Workshop on Tools and Technologies in Statistics, Machine Learning and Information Retrieval for Educational Data Mining
(SMLIR@EDM2015)

Within Intelligent Data Analysis (IDA) research area there may be found several subareas such as Machine Learning/Statistics, Information Retrieval, Data Mining and lately Big Data and Cloud Computing. The exact boundaries between these research areas are not very clear and may mislead research efforts, each one having its own particularities in terms of input type and size, data processing methodologies and obtained output. As Data Mining makes intensive use of all these research subareas, it is mandatory to be aware of their subtle differences and, therefore, design and implement IDA systems that make research efforts sound and effective. The goal of this workshop is to gather research efforts that fall into any of the categories of subareas and have results into the application area of Education. The workshop is looking for contributions that provide experimental results in the area of EDM/Information Retrieval and are focused on data processing fundamentals from Machine Learning/Statistics perspective. From the practical point of view, the focus should be on presenting the details regarding what and how tools and technologies are used in order to obtain relevant data analysis engines.

The integration of tools and technologies for building IDA processes is a key issue for developing applications that improve the effectiveness of the e-Learning platforms. The EDM community will benefit from the discussions related to the advantages and drawbacks of various options in a practical context with experimental results, by improving the efficiency of building high quality software systems supporting the research efforts.

The first step of developing an IDA application should focus on choosing the right tool or technology that fits the task requirements (e.g., input size, algorithm type, running time efficiency, scalability, etc.). The diversity of the available options is an indication of the necessity for a detailed analysis. From this point of view, the EDM community needs to be aware of success and failure attempts of many practical research efforts in order to provide the possibility of a proper future design choice.

Existing tools and technologies implement in different ways recent advances on techniques from statistical/machine learning, information retrieval and data mining domains in terms of programming language (e.g., Java, C/C++, C#, R, etc.), toolkits (e.g., Weka, Apache Mahout, MLTK, Maple, Matlab, etc.) and implementation details that may have a great impact on running times, scalability, effectiveness or efficiency.

This workshop brings together researchers from academia and industry practitioners with special interest in statistics/machine learning, information retrieval, data mining to (1) discuss current state of the art tools and technologies, (2) identify patterns for proper usage of various options for different tasks, and (3) lay out a vision regarding the modality in which tools and technologies will influence future applications. The organizers hope to obtain common background knowledge for integrating various tools and technologies in future EDM applications.

I gratefully acknowledge the following members of the workshop program committee:
Ivan Lukovic, University of Novi Sad, Serbia
Lucian Vintan, "Lucian Blaga" University of Sibiu, Romania
Stefan Trăuşan-Matu, University “Politehnica” of Bucharest, Romania
Andrea Zielinski, Fraunhofer IOSB, Karlsruhe, Germany
Mihai Lupu, Vienna University of Technology, Austria

Integrating an Advanced Classifier in WEKA

Paul Ştefan Popescu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
sppopescu@gmail.com

Mihai Mocanu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
mocanu@software.ucv.ro

Marian Cristian Mihăescu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
mihaescu@software.ucv.ro

ABSTRACT

In these days WEKA has become one of the most important data mining and machine learning tools. Despite the fact that it incorporates many algorithms, on the classification area there are still some unimplemented features. In this paper we cover some of the missing features that may be useful to researchers and developers when working with decision tree classifiers. The rest of the paper presents the design of a package compatible with the WEKA Package Manager, which is now under development. The functionalities provided by the tool include instance loading, successor/predecessor computation and an alternative visualization feature of an enhanced decision tree, using the J48 algorithm. The paper presents how a new data mining/machine learning classification algorithm can be adapted to be used integrated in the workbench of WEKA.

Keywords

Classifier, J48, WEKA, Machine learning, Data Mining

1. INTRODUCTION

Nowadays huge amounts of data can be gathered from many research areas or industry applications. There is a certain need for data mining or knowledge extraction [6] from data. From this large amount of data, the data analysts gather many variables/features and many machine learning techniques are needed to face this situation. There are many application domains such as medical, economics (i.e., marketing, sales, etc.), engineering or in our case educational research area [16] in which machine learning techniques can be applied. Educational data mining is a growing domain [4] in which a lot of work has been done.

Because the application domains are growing continuously, the tools that support the machine learning processes must live up to market standards providing good performances and intuitive visualization techniques. In these days there are many tools that deal with a wide variety of problems. In order to be more explicit, we have tools like RapidMiner [12], KEEL [2], WEKA, Knime [3] or Mahout [13]. RapidMiner is a graphical drag and drop analytics platform, formerly known as YALE, which provides an integrated environment for data mining, machine learning, business and predictive analytics. Keel is an application package of machine learning software tools, specialized on the evaluation of evolutionary algorithms. KNIME, the Konstanz Information Miner, is a modular data exploration platform, provided as an Eclipse plug-in, which offers a graphical workbench and various components for data mining and machine learning. Mahout is a highly scalable machine learning library based on the Hadoop framework [18], an implementation of the MapReduce programming model, which supports distributed processing of large data sets across clusters of computers.

For our approach we choose WEKA because it has become one of the most popular machine learning and data mining workbenches and its success is due to its constant improvement and development. Moreover, WEKA is a very popular tool used in many research domains, widely adopted by the educational data mining communities.

