Data Set for Enhanced Predicting the Performance of ATL Model Transformations

Raffaela Groner, Peter Bellmann, Stefan Höppner, Patrick Thiam,
Friedhelm Schwenker, Hans A. Kestler, and Matthias Tichy
<first name>.<last name>@uni-ulm.de

In the following we document the individual artifacts of this data set. With the help of these artifacts, the results of our experiments can be replicated and analyzed. We also document the changes in the ATL transformation modules that were necessary to execute them. Finally, we describe the naming scheme we used in the data to allow traceability of each input model used in our experiments and document the origin of the models used.

Structure of our provided data set that is also required to run our experiments:

```
<arbitrary directory>
        |--Algorithm
        |       | ml_experiemntsWithAttributes.py
        |--ATLModules
        |       |--lib
        |       |       | EMOOSE4EMF.atl
        |       |       | FLAME4EMF.atl
        |       |       | MOOD4EMF.atl
        |       |       | QMOOD4EMF.atl
        |       | EMF2Measure.atl
        |--DataBackup
        |       | predictionResults_RealWithAttributes<machine learning approach (LR|RF|SVR)>_<Module
                     name>_<Feature set name (FS1-8)>?_<Variance used to filter the feature set
                     (VAR0|Var75|Var85|VAR95|VAR99)>.csv
        |--EclipsePlugin
        |       |--JAR
        |       |       | performance.prediction_2.0.0.202205151504.jar
        |       |       | performance.statistics_2.0.0.202205151504.jar
        |       |       | performance.ui.menu_2.0.0.202205151504.jar
        |       |--Projects
        |       |       | performance.prediction_2.0.0.202205151504.zip
        |--InputData
        |       | <Module name>TimeDataRounded<*>(ALL|FS1-8).csv
        |--RawData
        |       |--<Module name>
        |       |       |--RawDataWithStringAttributes
        |       |       |       | PredictionExecutionData(ALL|FS1-8).csv
        |       |       |--TimeData
        |       |       |       | timeMeasure(PC2|AfterFix1|1-14).csv
        |--Results
        |       | predictionResults_RealWithAttributes<machine learning approach (LR|RF|SVR)>
                     _<Module name>_<Feature set name (FS1-8)>?_<Variance used to filter the
                     feature set (VAR0|Var75|Var85|VAR95|VAR99)>.csv
        |--Scripts
        |       | calculateStatisticsOverAllModules.py
        |       | calculateStatistics.py
        |       | generateInputData.py
```

1. Artifacts

In the following, we document the individual artifacts needed to perform our experiments, as well as the resulting ones.

- Algorithm/ml_experiemntsWithAttributes.py
  - Script to run our experiments. The script trains a linear regression, a random forest, and a support vector regression with a radial basis function kernel, and uses the obtained regression model to predict the execution time. The script uses the files in “InputData” as input, which describe the different feature sets. ml_experiemntsWithAttributes.py runs the experiments using 10-fold cross-validation.
  - In our experiments we used 45 different feature sets. Eight individual feature sets were defined by us and one additional feature set is the union of all our feature sets. We use five different thresholds for the variance of the features to create overall 45 feature sets out of our nine feature sets using Variance Thresholding. The thresholds used for filtering the variance are 0 (VAR0), the 75th quantile (VAR75), the 85th quantile (VAR85), the 95th quantile (VAR95) and the 99th quantile (VAR99). The following Table summarizes the feature sets defined by us:

No.	Feature Set
1	Number of model elements (only objects, no attributes or references) and ∅ size of string attributes per model element type per attribute
2	Number of model elements, number of references, and ∅ size of string attributes per model element type per attribute
3	Number of model elements, number of references, number of attributes, and ∅ size of string attributes per model element type per attribute
4	Number of model elements per type and ∅ size of string attributes per model element type per attribute
5	Number of model elements per type, ∅ Fan-In per model element type, ∅ Fan-Out per model element type, and ∅ size of string attributes per model element type per attribute
6	Number of model elements per type, ∅ Fan-In per model element type, ∅ Fan-Out per model element type, number of attributes per attribute type, and ∅ size of string attributes per model element type per attribute
7	Number of model elements per type, ∅ Fan-In per model element type and per reference type, ∅ Fan-Out per model element type and per reference type, and ∅ size of string attributes per model element type per attribute
8	Number of model elements per type, ∅ Fan-In per model element type and per reference type, ∅ Fan-Out per model element type and per reference type, number of attributes per model element type and per attribute type, and ∅ size of string attributes per model element type per attribute

