Journal of Machine Learning Research 11 (2010) 2533-2541 Submitted 6/10; Revised 8/10; Published 9/10

WEKA—Experiences with a Java Open-Source Project

Remco R. Bouckaert REMCO@CS.WAIKATO.AC.NZ
Eibe Frank EIBE@CS.WAIKATO.AC.NZ
Mark A. Hall MHALL@CS.WAIKATO.AC.NZ
Geoffrey Holmes GEOFF@CS.WAIKATO.AC.NZ
Bernhard Pfahringer BERNHARD@CS.WAIKATO.AC.NZ
Peter Reutemann FRACPETE@CS.WAIKATO.AC.NZ
Ian H. Witten IHW@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikatoi
Hamilton, New Zealand

Editor: Soeren Sonnenburg

Abstract

WEKA is a popular machine learning workbench with a development life of nearly two decades. This article provides an overview of the factors that we believe to be important to its success. Rather than focussing on the software's functionality, we review aspects of project management and historical development decisions that likely had an impact on the uptake of the project.

Keywords: machine learning software, open source software

1. Introduction

We present a brief account of the WEKA 3 software, which is distributed under the GNU General Public License, followed by some lessons learned over the period spanning its development and maintenance. We also include a brief historical mention of its predecessors.

WEKA contains implementations of algorithms for classification, clustering, and association rule mining, along with graphical user interfaces and visualization utilities for data exploration and algorithm evaluation. This article shares some background on software design and management decisions, in the hope that it may prove useful to others involved in the development of open-source machine learning software. Hall et al. (2009) give an overview of the system; more comprehensive sources of information are Witten and Frank's book *Data Mining* (2005) and the user manuals included in the software distribution.[1] Online sources, including the WEKA Wiki pages[2] and the API, provide the most complete coverage. The *wekalist* mailing list is a forum for discussion of WEKA-related queries, with nearly 3000 subscribers.

2. What is WEKA?

WEKA is a machine learning workbench that supports many activities of machine learning practitioners.

1. Available from `http://www.cs.waikato.ac.nz/ml/weka`.
2. Available at `http://weka.wikispaces.com/`.

2.1 Basic Functionality

Here is a summary of WEKA's main features.

- *Data preprocessing*. As well as a native file format (ARFF), WEKA supports various other formats (for instance CSV, Matlab ASCII files), and database connectivity through JDBC. Data can be filtered by a large number of methods (over 75), ranging from removing particular attributes to advanced operations such as principal component analysis.

- *Classification*. One of WEKA's drawing cards is the more than 100 classification methods it contains. Classifiers are divided into "Bayesian" methods (Naive Bayes, Bayesian nets, etc.), lazy methods (nearest neighbor and variants), rule-based methods (decision tables, OneR, RIPPER), tree learners (C4.5, Naive Bayes trees, M5), function-based learners (linear regression, SVMs, Gaussian processes), and miscellaneous methods. Furthermore, WEKA includes meta-classifiers like bagging, boosting, stacking; multiple instance classifiers; and interfaces for classifiers implemented in Groovy and Jython.

- *Clustering*. Unsupervised learning is supported by several clustering schemes, including EM-based mixture models, *k*-means, and various hierarchical clustering algorithms. Though not as many methods are available as for classification, most of the classic algorithms are included.

- *Attribute selection*. The set of attributes used is essential for classification performance. Various selection criteria and search methods are available.

- *Data visualization*. Data can be inspected visually by plotting attribute values against the class, or against other attribute values. Classifier output can be compared to training data in order to detect outliers and observe classifier characteristics and decision boundaries. For specific methods there are specialized tools for visualization, such as a tree viewer for any method that produces classification trees, a Bayes network viewer with automatic layout, and a dendrogram viewer for hierarchical clustering.

WEKA also includes support for association rule mining, comparing classifiers, data set generation, facilities for annotated documentation generation for source code, distribution estimation, and data conversion.

2.2 Graphical User Interfaces

WEKA's functionality can be accessed through various graphical user interfaces, principally the Explorer, and Experimenter interfaces shown in Figure 1, but also the Knowledge Flow interface. The most popular interface, the Explorer, allows quick exploration of data and supports all the main items mentioned above—data loading and filtering, classification, clustering, attribute selection and various forms of visualization—in an interactive fashion.

