Using the Architectural Decay Prediction Scripts and Data

This README file describes the prediction model scripts and data sets used for prediction.

Basic Requirements

You will need R installed with the Rscript command. The scripts have been built and tested on a Mac and utilize gsed (i.e., GNU sed). If you do not have a Mac, you will need to replace gsed with the appropriate GNU sed command (e.g., sed without the "g" in front of it).

Top-Level Directory Structure

The directories within PerRelease/ are structured as follows:

```
PerRelease
|
└──Plots
└──ProjectsData
```

- PerRelease/Plots contains R scripts, data as CSV files, and generated PNG files with box plots illustrating the data
- PerRelease/ProjectsData contains a variety of data for generating overall CSV data sets in PerRelease/Plots; individual R scripts containing prediction models; data for individual combinations of projects, releases, and recovery techniques; and output that we obtained running our experiments

Root of PerRelease/ProjectsData

The root of PerRelease/ProjectsData/ contains the following:

```
PerRelease/ProjectsData
|
|-- ./numberOfModules.csv # number of modules across all releases, projects, and reco
very techniques
|-- ./numberOfModules.R # the script for generating the above CSV file
|-- ./create_cf_spearman_csv.sh  # create CSV for spearman correlation and cluster fa
```

```
ctor
|-- ./create_cf_auc_csv.sh # create CSV for AUC and cluster factor
|-- ./create_df_auc_csv.sh # create CSV for architectural defects and AUC
|-- ./create_df_spearman_csv.sh # create CSV for spearman correlation and architectur
al defects
|-- ./create_smells_csv.sh # create CSV for architectural smells
|-- ./create_smell_em_csv.sh # create CSV for smell emergence
|-- ./run_all_co_prediction.sh # run all prediction models for concern overload
|-- ./run_all_sf_prediction.sh # run all prediction models for scattered functionalit
y
|-- ./run_all_dc_prediction.sh # run all prediction models for dependency cycle
|-- ./run_all_lo_prediction.sh # run all prediction models for link overload
|-- ./run_all_co_em_prediction.sh # run all prediction models for concern overload em
ergence
|-- ./run_all_sf_em_prediction.sh # run all prediction models for scattered functiona
lity emergence
|-- ./run_all_dc_em_prediction.sh # run all prediction models for dependency cycle em
ergence
|-- ./run_all_lo_em_prediction.sh # run all prediction models for link overload emerg
ence
|-- ./run_all_df_prediction.sh # run all prediction models for architectural defects
|-- ./run_all_cf_prediction.sh # run all prediction models for cluster factor
|-- ./cf_auc.csv # data set for AUC and cluster factor
|-- ./cf_auc.png # box plots for AUC and cluster factor
|-- ./cf_spearman.csv # data set for cluster factor and spearman correlation
|-- ./cf_spearman.png # box plots for cluster factor and spearman correlation
|-- ./smells.csv # data set for predicted smells (non-emergence)
|-- ./df_auc.csv # data set for architectural defects and AUC
|-- ./df_spearman.csv # data set for architectural defects and spearman correlation
|-- ./arc_defect_count_per_release.csv # data set with the number of defects across r
eleases, projects, and ARC
|-- ./pkg_defect_count_per_release.csv # data set with the number of defects across r
eleases, projects, and packages
|-- ./defects_per_release.R # generates box plots showing the number of defects acros
s releases, projects, and recovery techniques
|-- ./defects_per_release.csv # data set with the number of defects across releases,
projects, and recovery techniques
|-- ./defects_per_release.png # box plots showing the number of defects across releas
es, projects, and recovery techniques
|-- ./arc_smell_count_per_release.csv # data set with the number of smells across rel
eases, projects, and ARC
|-- ./pkg_smell_count_per_release.csv # data set with the number of smells across rel
eases, projects, and packages
|-- ./smells_per_release.R # generates box plots showing the number of smells across
releases, projects, and recovery techniques
```

```
|-- ./smells_per_release.csv # data set with the number of smells across releases, pr
ojects, and recovery techniques
|-- ./smells_per_release.png # box plots showing the number of smells across releases
, projects, and recovery techniques
```