WEKA is developed in Java and encapsulates a collection of algorithms that tackle many data mining or machine learning tasks like preprocessing, regression, clustering, association rules, classification and also visualization techniques. In some cases, these algorithms are referring only the basic implementation.

One aspect that needs to be taken into consideration is that WEKA has a package manager which simplifies the developers contribution process. There are two kind of packages that can be installed in WEKA and used via the application interface: official and unofficial packages. This is a very important feature because if there is an algorithm that fits your problem description and there is a package for it you can just add it to the application and use it further. Moreover, you don't need to be a programmer to do that, you don't need to write code, just install the package and then use the algorithm like it had been there forever.

According to the real life experiences, many of the included algorithms can hardly be used because of their lack of flexibility. For example, in standard decision trees from WEKA we can perform a classification process but we cannot access a particular instance from the tree. Suppose that we have a training data file and we create the tree model. When we try to see where is the place of the instance "X" in the tree we can't do that in the application interface, neither when you add the WEKA library in your code. This is a big drawback because retrieving the leaf to which the instance belongs to provides more information than retrieving its class. Usually, when performing a classification task, the data analyst divides test instances into classes that have little meaning from application domain of perspective.

In a real life scenario in a training dataset we may have a large number of features describing the instances. A data analyst should be able to parse a decision tree, see the rule that derived to a specific decision and then draw very accurate conclusions In this paper we will address classification and visualization issues by adding new functionalities and improving the decision tree visualization.

Several classification algorithms have been previously contributed to WEKA but non of them is able to output a data model that is loaded with instances. Based on the previous statement it is clear that there aren't WEKA visualization techniques that are able to present the data in the model in a efficient way and also, there are no available parsing methods ready to implement such functionalities. Traversal of leaves is another task that is missing and it is important because instances from neighbour leaves have a high degree of similarity and share many attributes with similar values.

One aspect that differs at WEKA from other similar software regards its architecture that allows developers to contribute in a productive way. All the work that needs to be done refers to creating a specific folders layout, completing a "description.props" file, adding the ".jar" file to the archive and the build script.

2. RELATED WORK

WEKA is a open source machine learning library that allows developers and researchers to contribute very easily. There are more than twenty years since WEKA had it's first release [9] and there were constant contributions added on it. Not only machine learning algorithms were implemented, for example, in 2005 a text data mining module was developed [20]. An overview of the actual software was made in [8].

Several classifiers were developed and contributed as packages to WEKA. In 2007 a classifier that was build based on a set of sub-samples was developed [14] and compared to C4.5 [15] which have it's implementation called J48 [11] in WEKA. Other classifiers refers the "Alternating Decision Trees Learning Algorithms" [7] which is a generalization of the decision trees, voted decision trees and voted decision stumps. This kind of classifiers are relatively easy to interpret and the rules are usually smaller in size. Classical decision trees, such as c4.5 were expanding nodes in a depth-first order; an improvement came from "Best-first decision trees" [17]which expands nodes in a best-first order. A package with these trees was contributed to WEKA.

Some other contributions refers libraries of algorithms that can be accessed via WEKA. One of them is JCLEC [5] an evolutionary computation framework which has been successfully employed for developing several evolutionary algorithms. Other environment for machine learning and data mining knowledge discovery that was contributed to WEKA is R [10]. This contribution was developed in order to include different sets of tools from both environments available in a single unified system.

Also as related work we must take into consideration some of the last algorithms development. In the last year it is presented a new fast decision tree algorithm [19]. Based on their experiments, the classifier outperforms C5.0 which is the commercial implementation of C4.5.

3. SYSTEM DESIGN

The package is designed to be used both by developers, in their Java applications, and researchers, using the WEKA Explorer. At the moment of writing this paper the package with the Advanced Classifier is still under development, offering more functionalities as a tool for developers than in the explorer view of WEKA.