The Experimenter is a tool for setting up machine learning experiments that evaluate classification and regression methods. It allows easy comparison of performance, and can tabulate summaries in ways that are easy to incorporate into publications. Experiments can be set up to run in parallel over different computers in a network so that multiple repetitions of cross validation (the default method of performance analysis) can be distributed over multiple machines.

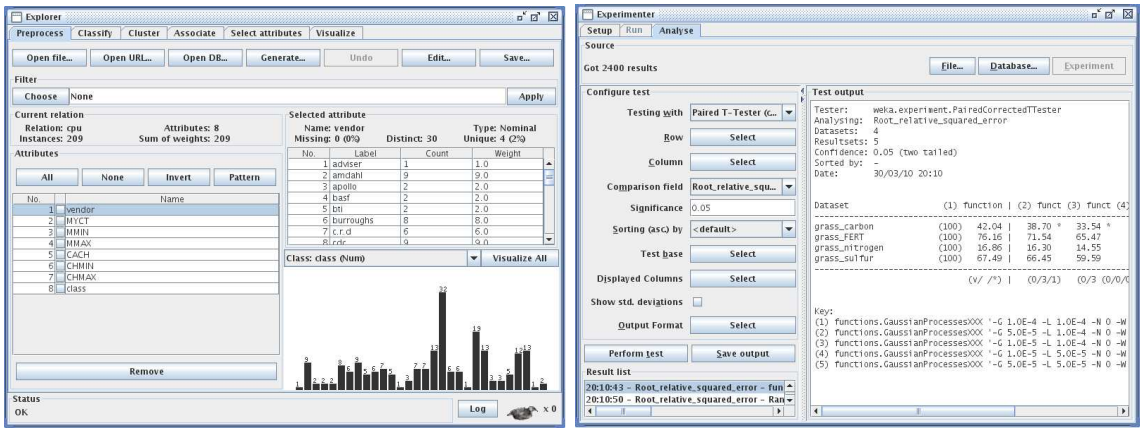

Figure 1: The Explorer and Experimenter interfaces.

The Knowledge Flow interface is a Java Beans application that allows the same kind of data exploration, processing and visualization as the Explorer (along with some extras), but in a workflow-oriented system. The user can define a workflow specifying how data is loaded, preprocessed, evaluated and visualized, which can be repeated multiple times. This makes it easy to optimize the workflow by tweaking parameters of algorithms, or to apply it to other data sources. In the Explorer, on the other hand, the individual steps must be invoked manually, one at a time. This is a rather tedious process, and is prone to errors such as omitting preprocessing steps.

WEKA also includes some specialized graphical interfaces, such as a Bayes network editor that focuses on Bayes network learning and inference, an SQL viewer for interaction with databases, and an ARFF data file viewer and editor.

All functionality and some more specialized functions can be accessed from a command line interface, so WEKA can be used without a windowing system.

2.3 Extending WEKA

One of WEKA's major strengths is that it is easily extended with customized or new classifiers, clusterers, attribute selection methods, and other components. For instance, all that is needed to add a new classifier is a class that derives from the `Classifier` class and implements the `buildClassifier` method for learning, and a `classifyInstance` method for testing/predicting the value for a data point. The code fragment in Figure 2 shows a minimal implementation of a classifier that returns the mean or mode of the class in the training set (double values are used to store indices of nominal attribute values).

Any new class is picked up by the graphical user interfaces through Java introspection: no further coding is needed to deploy it from WEKA's graphical user interfaces. This makes it easy to evaluate how new algorithms perform compared to any of the existing ones, which explains WEKA's popularity among machine learning researchers.

Besides being easy to extend, WEKA includes a wide range of support for adding functionality to basic implementations. For instance, a classifier can have various optional settings by implementing a pre-defined interface for option handling. Each option can be documented using a tool-tip text

```
package weka.classifiers.misc;

import weka.classifiers.Classifier;
import weka.core.*;

public class NewClassifier extends Classifier {
        double m_fMean;

        public void buildClassifier(Instances data) throws Exception {
                m_fMean = data.meanOrMode(data.classIndex());
        }

        public double classifyInstance(Instance instance) throws Exception {
                return m_fMean;
        }

}
```

Figure 2: Classifier code example.

method, which is picked up by the *help* dialogs of the graphical user interfaces. Classes typically implement methods described in papers, and this provenance metadata can be captured in a method that is used to generate documentation. Some methods only apply to certain kinds of data, such as numeric class values or discrete attribute values. A 'capabilities' mechanism allows classes to identify what kind of data is acceptable by any method, and the graphical user interfaces incorporate this by making methods available only if they are able to process the data at hand.

