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]

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

Dallas, TX, USA

{lghuang}@smu.edu[1]
{vince}@hlt.utdallas.edu[2]

Introduction

- Systematic classification and analysis of defect data bridge the gap between causal analysis and statistical quality control
  - provide valuable in-process feedback
  - improve system and software quality
- Orthogonal Defect Classification (ODC) – the most influential framework for software defect classification and analysis
- Issues with manual ODC generation
  - Manual ODC defect classification based on assimilation of defect repositories is extremely effort consuming esp. for novices.
  - Types of ODC-based defect analyses are often restricted by the limited defect classification results.
  - Our solution: ***AutoODC*** (Automating the Generation of Orthogonal Defect Classifications)

Major Contributions of *AutoODC*

- **Software Engineering:** Semi-automate ODC defect classification and evaluate our approach on the ODC "Impact" attribute.
  - *AutoODC* improves the confidence of defect classification results by reducing the investigation set of human analysts
- **AI**: 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.

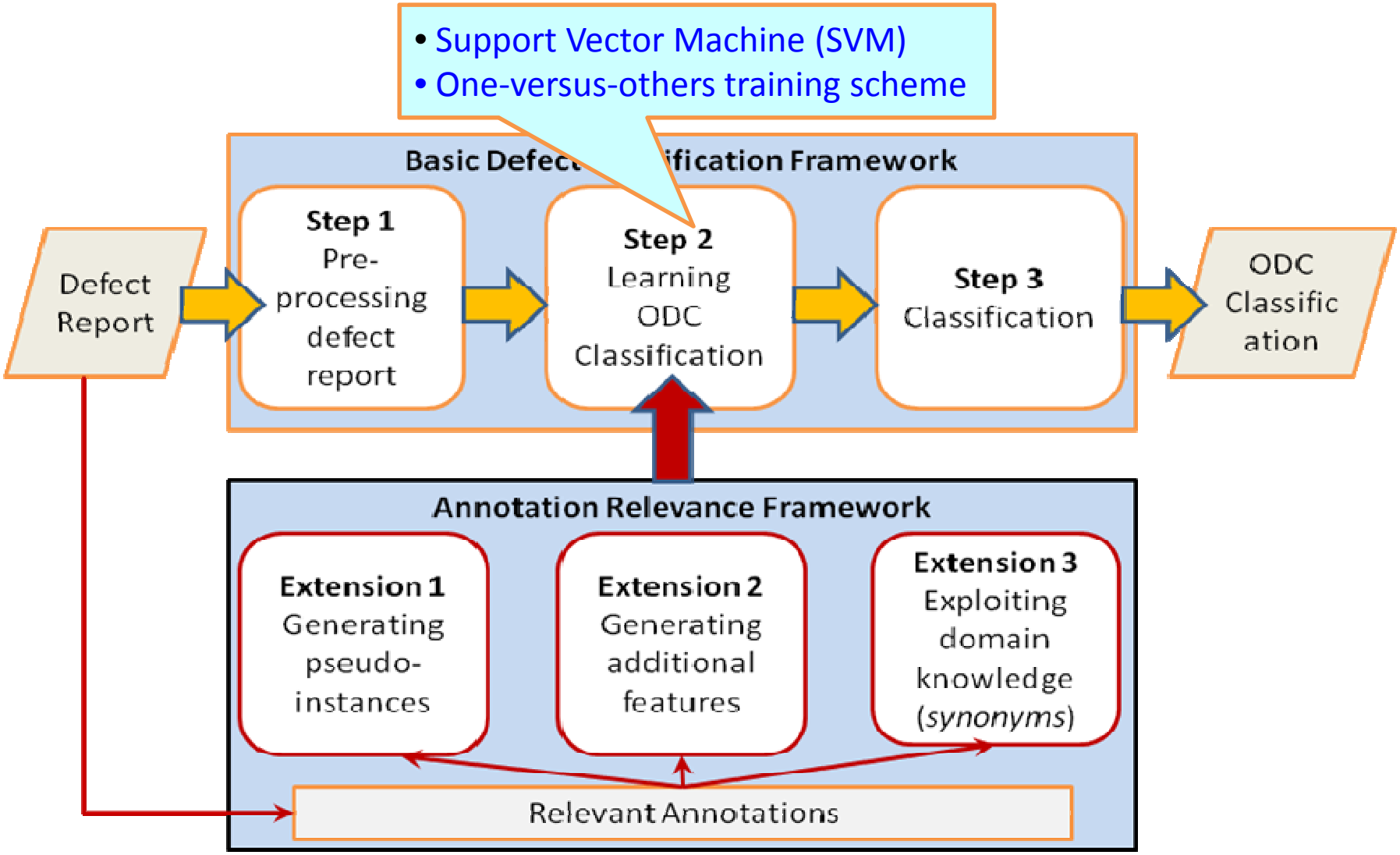
Why Relevant Annotations Needed

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

Weakened knowledge

Confusing Information

AutoODC Approach



Evaluation

- ***Experiment Setup:*** *AutoODC* is experimented on classifying defect records under the ODC "***Impact***" attribute.
- ***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.
    - Distribution over the 6 categories: ***Capability (284), Security (11), Performance (1), Reliability (8), Requirements (39), Usability (60)***
- ***Evaluation Methodology:*** 5-fold cross-validation
- **Classification Accuracy**: 80.2% when using manual defect classification as a basis of evaluation
  - **Accuracy**: the percentage of defect records correctly classified by our classification system.

Backup Charts

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:** the correct SVM classifier is less confident about its classification of a pseudo instance than a non-pseudo instance.
  - **Implementation:** create 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, create all possible bigrams (i.e., consecutive words of length two) from each relevant annotation 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. Human analyst partitions 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.