1: Generating pseudo-instances

- **Goal:** Augment the training set with additional positive and negative training instances known as pseudo-instances. To create a pseudo-instance, we
  - *remove from the defect report one or more relevant annotations*
  - *create a feature vector consisting of the remaining unigrams in the report*
- **Observation:** Since a pseudo-instance lacks one or more relevant annotations, the correct SVM classifier should be less confident about its classification than a non-pseudo instance.
  - *We implement this observation by creating additional inequality constraints in SVM's optimization problem.*

Extension 2: Generating additional features

- **Goal:** Exploit the relevant annotations to create additional features for training.
- **Method:** To reduce data sparseness, instead of using a relevant annotation directly as a feature, we create all possible bigrams (i.e., consecutive words of length two) from each relevant annotation and use them as additional features.

Extension 3: Exploiting domain knowledge

- **Goal:** Exploit human-supplied domain knowledge to create additional training features.
- **Method:**
  - Collect all relevant annotations from the training defect records. Have a human analyst partition them so that each cluster contains all and only synonymous relevant annotations.
  - Assign a unique *ID* to each cluster.
  - If a relevant annotation is present in the defect record, we create an additional feature that corresponds to the *ID* of the cluster containing the relevant annotation.

EVALUTION

Experiment Setup

AutoODC is experimented on classifying defect records under the ODC "***Impact***" attribute. Combinations of basic defect classification system with the 3 extensions were experimented.

Data Set

Industrial defect report: **403** defect records in a social network project domain from an industrial Company

Training/testing data preparation:

- Two expert analysts independently classified the 403 defect records into 6 categories under the "***Impact***" attribute with an initial agreement of 90% and then cross-validated to resolve the disagreements.
- Distribution over the 6 categories: ***Capability (284)***, ***Security (11)***, ***Performance (1)***, ***Reliability (8)***, ***Requirements (39)***, ***Usability (60)***
- Both experts marked the words/phrases relevant for their assigning a defect record to a particular ODC category and identified the synonyms in these relevant words/phrases.

Evaluation Methodology: 5-fold cross-validation

Evaluation Results (Partial)

RQ: To what extent does the annotation relevance framework help improve ODC defect classification? Are the 3 extensions all contributing positively to overall performance?

Experiment	Accuracy	Reliability			Capability			Integrity/Security			Usability			Requirements		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Basic	74.2	0.0	0.0	0.0	77.6	94.0	85.0	75.0	27.3	40.0	48.9	36.7	41.9	70.0	17.9	28.6
Basic+Ext 1	76.9	0.0	0.0	0.0	80.5	93.3	86.5	71.4	45.5	55.6	60.9	46.7	52.8	57.1	30.8	40.0
Basic+Ext 2	76.4	0.0	0.0	0.0	79.1	94.4	86.0	87.5	63.6	73.7	56.8	41.7	48.1	66.7	20.5	31.4
Basic+Ext1,2	78.9	100.0	12.5	22.2	81.1	95.1	87.5	71.4	45.5	55.6	70.7	48.3	57.4	61.9	33.3	43.3
Basic+Ext1,2,3	80.2	100.0	12.5	22.2	82.8	95.1	88.5	77.8	63.6	70.0	73.3	55.0	62.9	54.5	30.8	39.3

CONCLUSIONS

- *AutoODC* produces satisfactory classification results with an accuracy of 80.2% when using manual defect classification as a basis of evaluation, where accuracy is computed as the percentage of defect records correctly classified by our classification system.
- As a complement to the manual ODC classification, *AutoODC* improves the confidence of defect classification results by reducing the investigation set that a human analyst has to examine when performing ODC classification.