Large Experiment and Evaluation Tool for WEKA Classifiers

D. Baumgartner and G. Serpen
Department of Electrical Engineering and Computer Science, University of Toledo, Toledo, OH, USA

Abstract - This paper presents a new Windows®-based software utility for WEKA, a data mining software workbench, to simplify large-scale experiment and evaluation with many algorithms and datasets in the classification context. The proposed tool, LEET (Large Experiment and Evaluation Tool) makes it possible to accomplish a variety of tasks that are presently rather difficult or impractical through the standard WEKA interfaces. This includes allowing comparison of classifiers across multiple experiments, tracking execution time, calculating diversity measures, and summarizing the characteristics of many datasets. We have tested and validated LEET as part of a study with 50+ machine learning classification/ensemble algorithms, 46 datasets, and calculation of a variety of performance measures. With WEKA providing the algorithm implementations, LEET facilitates the execution and evaluation of large-scale experiments with greater ease than any existing interface.

Keywords: WEKA, large-scale experiment/evaluation, automation script, classification, diversity measure

1 Introduction

Large-scale machine learning and data mining experiments that process a significant number of algorithms through an equally large number of datasets pose unique challenges for efficient and effective management. Making large-scale experiments more controllable is of paramount interest since they typically tend to have very long run-times. Another aspect of interest is availability of tools to, at least partly, automate the entire experimentation process.

In machine learning and data mining, WEKA (Waikato Environment for Knowledge Analysis) provided the first comprehensive software workbench for research and development. Although it has been subjected to on-going improvements over the past decade, it is often necessary to develop custom scripts to facilitate experiments that may also require calculation of performance measures not readily available as part of the existing WEKA implementation. We present a Windows®-based GUI application, LEET (Large Experiment and Evaluation Tool), that, while using WEKA as the main computing engine, simplifies this process. The focus of LEET is on large-scale experiment and evaluation. Furthermore, LEET has some features that are difficult or impractical to perform within the existing WEKA interfaces.

The WEKA website lists over 50 related projects to the data mining workbench; however the utility of these are mostly limited to extensions/enhancements of existing algorithms, addition of new algorithms, interfaces for other software, or support for specific problem domains [8]. LEET offers features that are not only unique for WEKA, but also other 3rd-party software that we currently know of.

The remainder of this paper is organized as follows. First, WEKA and its interfaces are presented. Second, LEET is explained and its features are examined along with its interface. Finally, the conclusions for this paper are given.

2 WEKA: A machine learning toolkit

WEKA (Waikato Environment for Knowledge Analysis) is a machine learning and data mining software tool written in Java and distributed under the GNU Public License [9]. The goal of the WEKA project is to “build a state-of-the-art facility for developing machine learning techniques and to apply them to real-world data mining problems” [8]. It contains several standard data mining techniques, including data preprocessing, classification, regression, clustering, and association. Although most users of WEKA are researchers and industrial scientists, it is also widely used for academic purposes.

A brief discussion of WEKA is offered in this section. First, the background and history of the program are given, followed by a discussion of its interfaces, and finally a summary of the limitations of the interfaces in the context of large-scale experiment and evaluation of classifiers.

2.1 Background and history

The WEKA project has been funded by the New Zealand since 1993. The initial goal at this time was as follows [6]:

“The programme aims to build a state-of-the-art facility for developing techniques of machine learning and investigating their application in key areas of New Zealand economy. Specifically we will create a workbench for machine learning, determine the factors that contribute towards its successful application in the agriculture industries, and develop new methods of machine learning and ways of assessing their effectiveness.”

In 1996, a mostly C version of WEKA was released, while in 1999 it was redeveloped and released in Java to support platform independence. Today, there are three versions of
WEKA available to the public [8]. The GUI version (6.0) is the most recent release. The developer version (3.5.8) allows users to obtain and modify source code to add content or fix bugs. The book version (3.4.14) is as described in the data mining book released by Witten and Frank [9].

2.2 Interfaces

WEKA has four interfaces, which are started from the main GUI Chooser window, as shown in Fig. 1 (this discussion is based on WEKA version 6.0). The Explorer, Knowledge Flow, and Simple CLI (Command Line Interface) all handle data preprocessing, classification, regression, clustering, and association. The Experimenter handles only classification and regression problems. Each interface has different utilities and different benefits/detriments. Following, the features of each interface are described. This review focuses on the classification aspect, since that is the context of this paper.