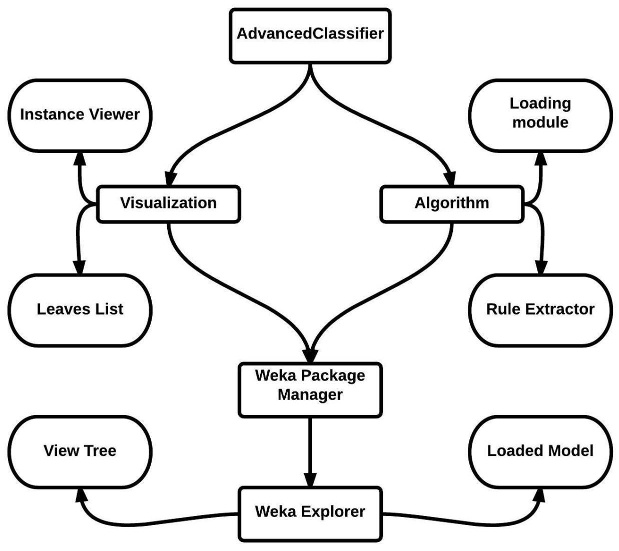


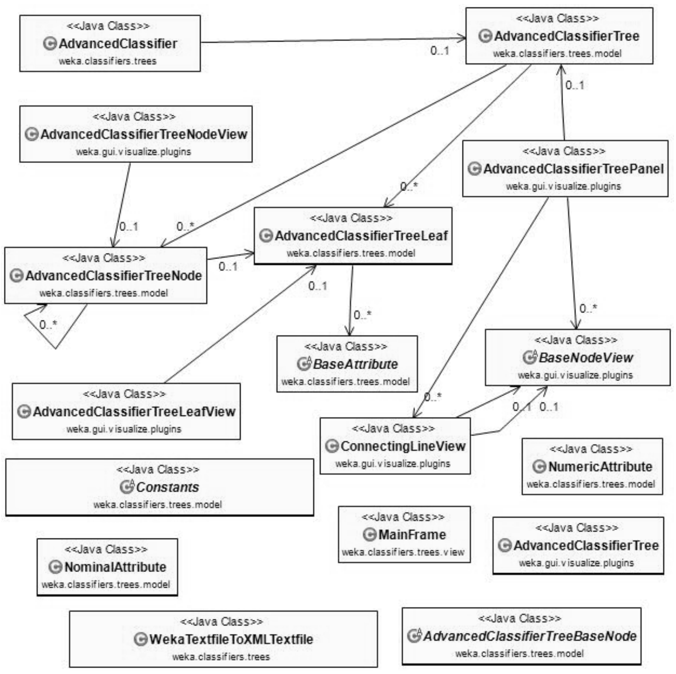
Figure 1: Package Integration in WEKA

In Fig. 1 we present the main design of the algorithm and how it can be used in WEKA. On the top of the figure we have the classifier which can be divided in two main modules: the algorithm and the visualization. As we can see on the next level, both of the modules can be divided further. All the functionalities are then installed in WEKA via the package manager and then, in the explorer, we can perform data analysis tasks using a model loaded with data and it's associated visualization techniques.

3.1 General Architecture

The packages is a zip archive, structured with respect to the WEKA guidelines. That is, it unpacks to the current directory and it contains: the source files, a folder with the required libraries, a build script, a properties file required by WEKA for installing and managing the package, and the actual ".jar" file. A detailed structure of the package is presented below.

```
<current directory>
  +-AdvancedClassifier.jar
  +-Description.props
  +-build_package.xml
  +-src
  |  +-main
  |    +-java
  |      +-resources
  |      |  +-background_node.png
  |      |  +-background_leaf.png
  |      |  +-background_leaf_pressed.png
  |      |  +-font_node.ttf
  |      |  +-font_decision.ttf
  |      +-weka
  |          +-classifiers
  |          |  +-trees
  |          |     +-model
  |          |     |  +-AdvancedClassifierTree.java
  |          |     |  +-AdvancedClassifierTreeBaseNode.java
  |          |     |  +-AdvancedClassifierTreeNode.java
  |          |     |  +-AdvancedClassifierTreeLeaf.java
  |          |     |  +-BaseAttributeValidator.java
  |          |     |  +-NominalAttributeValidator.java
  |          |     |  +-NumericAttributeValidator.java
  |          |     |  +-Constants.java
  |          |     +-AdvancedClassifier.java
  |          |     +-WekaTextfileToXMLTextfile.java
  |          +-gui
  |             +-visualize
  |                +-plugins
  |                   +-AdvancedClassifierTree.java
  |                   +-AdvancedClassifierTreePanel.java
  |                   +-BaseNodeView.java
  |                   +-AdvancedClassifierTreeNodeView.java
  |                   +-AdvancedClassifierTreeLeafView.java
  |                   +-ConnectingLineView.java
  +-lib
    +-weka.jar
    +-simple-xml.jar
    +-rt.jar
```



<<Java Class>> AdvancedClassifier weka.classifiers.trees
<<Java Class>> AdvancedClassifierTree weka.classifiers.trees.model
<<Java Class>> AdvancedClassifierTreeNodeView weka.gui.visualize.plugins
<<Java Class>> AdvancedClassifierTreePanel weka.gui.visualize.plugins
<<Java Class>> AdvancedClassifierTreeLeaf weka.classifiers.trees.model
<<Java Class>> AdvancedClassifierTreeNode weka.classifiers.trees.model
<<Java Class>> BaseAttribute weka.classifiers.trees.model
<<Java Class>> BaseNodeView weka.gui.visualize.plugins
<<Java Class>> AdvancedClassifierTreeLeafView weka.gui.visualize.plugins
<<Java Class>> ConnectingLineView weka.gui.visualize.plugins
<<Java Class>> NumericAttribute weka.classifiers.trees.model
<<Java Class>> Constants weka.classifiers.trees.model
<<Java Class>> MainFrame weka.classifiers.trees.view
<<Java Class>> AdvancedClassifierTree weka.gui.visualize.plugins
<<Java Class>> NominalAttribute weka.classifiers.trees.model
<<Java Class>> WekaTextfileToXMLTextfile weka.classifiers.trees
<<Java Class>> AdvancedClassifierTreeBaseNode weka.classifiers.trees.model
0..1
0..*

Figure 2: Class Diagram

In Figure 2 is presented the system's class diagram. This diagram includes all the java packages from the project and their relations. As we can see in the above mentioned figure, we have two type of classes: independent classes and composed. Independent classes are gathered from the model part of the Model-View-Controller architecture or just classes that perform one time tasks like "WekaTextFileToXMLTextFile" which is able to generate an XML based on the text file outputted by WEKA. On the other side, the composed classes are dependent on each other and these relations are shared across packages. One important class that is worth to be mentioned is "AdvancedClassifierTreeLeaf.java" in which we store the leaves of our tree along with rules that define the leaf. Discussions about implementation of the packages are more related to the software engineering research area and beyond of the scope of this paper.