There are many projects that build on top WEKA, about fifty of which are listed on the WEKA Wiki.

3. Origins

The Machine Learning project at Waikato was launched in 1993 with a successful grant application to the New Zealand Foundation for Research, Science, and Technology. The underlying intention behind the request was not so much to further a specific agenda in machine learning research as to create a research culture in a small and obscure computer science department that brought different people together. Machine learning was selected because of prior expertise and its potential applicability to agriculture, New Zealand's core industry; the grant was justified in terms of applications research rather than the development of new learning techniques.

This gave the research team a license to incorporate and reimplement existing methods, and work soon began on a workbench, written in C, that was intended to provide a common interface to a growing collection of machine learning algorithms. It contained some learning algorithms written mostly in C, data I/O and pre-processing tools, also written in C, and graphical user interfaces written in TCL/TK. The number of learning algorithms was limited and they came from different sources; wrapper shell scripts were employed to bind them into the framework. The acronym WEKA for "Waikato Environment for Knowledge Analysis" was coined, and the system gradually became known in the international ML community, along with another machine learning library in C++ from the University of Stanford called MLC++, developed by Kohavi et al. (1997).[3]

3. MLC++ is now distributed by Silicon Graphics at `http://www.sgi.com/Technology/mlc`.

Because of dependencies on other libraries, mainly related to the graphical user interfaces, the software became increasingly unwieldy and hard to maintain. In 1997 work began on reimplementing WEKA from scratch in Java into what we now term WEKA 3. One of the authors, Eibe Frank, had earlier decided to adopt Java to implement algorithm prototypes, abandoning C++ because Java development was rapid and debugging was dramatically simplified. This positive experience, the promise of platform independence through virtual machine technology, and the fact that, as part of the research code, classes for reading WEKA's standard ARFF file format into appropriate data structures already existed, led to the decision to re-write WEKA in Java.

Many of the classes in today's code archive (including, for example, J48, the WEKA implementation of Quinlan's C4.5) date from mid-1997. Rapid early development was stimulated by the need to teach a course on machine learning at the University of Calgary during a Fall 1997 sabbatical visit by Witten, along with several students, including Frank. The Java version was called JAWS for "Java Weka system" to avoid confusion with WEKA itself, and after some debate was changed to WEKA in March 1999. The first paper written using this system was Frank and Witten (1998), written in late 1997. By the end of 1998 WEKA included packages for classifiers, association rule learners, filters, and evaluation, as well as a core package. Several methods that were relatively advanced for the day, such as bagging, were in place, as well as old chestnuts like instance-based learning and Naive Bayes. Attribute selection was added soon afterwards, in 1999.

Work began on the first edition of the *Data Mining* book in 1997, based on earlier notes for Witten's courses at the University of Waikato, and a proposal was submitted to Morgan Kaufmann late that year. It finally appeared in 1999 (though for reasons that are not clear to us the official publication date is 2000). WEKA was seen as an important adjunct to the book, and the original title, *Practical machine learning*, was changed to *Data Mining: Practical machine learning tools and techniques with Java* to reflect this. The term "data mining" was prepended primarily for marketing reasons. The WEKA software described in that edition was command-line oriented and the book makes no mention of a graphical user interface, for which design began in 1999. By the time the second edition appeared in 2005 the interactive versions of WEKA—the Explorer, Experimenter, and Knowledge Flow interface—were mature and well-tested pieces of software.

4. How Did WEKA Become Widely Adopted?

The size of the mailing list, the volume of downloads, and the number of academic papers citing WEKA-based results show that the software is widely deployed. It is used in the machine learning and data mining community as an educational tool for teaching both applications and technical internals of machine learning algorithms, and as a research tool for developing and empirically comparing new techniques. It is applied increasingly widely in other academic fields, and in commercial settings. We are often asked what is the secret of WEKA's success, and here we speculate on reasons for the software's broad uptake. An obvious one is that it is free and open-source software. However, there are several other factors, many of which are exposed by the above brief historical review.