2.2.1 Explorer

The Explorer is possibly the first interface that new users will run simulations in. It allows for data visualization and preprocessing. Execution of a classifier allows for only a single algorithm on a single dataset. Training and testing data can be either from the original dataset (exact data, cross-validation, or percentage split) or a different dataset. The output results can range from simple statistics like the percent of correctly classified instances to a textual or graphical model of the classifier, entropy evaluation measures, and a full listing of predictions for each instance.

2.2.2 Experimenter

The Experimenter allows a user to set up and execute a set of simulations. There is no data visualization or preprocessing, but many algorithms can be executed on many datasets. Training and testing data are limited to only cross validation and percentage split. The results of an experiment can easily be stored, but the user has no control over what the stored contents are.

The Experimenter only stores a specific set of performance measures. That is, the predictions for each instance are not saved. Although this makes storage of results more efficient, no new measures can be calculated since the predictions are not known. Analysis in the Experimenter places the results in a tabular manner (e.g. each row is a dataset and each column is a classifier), where the performance measure of interest can be changed. Furthermore, the paired t-test, which is a commonly accepted statistical significance test for comparing classifiers [4], is implemented and can be applied to the table. This allows for comparison of classifiers across multiple datasets.

![Fig. 1: The interfaces of WEKA. GUI Chooser at center, Explorer at top-left, Experimenter at top-right, Knowledge Flow at bottom-left, and Simple CLI (Command Line Interface) at bottom-right.](image-url)
There are a few usability issues that the authors have noticed with the Experimenter interface. First, the results of simulations are not saved to the hard drive until the entire experiment is complete (for the ARFF and CSV file types, which are the most common). If WEKA is closed prematurely or crashes before the execution of the entire set of simulations is complete, then all of the results are lost. When an experiment is expected to take a long time to complete because of a large number of classifiers and/or datasets, there is a risk of hours, days, or even weeks of simulations being lost. Splitting up experiments into smaller pieces makes the possibility of this occurring less likely, but it leads to the second issue with the Experimenter usability.

In a typical research undertaking, it may not always be possible to determine all simulations that need to be performed. For example, after running an initial set of experiments a researcher may find a new dataset that should be included in their analysis. The Experimenter allows for classifier comparisons within only a single experiment output file. Merging the outputs of different experiments is possible, although it is difficult, time-consuming and error prone. A more flexible alternative to comparing classifiers across experiments is desirable.

### 2.2.3 Knowledge flow

The Knowledge Flow offers a data-flow inspired interface for WEKA [2]. It is a graphical alternative to Explorer, although not all functionality from the Explorer is available in Knowledge Flow and vice-versa. For the purposes of this paper, no distinction is made between the two interfaces.

### 2.2.4 Simple CLI

The Simple CLI provides a command-line interface that allows direct execution of WEKA commands [2]. This is included within the GUI of WEKA for operating systems that do not provide their own command-line interfaces. Simple CLI provides full access to all WEKA features through calls to the Java classes, although it still has the memory limitations of the other interfaces. The output is similar to what can be obtained from the Explorer, and it can be easily piped to an external file for storage.

### 2.3 Executing large experiments with WEKA

Each interface has a specific purpose and utility. Whereas the Explorer and Knowledge Flow are tailored to beginning users, the Experimenter and Simple CLI target more advanced users. We believe that the Experimenter is too restricting on the data that it outputs (i.e. no predictions for each instance) and not scalable for very large sets of simulations (i.e. interruption before execution is finished will lead to loss of results). The Simple CLI has memory limitations and it is rather inefficient to use since each simulation must be manually entered in. The only other alternative is to use the operating system command-line to call WEKA.

The WEKA documentation reads, “while for initial experiments the included graphical user interface is quite sufficient, for in-depth usage the command-line interface is recommended, because it offers some functionality which is not available via the GUI – and uses far less memory” [2]. In the context of this statement, the command-line interface being referred to is of the operating system and not the Simple CLI. Instead of executing each simulation manually, a user can create a script to execute them sequentially.

Traditionally, researchers develop their own scripts that are catered to the specific needs of their work. Things such as folder/file structure, type of results to store, and calculation of performance measures need to be handled before simulation can begin, which can be costly with regards to time and money. Whereas the community supports WEKA with extensions to functionality (e.g. adding classifiers), there are no publicly available resources for interfaces to handle large-scale experiments [8].