  - **Input:** Expects the input data for each module and feature set to be located in folder “../InputData”. The names of the csv files must match following pattern: <Module name>TimeDataRounded<*>(ALL|FS1-8).csv

- **Output:** Results of the experiments, are stored in the folder "Results/". These files documents for each experiment the real time values and the predicted ones. The result files are named according to the following scheme: predictionResults_RealWithAttributes<machine learning approach (LR|RF|SVR)> _<Module name>_<Feature set name (FS1-8)>?_<Variance used to filter the feature set (VAR0|Var75|Var85|VAR95|VAR99)>.csv
- **Requirements:** Python with scikitlearn and panda

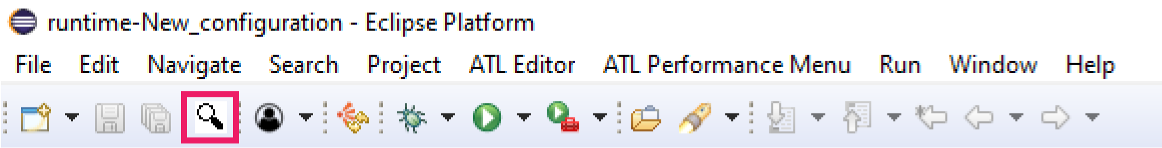
- ATLModules
  - Contains the ATL module EMF2Measure.atl used in our experiments that we implemented. The remaining modules are available on the ATL Transformation Zoo website (cf. list below).
  - In the following we list the sources for the individual modules. Necessary changes to the original modules referenced here are documented in Section 3.
    - ATL2Debugger.atl
      https://www.eclipse.org/atl/atlTransformations/#ATL2BindingDebugger
    - ATL2Problem.atl
      https://www.eclipse.org/atl/atlTransformations/#ATL2Problem
    - ATL2Tracer.atl
      https://www.eclipse.org/atl/atlTransformations/#ATL2Tracer
    - EMF2KM3.atl
      https://www.eclipse.org/atl/atlTransformations/#EMF2KM3
    - Make2Ant.atl
      https://www.eclipse.org/atl/atlTransformations/#Make2Ant
    - EMF2Measure.atl is a reimplementation of KM32Measure (https://www.eclipse.org/atl/atlTransformations/#KM32Measure)

- ATLModules/lib
  - Contains the additional libraries that the ATL module EMF2Measure.atl requires.

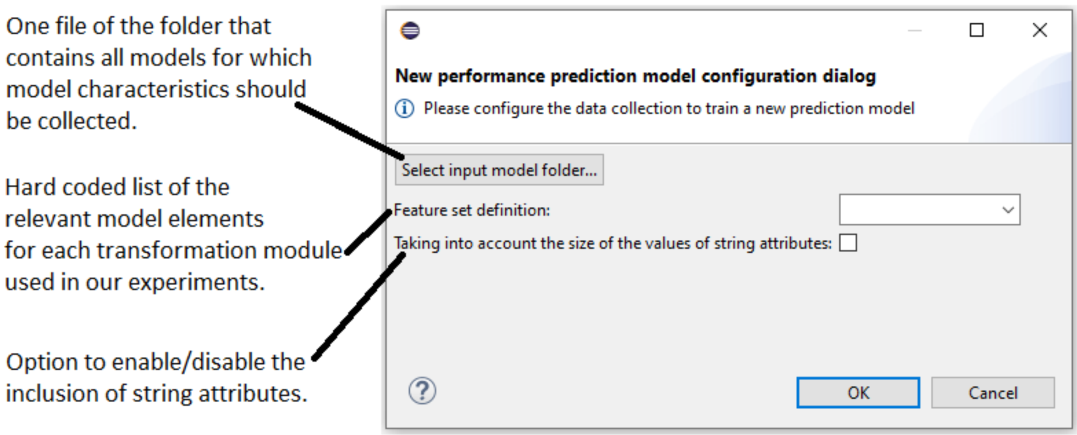
- DataBackup
  - This folder contains a backup of the results of our experiments since running the python scripts will overwrite them. The data follows the same structure and naming scheme as already described for the folder "Results".