4.1 Portability

Pre-Java versions of WEKA were limited to UNIX operating systems and distributions were made available for Linux, Solaris, and SGI. The present popularity of the software owes much to the existence of Java Virtual Machines for all important platforms, along with the fact that all code

necessary to compile and run WEKA is included in the distribution. The lack of dependency on externally-maintained libraries makes maintenance of the code base much easier.

The decision to rewrite WEKA from scratch in Java may seem obvious in hindsight, but was a large step into the unknown at the time. For one thing, portability seemed less important in the days when everyone we knew was using Unix! More importantly, in 1997 no-one could forsee that Java would turn out to be such a suitable platform. Just-in-time compilation was unknown (Sun's Java 1.2, which included a just-in-time compiler, was introduced in December 1998). Fully-interpreted execution suffered a substantial performance hit compared to C++. And deficiencies in early Java technology still affect the WEKA code today: the original design of the basic data structures was clean, object-oriented, and elegant, but the performance penalty incurred by extensive use of objects led to a less generic array-based representation, where all data is stored in arrays of doubles. Many of the design characteristics of WEKA that seem inelegant are due to pragmatic decisions based on the Java technology available at the time.

Later versions of the Java Virtual Machine virtually eliminated the performance gap from C++ through just-in-time compilation and adaptive optimization. Although there still persists a perception that execution of Java code is too slow for scientific computing, this is not our experience and does not appear to be shared by the WEKA community.

4.2 Graphical User Interface

Early releases of the WEKA 3 software were command-line driven and did not include graphical user interfaces. Although many experienced users still shun them, the graphical interfaces undoubtedly contributed to the popularity of WEKA. The introduction of the Explorer in particular has made the software more accessible to users who want to employ machine learning in practice. It has allowed many universities (including our own) to offer courses in applied machine learning and data mining, and has certainly contributed to WEKA's popularity for commercial and industrial use. Again, while obvious in hindsight, the development of graphical user interfaces was a significant risk, because valuable programming effort had to be diverted from the main job of implementing learning algorithms into relatively superficial areas of presentation and interaction.

The WEKA 3 graphical user interface development benefited from the fact that the core system was already up and running, and relatively mature—as evidenced by the first edition of *Data Mining*—before any work began on interactive interfaces. The early WEKA project probably suffered from attempting to develop interactive interfaces (in TCL/TK) at the same time as the basic algorithms and data structures were being commissioned, a mistake that was avoided in the later system.

4.3 The *Data Mining* Book

The first stable release of the WEKA 3 software coincided with the publication of the first edition of *Data Mining* by Witten and Frank, which contained a chapter describing it, both interactively via the command-line and programmatically via the API and extension of superclasses. There has been a symbiotic relationship between the software and the book: users of the software are likely to consult the book and readers of the book are likely to try out the software. The combination of a book explaining the core algorithms in a corresponding piece of free software is particularly suitable for education. It seems likely that the feedback loop between the readership of the book and the users of the software has bolstered the size of both populations. The early existence of a

companion book, unusual for open source software, is particularly valuable for machine learning because the techniques involved are quite simple and easy to explain, but by no means obvious.

4.4 Extensibility

WEKA can be used both for educational purposes and as a research platform. New algorithms can easily be incorporated and compared to existing ones on a collection of data sets. As noted earlier, only two methods from the `Classifier` superclass need be implemented to come up with a classifier that takes advantage of all the infrastructure in WEKA, including I/O, evaluation, and so-called "meta" algorithms.

4.5 Documentation

In recent years there has been a dramatic growth in the volume of WEKA-related online documentation—notably as part of the WEKA Wiki. This, in conjunction with the information in the mailing list archives, provides a wealth of information for all users. The existence of a steadily increasing, knowledgeable and enthusiastic user community, and the combined knowledge they share, has played a significant role in the success of the software.

4.6 Comprehensiveness

Perhaps the foremost reason for the adoption of WEKA 3 by the research community has been the inclusion of faithful reimplementations of classic benchmark methods, in particular the C4.5 decision tree learner (Quinlan, 1992), but also other algorithms such as the RIPPER rule learner (Cohen, 1995). The original implementations of these algorithms were already very successful software projects in themselves. Bringing them together in a common framework has been a strong drawing card.