### 3 Large experiment and evaluation tool for WEKA classifiers

We present a Windows®-based GUI application, named LEET (Large Experiment and Evaluation Tool), which facilitates execution and evaluation of large-scale classification experiments with WEKA.

1. Executes classification experiments using WEKA’s built-in classifiers.
2. Evaluates the executed experiments to obtain performance measures.
3. Evaluates datasets to calculate characteristics.

LEET is created in response to requirements formed during our research of classification ensembles and the limitations of the WEKA interfaces discussed earlier. Specifically, we require the ability to run large-scale experiments and collect the predictions for each instance so that performance measures not found in WEKA can be calculated. Traditionally, researchers must create static scripts that execute experiments and parse the result files. Instead, we have developed LEET as a GUI application that makes the experiment and evaluation process simpler.

The current implementation of LEET has been validated in versions 3.5.8 and 3.6.0 of WEKA. It can execute any classifier that WEKA allows and calculates the prediction accuracy average, prediction accuracy standard deviation, execution time, and diversity of classifiers.

---

1 [LEET](https://sourceforge.net/projects/largeexperiment/) is available at <https://sourceforge.net/projects/largeexperiment/>.
Furthermore, LEET offers a utility to calculate the characteristics (e.g. number of instances, classes, and features) for a large set of datasets; something that cannot be done in WEKA.

In the remainder of this section LEET is described. First, specifications of the environments in which LEET is developed, and can consequently run are presented. Second, the tasks that the program performs are elaborated upon. Finally, validation of LEET is offered through its use in our own large-scale research experiments.

3.1 Environment

The development of LEET was originally intended for in-house use only, so it was developed in the environment that we are most familiar with – C# in Visual Studio 2008. Consequently, the program requires the free .NET Framework 3.5, and unless third party software is used, LEET is restricted to Windows XP and Windows Vista.

LEEP is an application outside of the WEKA interface. This has two main benefits. First, as previously mentioned, using the operating system command-line to call WEKA uses memory more efficiently than running the traditional interfaces. Second, LEET can work across multiple versions of WEKA since there is no integration of code. The functionality of LEET depends only on the WEKA input and output. The same may not be true if the programs were integrated – the GUIs alone of the three current versions of WEKA would need separate implementations to support an additional interface such as LEET. The concept of support for multiple versions has already been validated, as LEET was originally developed for version 3.5.8 of WEKA, but has been verified to work without modification for the newest 3.6.0 version.

3.2 Features of LEET

The LEET application is shown in Fig. 2. There are three main areas in the window. The menu area at the top contains information about the program and the program settings. The settings contains properties such as the WEKA install directory, default directory for the experiments to be saved, and the maximum heap size of the Java virtual machine that WEKA runs on. The area at the bottom with the three tabs is where the experiments and evaluations are performed. The area between the prior two is for dataset selection. Since dataset selection is a common operation for all three tabs, having it at a single location helps to reduce the clutter and increase usability of the interface. The functionality of each tab is explained below.

3.2.1 Execute - Experiments

The Execute - Experiments tab of LEET is shown in Fig. 2. This component serves as a GUI for creating and executing command-line simulations in WEKA. There are two main text boxes for input. The user enters classifier configurations in “Classifiers”. These can be formed in the Explorer interface within WEKA or manually typed. Each classifier must have an alias associated with it, which is specified in “Classifier Alias”. Notice in Fig. 2 that there are three J48 classifier configurations and three classifier aliases. The aliases are used to build the directory structure that stores the simulation results.

The last parameter needed for an experiment is the number of folds in cross-validation. Although this is the only sampling method supported in LEET, it is the most common and accepted method in the literature [4]. The number of folds can be specified in the “k fold (cross-validation)” text box, and is defaulted to 10.

The “Execute” button starts the experiment. The flowchart for the experiment process is shown in Fig. 3. Each classifier is run on the specified datasets. After a simulation (i.e. a classifier and dataset pair) is complete, the execution time and prediction results are stored in files within a directory named by the classifier alias.