- EclipsePlugin/Projects
  - Contains the exported Eclipse projects of our Eclipse plugin, that we used to collect the model characteristics and create the feature sets.
  - **Requirements:** Java 8, Eclipse Modelling Tools Photon-R with ATL installed

- EclipsePlugin/JAR
  - Contains the jar files to install our Eclipse plugin, that we used to collect the model characteristics and create the feature sets.
  - **Note:** The plugin is completely independent of ATL modules. We hardcoded the information about the relevant types of model elements into the plugin for the modules used in the experiments. But one can collect the data for the features for other modules via the feature set definition *"all model elements"* for all types of model elements.
  - **Requirements:** Java 8, Eclipse Modelling Tools Photon-R with ATL installed
    - The requirement that ATL must be installed is because we need ATL specific code in our plugin to load ATL modules as input models. No transformations are executed when using our Eclipse plugin.
  - **Use:**
    - Ensure that the ecore metamodel related to the input models is registered, else the model data cannot be collected.
      - *Tip:* If the plugin Epsilon 2.4 is installed via the Marketplace, it is possible to right-click on an ecore model and select "Register EPackages" to register it.
    - Start the selection by clicking on the small magnifying glass icon.



    - Confirm the dialog to reach the configuration dialog.
    - Select one input model, and the set of relevant model elements. Currently, the sets of relevant model elements we use in our experiments are hard coded and can only be controlled by their selection. Note that the plugin collects the data for all models located in the folder with the initially selected model. Please remove all other files from the respective folder beforehand.



    - After confirming the configuration, the data for the feature sets are collected and the results are stored in the tmp folder of the system.

InputData

- Contains one file for each module and each feature set, which is used as input for the machine learning approach.
- **Naming scheme:** <Module name>TimeDataRounded<*>(ALL|FS1-8).csv
- **Columns of the csv file:**

ModelName	NoAllObjects	<Features of the feature set>	TimeMS

  - ModelName = Name of the input model
  - NoAllObjects = Number of all model elements in the input model independent of their type
  - <Features of the feature set> = One column for each feature in the respective feature set
  - TimeMS = The measured execution time in milliseconds
- **Obtained:** Scripts/generateInpuData.py

- RawData
  - Contains for each transformation module a folder that contains the raw data. The raw data consists of the model characteristics for the feature sets (cvs files in the folder “RawDataWithStringAttributes”) and the measurement results of the time measurements (in folder “TimeData”).
  - **Columns of the csv file:**

ModelName	NoAllObjects	<Features of the feature set>

    - ModelName = Name of the input model
    - NoAllObjects = Number of all model elements in the input model independent of their type
    - <Features of the feature set> = One column for each feature in the respective feature set
  - **Obtained:** The time measurements were conducted using Java Microbenchmark Harness and the collection of model characteristics was performed with the help of our Eclipse plugin.

- Results/
  - Contains for each combination of module, machine learning approach and feature set a csv file which contains the measured execution times and the corresponding predicted ones.
  - **Naming scheme:** predictionResults_RealWithAttributes<machine learning approach (LR|RF|SVR)> _<Module name>_<Feature set name (FS1-8)>?_<Variance used to filter the feature set (VAR0|Var75|Var85|VAR95|VAR99)>.csv
  - **Columns of the csv file:**

RealValue	PredictedValue

    - RealValue = Measured execution time
    - PredictedValue = Predicted execution time
  - **Obtained:** Algorithm/ml_experiemntsWithAttributes.py

- Scripts/calculateStatistics.py
    - Calculates the MAPE in % and the P95(APE) in % for all results that are in the folder “Results/”
    - **Input:** Expects its input data in the folder “Results/”. The files must be csv files, containing of the columns “RealValue” (measured time) and “PredictedValue" (prediction results). These files are created by Algorithm/ml_experiemntsWithAttributes.py.
    - **Output:** Results/experiemntsWithAttributesPerFeatureSet.txt
    - **Requirements:** Python3 with scikitlearn and numpy version>1.22.0

- Scripts/calculateStatisticsOverAllModules.py
    - Calculates the MAPE in % and the P95(APE) in % over all modules except *ATL2Tracer* based on the data in the folder “Results/ExperimentsWithAttributes”
    - **Input:** Expects its input data in the folder “Results/ExperimentsWithAttributes”. The files must be csv files, containing of the columns “RealValue” (measured time) and “PredictedValue" (prediction results). These files are created by Algorithm/ml_experiemntsWithAttributes.py and must follow the specified naming scheme.
    - **Output:** Results/experiemntsWithAttributesPerFeatureSet.txt
    - **Requirements:** Python3 with scikitlearn and numpy version>1.22.0