Workflow for Replication

When executed, the run_all*_prediction.sh script will descend into the sub-directories of the projects and recovery technique, run the individual PredictingSpecificRelease*.R scripts, and produce the *.[dependent variable abbreviation].out files corresponding to each PredictingSpecificRelease*.R script. Note that each run_all*.sh script corresponds to a different dependent variable to be predicted. For example, consider the following simplified directory contents for the Apache Camel project recovered using ARC:

```
|-- ./CamelARC
|   |-- ./CamelARC/2.8.3.lo_em.out
|   |-- ./CamelARC/2.8.3
|   |   |-- ./CamelARC/2.8.3/PredictingSpecificReleaseLOsmellEmergence.R
|   |   |-- ./CamelARC/2.8.3/TestData.csv
|   |   `-- ./CamelARC/2.8.3/TrainingData.csv
```

run_all_lo_em_prediction.sh will execute PredictingSpecificReleaseLOsmellEmergence.R, among other R scripts, using TrainingData.CSV as the training set and TestData.csv as the test set, to produce 2.8.3.lo_em.out. Every [Project Name][Recovery Technique]/[Version X.Y.Z]/ directory has a TrainingData.csv file containing the data for releases prior to version X.Y.Z and TestData.CSV has the data for version X.Y.Z. In the example above, TrainingData.csv has data for versions 1.6.0 ,2.0.M ,2.2.0 ,2.4.0, 2.5.0, 2.6.0, 2.7.1, and 2.8.0; TestData.csv has data for version 2.8.3.

The create_smell_em_csv.sh script will post-process the *.[smell]_em.out files to create a CSV file written to standard output, e.g., SmellEmergence.CSV, which in turn can be saved to Plots/SmellEmergence.csv. Executing Plots/SmellEmergence.R—after first changing the lines for change<-read.csv() and ggsave(file="") in that script to your PerRelease/ directory—will allow generation of Plots/SmellEmergence.png. The rest of the run_all*.sh, create*csv.sh, and Plots/*.R scripts follow the same workflow.

Data for a Recovery Technique and Project

Each [Project Name][Recovery Technique]/ directory (e.g., HBaseARC), contains a set of files similar to the following:

```
|-- ./HBaseARC
|   |-- ./HBaseARC/hbase-0.1.0-2008-03-28-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.1.3-2008-06-27-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.18.0-2008-09-21-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.19.0-2009-01-18-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.19.3-2009-05-21-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.20.2-2009-11-10-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.89.20100621-2010-06-25-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.89.20100924-2010-10-05-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.90.2-2011-03-27-ArcLowLevel.txt
|   |-- ./HBaseARC/hbase-0.90.4-2011-07-24-ArcLowLevel.txt
|   |-- ./HBaseARC/ResultsMetrics.txt
|   |-- ./HBaseARC/hbase-0.92.0-2012-01-23-ArcLowLevel.txt
|   |-- ./HBaseARC/TrainingUpTo-0.92.0.txt
|   |-- ./HBaseARC/TrainingUpTo-0.90.4.txt
|   |-- ./HBaseARC/TrainingUpTo-0.90.2.txt
```

The [project name]-[version info]-[recovery technique].txt (e.g., the first 11 files above). shows the data for each release. Each row shows a module and its corresponding data: (26 columns)

ModuleName IMC CMC NCF Defects NC LOC CO SF DC LO CBO DIT LCM SCC CMD OMD IMD XMD TCMD TOMD CO_NextRelease SF_NextRelease DC_NextRelease LO_NextRelease CF_NextRelease

Basically for each module, we have the values of 19 independent variables for the current release (k) and 6 dependent variables for the next release (k + 1) which are defects, CO_NextRelease, SF_NextRelease DC_NextRelease, LO_NextRelease and CF_NextRelease.

ResultsMetrics.txt contains the results of all the versions together.

The TrainingUpTo-[version info].txt files have the training data up to a certain release. For example, in the case of HBase, we have the results for the last three releases. TrainingUpTo-0.90.2.txt has the data for the first 8 releases. In the 0.90.2 directory, the TrainingData.csv file includes the data of TrainingUpTo-0.90.2.txt. The TestData.csv file includes the data in hbase-0.90.2-2011-03-27-ArcLowLevel.txt. As another example, in the case of the 0.90.4 directory, the TrainingData.csv file includes the data of TrainingUpTo-0.90.4.txt and TestData.csv is equivalent to hbase-0.90.4-2011-07-24-ArcLowLevel.txt.