4.7 Support

"Given sufficient funding, anyone could have done that!" is a refrain often heard from sceptics. We were lucky to receive an initial research grant for *applied* machine learning research from a New Zealand funding agency that approved of our aspirations to investigate the application of this technology in agricultural domains. Yet it is hard to reconcile the practical need to win academic credit for research publications with the production of usable software, particularly when there is a constantly growing pressure to commercialize. We continued to apply for, and receive, follow-on funding from the same source, but—particularly as time went on—this compelled us to channel much of our research in the direction of target applications rather than basic research in machine learning.

5. Maintaining the Project

A software project can only become and remain successful if it is consistently maintained to a high standard. It has been our experience that this requires a group of people who are continually involved in the management and development of the software for an extended period of time, spanning several years. The core development team of WEKA has always been small and close-knit: having a small team helps maintain code quality and overall coherence.

Development of the software would not have been possible without financial support from the New Zealand government in the form of successive research grants by the Foundation for Research, Science and Technology, which have been awarded over a significant period of time—from 1993 to 2007. Over the years, work on the project has been done by a couple of academic staff, who were involved in the longer term and fitted it in with their teaching and research duties, and a succession of one (or one and a half) full-time equivalent research programmers. A fair amount of work was undertaken by students on casual contracts or as part of their studies. The community also contributed many algorithm implementations that are included in the WEKA distribution, along with some patches for the basic framework.

The project has always had a policy of maintaining close control of what became part of the software. Only a handful of developers have ever had write access to the source code repository. The drawback of this policy is reduced functionality; the advantages are improved code quality, ease of maintenance, and greater coherence for both developer and end user. When new algorithm implementations were considered for inclusion, we generally insisted on a backing publication describing the new method. In a few cases we have rejected submitted code, despite publication, when our experiments revealed that the method did not appear to improve on what was already present in WEKA.

The research contract that sponsored WEKA development required some measure of commercialization, and a few commercial licenses to parts of the WEKA code base owned by the University of Waikato have been sold. It eventually became clear that the succession of research contracts had a finite life span, and support by a commercial organization was necessary to keep WEKA healthy. Since 2007, Pentaho Corporation, a company that provides open-source business intelligence software and support, has contributed substantially to the maintenance of WEKA by hiring one of the chief developers and providing online help. As part of the requirement to commercialize the software, it has been necessary to maintain a branch in the source code repository that only contains code owned by the University of Waikato, an onerous but necessary facet of project maintenance.

6. Concluding Remarks

Obviously, in almost two decades of project development, many mistakes were made—but most were quickly corrected. One, mentioned above, regards the premature design of interactive interfaces, where WEKA 3 benefited from a strategic error made in the early WEKA project. Below are two instances of how adoption might have been strengthened had the project been managed differently.

One of the most challenging aspects of managing open source software development is to decide what to include in the software. Although careful control of contributions has substantial benefits, it limits community involvement (Bacon, 2009). A package-based architecture (such as that adopted by the R Development Team, 2009) provides a better platform for more widespread ownership and development. Under such a scheme, packages maintained by their developers can be loaded into the system on demand, opening it up to greater diversity and flexibility. A recent development in WEKA is the inclusion of package management, so that packages can easily be added to a given installation.[4] The project would probably have benefited by moving in this direction earlier.

We have learned that mailing lists for open source software are easier to maintain if the users are researchers rather than teachers. WEKA's widespread role in education has led to a repetitive

4. Currently in the development version only.

and distracting deluge of basic questions. Requests from students all over the world for assistance with their assignments and projects present a significant (and growing) problem; moreover, students often depart from proper mailing-list etiquette. It would have been better to identify the clientele for WEKA as a teaching tool, and offer a one-stop-shop for software, documentation and help that is distinct from the support infrastructure used by researchers.

All in all, WEKA has been a resounding success which we believe has significantly advanced the application of machine learning techniques in today's world. One of the most satisfying aspects of participating in the project is that the software has been incorporated into, and spawned, many other open-source projects.

Acknowledgments

We gratefully acknowledge the input of all contributors to the WEKA project, and want in particular to mention significant early contributions to the WEKA 3 codebase by Len Trigg and many later contributions by Richard Kirkby, Ashraf Kibriya and Xin Xu.

References

J. Bacon. *The Art of Community*. O'Reilly Media, 2009.

W. W. Cohen. Fast effective rule induction. In *Proc 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proc 15th International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

R. Kohavi, D. Sommerfield, and J. Dougherty. Data mining using MLC++ a machine learning library in C++. *International Journal on Artificial Intelligence Tools*, 6(4):537–566, 1997.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.