3.1.1 Design and Implementation of the Algorithm

The algorithm needs to generate custom rules (dependent on the training dataset) for every leaf of the decision tree. These rules are computed by tracing the path from the root of the tree to the specified leaf. Each decision that leads to a leaf is therefore translated into a rule that encapsulates the name of the attribute and the value on which the decision was made. For each type of attribute defined by WEKA, we need to have a corresponding rule that matches that type. For this purpose an abstract class has been created to act as a base class for any of the custom rules. The name of this class is "BaseAttributeValidator" and exposes the required methods that a superclass needs to implement: a "clone" method required by the workflow of the system and methods that validate if an instance or set of instances have the required values of the attribute targeted by the rule. At the moment, the only implemented rules are the ones that handle "NOMINAL" and "NUMERIC" attribute types.

The rule that validates each nominal attribute is called "NominalAttributeValidator" and receives as parameters the name of the targeted attribute and a string variable representing the accepted value of the attribute. The rule that handles the numeric attributes is called "NumericAttributeValidator" and also receives the name of the attribute and either a particular value or the boundaries of an interval.

In the following paragraphs, we present a brief overview of the algorithm for which we adopt a straightforward approach.

Firstly, the algorithm retrieves instances from the ".arff" file using the methods provided by WEKA. The next step is applying the desired classification process. Currently the only supported classifier is J48, but employing other decision tree classifiers is foreseen as future work. Using the text representation of the outputted model and a predefined set of rules and tags, an XML is then generated. This is an important step during the workflow because the structured XML format allows us to obtain the base model for our decision tree. The deserialization is done using a third-party Java library("Simple XML" [1]).

The model obtained this way contains a list of nodes and leaves with the following significance: each node corresponds to a decision in the tree; the data stored in each object (node) refers the information about the name of the actual attribute, operator and value on which the decision was made; and the results to which making the decision leads (a list of other nodes or an output leaf). Using this model and the set of attributes provided by WEKA, the set of rules is computed. This step is performed by parsing the model from the first node (i.e., the root) to the last available leaf and gradually composing the set of rules that defines each leaf. The setup of the algorithm is finally completed with the loading of the training dataset into the model.

The classifier and processed data can now be easily handled and different operations can be applied. The method currently implemented include basic per leaf manipulation of instances, i.e. loading new instances into the model and retrieving the part of the dataset contained in each leaf, as well as predecessor and successor computation.

3.1.2 *Visualization Plugin*

For the visualization feature, a custom panel has been designed to hold the components that build up the decision tree and expose the data available in the leaves. The contructor of the panel requires the decision tree model as a parameter, and takes care of adding the corresponding views to the interface. In order to include this functionality in WEKA, a specialized class that implements WEKA's TreeVisualizePlugin interface has been created. After adding the package through the Package Manager and selecting this visualization option, a new JFrame that holds the custom panel is displayed.

```
@RELATION StudentClass

@ATTRIBUTE userid numeric
@ATTRIBUTE nrHours numeric
@ATTRIBUTE avgMark numeric
@ATTRIBUTE present {NO,YES}
@ATTRIBUTE class {low,high}

@DATA
4,17,6.025,NO,low
5,11,7.0,YES,low
7,30,7.375,NO,low
8,10,7.714286,NO,low
9,43,5.1666665,YES,high
10,31,4.5,YES,high
```

Figure 3: Sample from the Dataset

In Figure 3 we present a dataset sample. In order to validate the classifier and it's extra functionalities several tests have been made but for this case study we used three attributes and 788 instances. The feature called "userid" doesn't provide any information gain but can be easily used for instances localization in leaves. The attributes significance is beyond the scope of this paper.

In Figure 4 is presented a screen-shot of the tree generated based on the dataset from figure 3. Each node contains