![Fig. 2: The Execute – Experiments tab of LEET.](image)

![Fig. 3: Flowchart for the execution of experiments in LEET.](image)
Execution times for simulations of a classifier are aggregated in a single file. Each line in this file specifies the execution time in “hours:minutes:seconds” format for a different dataset. Individual files store the prediction results for each simulation. These files are named by the dataset used in the simulation. A partial sample of a prediction results file is shown in Fig. 4. The first line shows the classifier configuration, while the second displays the date/time of the execution. The remainder of file contains the predictions as output by WEKA. Notice that the actual and predicted class information is provided. Furthermore, the final column specifies the probability of the prediction. This extra data facilitates calculations of more advanced measures than simply accuracy.

The performance measures supported in the current version of LEET include prediction accuracy average, prediction accuracy standard deviation, execution time, and diversity. This set of measures are required for our current research with classifier ensembles, and besides the prediction accuracy values, they are not calculated within WEKA. Since the actual predictions for each instance are stored rather than a pre-specified set of measures, additional measures can be calculated in the future without requiring simulations to be re-executed.

Diversity between classifiers is a concept of classification ensembles. Those included in LEET are Q-statistic, disagreement, weighted count of errors and correct, and exponential error count \[1,3\]. Furthermore, there are options to output the diversities in a matrix form, which can be used as input for a method of visualizing the diversity between classifiers in a two-dimensional space \[5\].

The results of the performance measure are output in a CSV format in the “Results” text box. This allows for easy copy-and-paste into a spreadsheet application, where further processing can be performed. Statistical significance tests have not been implemented within the current version of LEET because of their algorithmic complexity.

It is common for researchers to require simulations that are supplemental to their original experimentation (e.g. a new dataset is explored). These two sets of results need to be handled as one to allow for evaluation. While WEKA’s Experimenter requires a manual merging of result files to accomplish this, LEET makes the process simple. Any
classifiers that have been executed with LEET can be considered as a single group for evaluation.

3.2.3 Evaluate - Datasets

WEKA has the ability to visualize and give characteristics about datasets via the Explorer, but this can only be performed for one dataset at a time. Large research projects may contain 10’s or even 100’s of datasets, and it is customary in the literature to present some basic characteristics of each dataset [7]. LEET offers an alternative to WEKA that makes obtaining characteristics for many datasets simple.

The Evaluate – Datasets tab of LEET is shown in Fig. 6. The only parameter for this is choosing between number and percent for the feature outputs. For all of the selected datasets, the following characteristics are output to the “Results” text box: number of instances, number of classes, number/percent of unary features, number/percent of binary features, number/percent of nominal features, number/percent of numeric features, percent of majority classes, and percent of minority classes. Like in the experiment evaluation, the output is in CSV format which allows for easy exporting to a spreadsheet for formatting.

![Fig. 6: The Evaluate – Datasets tab of LEET.](image)

3.3 Large-scale experiments with LEET

LEET has played an important role in our current research with classification ensembles. It not only makes the process of experimentation and evaluation simpler, more controllable, and manageable but also provides an opportunity to verify the functionality of LEET. Our research endeavors require experiments with 50+ single-classifier and classification ensembles; each of which is executed on 46 datasets. To our knowledge, this is among the largest empirical studies in the literature.

Furthermore, we have used the experiment evaluation functionality within LEET to create groups of classifiers for which performance comparison is carried out. Since the output of LEET is in CSV format, exporting the data to other programs for further processing is trivial. For example, we have generated a spreadsheet that automatically calculates statistical significance between classifiers within the output generated by LEET.

4 Conclusions

Traditionally, data mining experiments are large and difficult to manage effectively. Researchers must create static scripts to execute and calculate performance measures. The same is true with WEKA, a popular and powerful data mining workbench. In this paper, we have presented LEET as an application for executing and evaluating large-scale classification experiments using WEKA.

LEET can execute any classifier within WEKA and stores the results in a manner to allow for the calculation any performance measures. LEET calculates the average and standard deviation of classifier prediction accuracy, but also includes execution time and diversity measures, which are not currently available in WEKA. Furthermore, LEET offers a utility to calculate the characteristics (e.g. number of instances, classes, and features) for a large set of datasets; something that cannot be done in WEKA.

The development of LEET has been concurrent with our research in classification ensembles. This has allowed for in-depth validation of the features of the application. LEET is a work-in-progress, and additional performance measures are likely to be added – especially as our research of classifiers encourages. Porting LEET to Java to allow for platform independence would be beneficial for the community. Another possible modification is to integrate LEET into WEKA, although being external does have benefits, as discussed in earlier in this paper.

5 References