- Scripts/generateInputData.py
    - Processes the raw data and stores the results in the folder “InputData”. The module names and the number of decimal places for rounding the execution time are hardcoded in the script.
    - **Input:** Expects the raw data in the folder “RawData/”<module name>. The files containing the results from the time measures must be contained in a folder “TimeData”. The model characteristics used in the feature sets must be contained in the folder “RawDataWithStringAttributes”.
    - **Output:** Prepared input data used for the experiments and stores the data in “InputData”. The output files are named according to the following scheme: <Module name>TimeDataRounded<Feature set name (ALL|FS1-8)>.csv
    - **Requirements:** Python

2. Replication of the experiments

In order to perform our experiments, the scripts must be executed in the following order. Please run **all scripts** always in their respective folders to ensure that the individual paths can be found. Please note that running these scripts overwrites the old data and that the execution of ml_experimentsWithAttributes.py takes several days:

1. Generate input data from raw data: Scripts/generateInpuData.py
2. Run Experiments: Algorithm/ml_experiemntsWithAttributes.py
3. Analyze results: Scripts/calculateStatistics.py and Scripts/calculateStatisticsOverAllModules.py

3. Changes in the ATL transformation modules

In the following, we document the changes in the ATL transformation modules from the ATL transformation zoo[1] that were necessary to execute them. In the context of our experiments it is only necessary to adjust the modules and execute the transformations if one wants to repeat the time measurements. The collection of data for the feature sets with our Eclipse plugin, as well as the execution of our experiments are independent of the modules.

- *ATL2Problem*:
    - Line 141 extended by the following if-statement:
      ```
      if self.outPattern.oclIsUndefined() then Sequence{}
      else self.outPattern.elements->asSequence() endif
      ```
    - All occurrences of ATL!Element replaced by ATL!LocatedElement
    - Line 241 & 242 commented out
    - Line 552 extended by the following if-statement:
      ```
      if s.definition.feature.oclIsUndefined() or e.definition.feature.oclIsUndefined()
      then false
      else s.definition.feature.name = e.definition.feature.name endif
      ```
    - Line 839 extended by the following if-statement:
      ```
      s : ATL!Parameter (
         if s.operation.oclIsUndefined() then false
           else s.operation.parameters ->exists(e | s.varName = e.varName and s <> e)
         endif )
      ```
- *ATL2Tracer*:
    - Line 20: elements <-Sequence {} replaced by elements <- sourceVars
    - Line 62-64: commented out
    - Line 90: elements <- targetVars replaced by elements <-Sequence {}
    - Line 92-94: commented out
- *EMF2KM3*:
    - Line 13 extended by t : KM3!Metamodel (
      ```
        location<-'1:1-175:2')
      ```
- *MAKE2ANT*:
    - Line 15 extended by:
      ```
      description <- if not m.comment.oclIsUndefined() then
            m.comment->iterate(c; res : String = '' | res + c.text+' ')
         else ' '
         endif
                                   ,
      ```

[1] https://www.eclipse.org/atl/atlTransformations/

4. Naming Scheme and Origin of the Models Used

In the following, we document the naming scheme used by us to trace back the origin of a model and reference the data sources used to obtain the models.

- We have coded in the csv files in "Data/RawData" and "Data/InputData" the origin of each model in their name with:
  <project_id>-<artefact_id>-<version_id>.ecore
  - **Data source:** The Ecore metamodels are selected from the data set "A Dataset of EMF Models from Eclipse Projects" from Kögel and Tichy, 2018 (available at: https://doi.org/10.18725/OPARU-9850).

- The Makefiles are obtained by mining GitHub. We have coded the origin of the Makefiles according to the following scheme:
  <full name of the repository (we replaced '/' with '_')>_<id> _Makefile.xmi
  - **Data source:** We mined GitHub via its API on the 22nd of April 2021 using our two scripts Scripts/MineGitHub/gitHubMineClone.py and Scripts/MineGitHub/gitHubMine.py to obtain Makefiles. We used the script provided by the Maven to Ant transformation from the ATL Transformation Zoo to format the Makefiles into suitable input models (Transfos/make2xml.sed in the source files of https://www.eclipse.org/atl/atlTransformations/#Make2Ant).

- The ATL transformations are selected from the ATL Transformation zoo. We have coded the origin of the Transformation according to the following scheme:
  <Transformation>_<optional additional information>_<name of the ATL file>.atl
  - **Data source:** We used the ATL modules provided by the ATL Transformation Zoo, which is available at https://www.eclipse.org/atl/atlTransformations/.