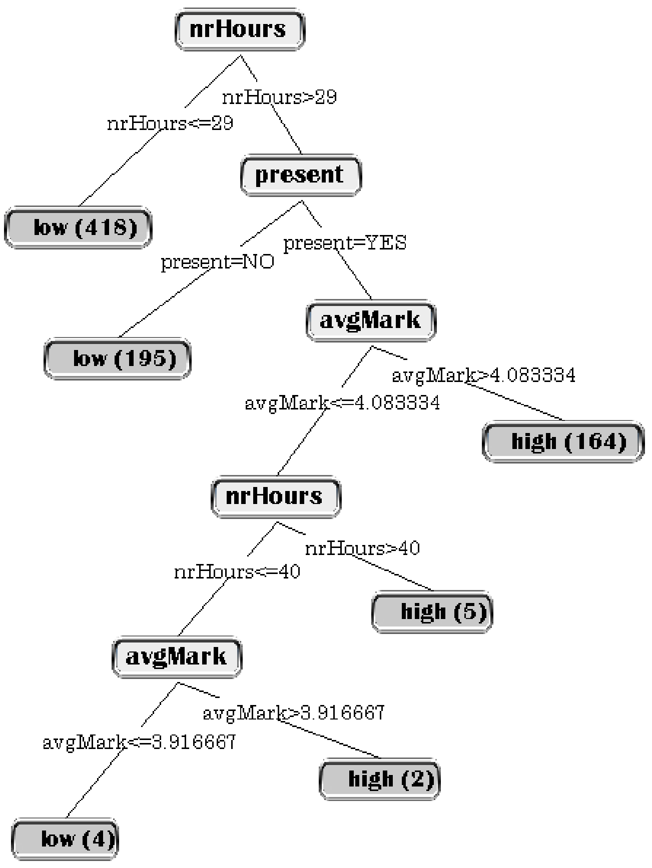


Figure 4: Tree Sample

the name of the attribute, and each decision is printed on top of the connecting line. Surely, each leaf can be clicked, and the set of enclosed instances is displayed. As previously noted, there is still some work to be made to finalize the development of the package, and the visualization tool needs to be included as well. Efforts will have to be made toward providing the means to visualize and handle the successors/predecessors, outliers and other relevant information.

4. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the integration of a data analysis tool in WEKA. This tool is important because brings a new classifier to WEKA that aims to improve the classification procedures. Here, are also presented some implementing procedures and details.

A workflow is also described and all the mechanism that is used to bring new features for the users. One important thing that needs to be mentioned is that the data loading module opens new data analysis opportunities for researchers.

As future work we plan to implement Other types of attributes supported by WEKA like "DATE", "String" and "Relational".

5. REFERENCES

[1] Simple xml. http://simple.sourceforge.net.

[2] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. FernÃąndez, and F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[3] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. Knime - the konstanz information miner: Version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31, Nov. 2009.

[4] R. Campagni, D. Merlini, R. Sprugnoli, and M. C. Verri. Data mining models for student careers. *Expert Systems with Applications*, (0):–, 2015.

[5] A. Cano, J. M. Luna, J. L. Olmo, and S. Ventura. Jclec meets weka! In E. Corchado, M. Kurzynski, and M. Wozniak, editors, *HAIS (1)*, volume 6678 of *Lecture Notes in Computer Science*, pages 388–395. Springer, 2011.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, Nov. 1996.

[7] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 124–133, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[9] G. Holmes, A. Donkin, and I. H. Witten. Weka: a machine learning workbench. pages 357–361, August 1994.

[10] K. Hornik, C. Buchta, and A. Zeileis. Open-source machine learning: R meets weka. *Computational Statistics*, 24(2):225–232, 2009.

[11] W.-Y. Loh. *Classification and Regression Tree Methods.* John Wiley & Sons, Ltd, 2008.

[12] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 935–940, New York, NY, USA, 2006. ACM.

[13] S. Owen, R. Anil, T. Dunning, and E. Friedman. *Mahout in Action.* Manning Publications Co., Greenwich, CT, USA, 2011.

[14] J. M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, and J. I. Martín. Combining multiple class distribution modified subsamples in a single tree. *Pattern Recognition Letters*, 28(4):414–422, 2007.

[15] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[16] C. Romero and S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33(1):135 – 146, 2007.

[17] H. Shi. Best-first decision tree learning. Technical report, University of Waikato, 2007.

[18] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[19] S.-G. P. V. Purdila. Fast decision tree algorithm. *Advances in Electrical and Computer Engineering*, 14(1):65–68, 2014.

[20] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, New York, NY, USA, 1999. ACM.

A Systematic Mapping Study on the Usage of Software Tools for Graphs within the EDM Community

Vladimir Ivančević*
University of Novi Sad, Faculty of Technical Sciences
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
dragoman@uns.ac.rs

Ivan Luković
University of Novi Sad, Faculty of Technical Sciences
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
ivan@uns.ac.rs

ABSTRACT

The field of educational data mining (EDM) has been slowly expanding to embrace various graph-based approaches to interpretation and analysis of educational data. However, there is a great wealth of software tools for graph creation, visualization, and analysis, both general-purpose and domain-specific, which may discourage EDM practitioners from finding a tool suitable for their graph-related problem. For this reason, we conducted a systematic mapping study on the usage of software tools for graphs in the EDM domain. By analysing papers from the proceedings of previous EDM conferences we tried to understand how and to what end graph tools were used, as well as whether researchers faced any particular challenges in those cases. In this paper, we compile studies that relied on graph tools and provide answers to the posed questions.

Keywords

Systematic Mapping Study, Graphs, Software Tools, Educational Data Mining.

1. INTRODUCTION

The field of educational data mining (EDM) has significantly expanded over the past two decades. It has attracted numerous researchers with various backgrounds around the common goal of understanding educational data through intelligent analysis and using the extracted knowledge to improve and facilitate learning, as well as educational process. In 2010, Romero and Ventura published a comprehensive overview of the field with 306 references [26]. In this review, the authors identified 11 categories of educational tasks, two of which dealt with graph structures (for brevity these will be referred to as graphs): social network analysis (SNA) and developing concept maps. However, the authors noted that these two categories featured a lower number of papers (15 or less references collected). Somewhat different categories of work were presented in another review of EDM [2] but they did not include any explicit references to graphs.

However, since that time, the interest in approaches and technologies utilizing graphs has increased within EDM. In addition to the results of a literature search on the topic, this could be also evidenced by the appearance of the Workshop on Graph-Based Educational Data Mining (G-EDM)[1] in 2014. As a result, software tools that help researchers or any other user group to utilize graphs or graph-based structures (for brevity these will be referred to as graph tools) are becoming a valuable resource for both the G-EDM and the broader EDM community. As graphs are only slowly gaining wider recognition in EDM, there could still be a lot of questions about which graph tools exist or what educational tasks might be supported by these tools.

In an attempt to help EDM researchers discover more useful information about potentially suitable graph tools, we reviewed the papers presented at the past EDM conferences, selected those that mentioned any usage of graph tools, and extracted from them information about which graph tools the authors employed, what features of these tools were used, to what end the research in question was conducted, and if there were any particular challenges while using these tools.

The present study may be classified as a secondary study since we base our approach on collecting other research works and assembling relevant information from them. Secondary studies might be more typical of medical and social sciences but there are proposed methodologies concerning secondary studies in software engineering as well [13]. Two kinds of secondary studies might be particularly important in this context: systematic review studies and systematic mapping studies [20]. In both cases, there is a clear methodology that is set to reduce bias when selecting other research works, which gives these secondary studies the quality of being systematic. Some of the differences pointed out by Petersen et al. [20] are that systematic reviews tend to focus on the quality of reviewed studies with the aim of identifying best practices, while systematic maps focus more on classification and thematic analysis but with less detailed evaluation of collected studies. Moreover, the same authors consider that the two study types form a continuum, which might complicate some attempts at categorization.

We categorize the present study as a systematic mapping study. This classification is justified by the fact that:

1. we employed a concrete methodology,
2. we did not evaluate the quality of collected papers or the presented results, but
3. we focused on identifying the employed graph tools and the manner in which these tools were used, with the aim

*Corresponding Author

[1] http://ceur-ws.org/Vol -1183/

of providing an overview of the current practice of using graph tools within the EDM community

However, we did not restrict our investigation to analysing exclusively titles, abstracts, or keywords, but went through the complete texts to find the necessary information. This aspect might better suit systematic reviews, but it does not change the principal goal or character of our study.

The exact details of the employed methodology, including the research questions, sources of studies, and study selection criteria, are given in Section 2. Section 3 contains the answers to the research question, most importantly the list of identified graph tools and the trends in their usage in EDM. Section 4 covers the potential limitations of the present study.

2. METHODOLOGY

We mainly followed the guidelines given in [20] but also relied on the example of a mapping study presented in [21]. Given the specificity of our study and the posed research questions, there were some necessary deviations from the standard suggested procedure. The overall process of selecting papers and extracting information, together with the resolution methods for non-standard cases, is presented and discussed in the following subsections.

2.1 Overview

The first step was defining research questions to be answered by the present study. The choice of research questions influenced the subsequent steps: conducting the search for papers, screening the papers, devising the classification scheme, extracting data, and creating a map.

2.2 Research Questions

We defined four principal research questions (RQ1-RQ4) concerning the use of graphs and graph tools in studies by EDM researchers:

- RQ1: Which graph tools were directly employed by researchers in their studies?
- RQ2: Which features of the employed graph tools were used by researchers?
- RQ3: What was the overall purpose of the research that involved or relied on graph tools?
- RQ4: What features did researchers consider to be missing or inadequate in the employed graph tools?

2.3 Search for Papers

We searched through all the papers that were published in the proceedings of the EDM conference series till this date, i.e., papers from the first EDM conference in 2008 to the latest, seventh, EDM conference in 2014. The latest EDM conference was special because it also included four workshops (G-EDM being one of them) for the first time. The papers from these workshops were also considered in our search. This amounted to eight relevant conference proceedings that represented the complete source of research works for our study:

1. Proceedings of the 1st International Conference on Educational Data Mining 2008 (Montreal, Canada)
2. Proceedings of the 2nd International Conference on Educational Data Mining 2009 (Cordoba, Spain)
3. Proceedings of the 3rd International Conference on Educational Data Mining 2010 (Pittsburgh, Pennsylvania, USA)
4. Proceedings of the 4th International Conference on Educational Data Mining 2011 (Eindhoven, Netherlands)
5. Proceedings of the 5th International Conference on Educational Data Mining 2012 (Chania, Greece)
6. Proceedings of the 6th International Conference on Educational Data Mining 2013 (Memphis, Tennessee, USA)
7. Proceedings of the 7th International Conference on Educational Data Mining 2014 (London, UK)
8. Extended Proceedings of the 7th International Conference on Educational Data Mining 2014 (London, UK), which included only the workshop papers

All the proceedings are freely offered as PDF files by the International Society of Educational Data Mining[2] and may be accessed through a dedicated web page.[3]

The papers from these proceeding represented our Level 0 (L0) papers, i.e., the starting set of 494 papers. This set included different categories of papers: full (regular) papers, short papers, different subcategories of posters, as well as works from the young researcher track (YRT) or demos/interactive events. The starting set did not include abstracts of invited talks (keynotes), prefaces of proceedings, or workshop summaries.

These papers were then searched and evaluated against our keyword criterion (KC), which led to a set of Level 1 (L1) papers. Our keyword string is of the form KC1 AND KC2 where KC1 and KC2 are defined in the following manner:

- KC1: graph OR subgraph OR clique
- KC2: tool OR application OR software OR framework OR suite OR package OR toolkit OR environment OR editor

The first part of the criterion (KC1) was defined to restrict the choice to papers that dealt with graphs, while the second part (KC2) served to narrow down the initial set of papers to those mentioning some kind of a tool or program in general.

When evaluating KC on each L0 paper, we did a case-insensitive search for whole words only, whether in their singular form (as written in KC1 and KC2) or their plural form (except for the case of "software"). This search also included hyphenated forms that featured one of the keywords from KC, e.g., "sub-graph" was considered to match the "graph" keyword.

As each proceedings file is a PDF document, we implemented a search in the Java programming language using the Apache PDFBox[4] library for PDF manipulation in Java. However, when extracting content from some papers, i.e., page ranges of a proceedings file, we could not retrieve text in English that could be easily searched. This was most probably caused by the fact that

[2] http ://www.educationaldatamining.org/

[3] http://www.educationaldatamining.org/proceedings

[4] https://pdfbox.apache.org/

authors used different tools to produce camera ready versions in PDF, which were later integrated into a single PDF file.

In these instances, usually one of the two main problems occurred: no valid text could be extracted or valid text was extracted but without spacing. In the case of invalid text, we had to perform optical character recognition (OCR) on the problematic page ranges. We used the OCR feature of PDF-XChange Viewer,[5] which was sufficient as confirmed by our manual inspection of the problematic page ranges (six problematic papers in total). In the case of missing spacing, we had to fine-tune the extraction process using the capabilities of the PDFBox library.

This PDF library proved adequate for our task because we had to search only through PDF files and could customize the text extraction process to solve the spacing problem. However, in the case of a more varied data source, a more advanced toolkit for content indexing and analysis would be needed.

2.4 Screening of Papers

EDM researchers used many of our keywords with several different meanings, e.g., a graph could denote a structure consisting of nodes and edges, which was the meaning that we looked for, or some form of a plot. In order to determine the final set of papers we performed a two-phase selection on L1 papers:

1. We examined the portions of L1 papers that contained some KC1 keyword and eliminated papers that did not significantly deal with graphs (as structures) – this led to a set of Level 2 (L2) papers.
2. We read each L2 paper and eliminated those that did not mention some use of graphs tools – this led to the final set of Level 3 (L3) papers.

In the first phase of selection, we examined the sentences that contain KC1 keywords. If this proved insufficient to determine the nature or scope of use of the mentioned graphs, we read the whole paragraph, and sometimes even the paragraph before and the paragraph after. In these cases, we also checked the referenced figures, tables, or titles of the cited papers. If there were still any doubts, we consulted the paper's title and abstract, as well as glanced over the figures looking for graph examples. If the authors did not use graphs in their presented study or just made a short comment about graphs giving an analogy or mentioning graphs in the context of related or future work, we did not select the paper for the next phase.

In the second phase of selection, we kept only those papers that mention explicit use of a graph tool by the authors. In the cases when the actual use of a mentioned graph tool was not clear, the paper was selected if some of its figures contain a screenshot featuring the tool or a graph visualized using that tool.

The term tool was considered rather broadly in the present study. We did not restrict the search only to well-rounded software applications, but also included libraries for various computer languages, and even computer languages or file formats that were used by researchers to manipulate graphs. By making this decision, we aimed to provide a greater breadth of information to researchers interested in applying graphs within their studies.

[5] http://www.tracker-software.com/product/pdf-xchange-viewer

2.5 Classification Scheme

The mode of tool usage was categorized in the following manner:

1. CREATION (C) – the tool was developed by the paper authors and introduced in the paper;
2. MODIFICATION (M) – the tool being modified, either through source code or by adding extensions/plugins; and.
3. UTILIZATION (U) – the tool being utilized without modification.

We also checked the distribution of the collected studies by the continent and the country corresponding to the authors' affiliation. In cases when there were authors from different countries, we indicated the country of the majority of authors, or, if there was no majority then the country corresponding to the affiliation of the first author.

2.6 Data Extraction and Map Creation

Relevant data from L3 papers was extracted into a table that for each paper included the following information: author list, title, proceedings where it was published, page range within the proceedings, answers to the research question and classifications according to the scheme presented in the previous subsection.

3. RESULTS AND DISCUSSION

An overview of the paper selection process is given in Table 1. In each step, the number of relevant papers is significantly reduced. As expected, the required effort in paper analysis was inversely proportional to the number of selected papers. In the L1 step, the usage of the keyword criterion relatively quickly eliminated many papers. However, in subsequent steps, the selected papers had to be read, either partially (in the L2 step) or fully (in the L3 step). The set of L3 papers represents a selection of EDM studies that were used to identify the usage patterns concerning graph tools. The list of the selected papers is publicly available.[6]

Table 1. The number of selected papers at each step

Step	Number of papers
L0 – papers from EDM proceedings	494
L1 – papers containing keywords	146
L2 – papers mentioning graphs	82
L3 – papers mentioning graph tools	27

Most studies (15) are from North America: USA (14) and Canada (1). Europe is represented by 8 studies from 6 countries: Czech Republic (2), Spain (2), Germany (1), Ireland (1), Russia (1), and UK (1). The remaining two continents represented are Asia (Japan only) and Australia, each providing 2 studies. This somewhat resembles the EDM community present at the EDM conferences and differs little from the structure of the EDM community as reported in 2009 [2].

3.1 Overview of Graph Tools

In Table 2, we list 28 graph tools mentioned in the 27 selected papers.

[6] http://www.acs.uns.ac.rs/en/user/31

Table 2. Overview of graph tools from the selected papers

No	Tool	Usage	Features	Purpose	Issues
1	<Untitled framework>	C[1]	argument database – retrieval and mining	retrieve, analyse, and reuse arguments	WIP
2	<Untitled tool>	C[25]	vis. and mine visit trails from WBESs	discover student trails in WBESs	/
3	AGG Engine	C[14], U[15]	augmented graph grammar engine with recursive graph matching	analyse student-produced argument diagrams	inefficiency in some cases
4	CASSI	C[19]	collect bullying data via web-form and use them to form a social graph	support classroom management	/
5	CLOVER framework	U[25]	generate graph vis.	(used in vis. in No. 2)	/
6	Cmate	U[16]	provide a list of concepts and linking words to build a concept map	tabletop concept mapping	/
7	D3.js	U[17]	program interactive graph vis.	facilitate graph interpretation in EDA	/
8	DOT	U[28]	describe graphs	(used in export in No. 14)	/
9	EDM Vis	C[9], M[10]	interactively vis. ITS log data	understand student problem solving in ITSs	WIP
10	eJUNG lib.	U[11]	layout graphs	(used in vis. in No. 14)	/
11	FuzzyMiner (ProM)	U[16]	generate fuzzy models (of student collaboration processes)	discover and analyse student strategies in tabletop collaboration	/
12	Gephi	U[7]	vis. graphs	identify similarities between LE course content (used together with No. 22)	/
13	graphML	U[30]	describe graphs (of student resolution proofs)	analyse student solutions of resolution proofs	/
14	InVis	C[11], M[12, 28]	interactively vis. and edit ITS log data	understand student interaction in ITSs	WIP
15	LeMo	C[18]	interactively vis. learning object networks	understand how students perform and succeed with resources in LMSs and LPs	/
16	Meerkat-ED toolbox	C[22]	vis., monitor, and evaluate participation of students in discussion forums	analyse student interaction and messages in discussion forums	/
17	meud	U[24]	create diagrams (concept lattices)	analyse choices of study programmes	/
18	Ora	U[6]	calculate SNA metrics	study SNA metrics to improve student performance classifiers	/
19	pajek	U[3],[32]	vis. networks and calculate network measures	use student social data to predict drop-out and failure; understand growth of communities on SNSs	/
20	R	U[8]	use scripts to vis. ELE interaction data	explore ELE interaction data and improve ELEs	WIP
21	R – igraph package	U[5],[32]	create, refine, vis., and analyse networks	compare student problem solving-approaches in ITSs; understand growth of communities on SNSs	/
22	RapidMiner	M[7]	create an operator for graph generation	identify similarities between LE course content (used together with No. 12)	/
23	RSP	C[4]	discover issues in the ITS process	support teachers through AT adaptation	/
24	SEMILAR toolkit	C[27]	semantic similarity methods for text	assess student natural language input in ITSs	/
25	SketchMiner	C[29]	generate graphs for student symbolic drawings; compare and cluster drawings	assess student symbolic drawings in ITSs	/
26	STG	C[4]	interactively vis. student interaction in ITSs	understand student problem solving in ITSs	/
27	TRADEM	C[23]	perform analysis on content corpus and generate a concept map in ITSs	support development of instructional content in ITSs	/
28	Visone	U[31]	vis. and analyse SNs, clique analysis	analyse user relationships in WBATs	/

The rows (graph tools) are ordered alphabetically by the tool name (the "Tool" column), which represents the answer to RQ1. In general, we discovered a diverse list of infrequently used graph tools. The usage of the graph tools, which represents the answer to RQ2, is covered by the columns "Usage" and "Features". In "Usage", we listed the mode of usage (see Section 2.5) and the references to the papers mentioning the graph tool. In "Features", we listed tool functionalities and capabilities that were created or employed by the researchers. The most often used feature was to visualize (vis.) graphs. The purpose of the selected studies, which represents the answer to RQ3, is given in the "Purpose" column. Researchers often analysed data from various interrelated systems: intelligent tutoring systems (ITSs) and adaptive tutorials (ATs), learning environments (LEs) including exploratory learning environments (ELEs), learning management systems (LMSs), learning portals (LPs), social network services (SNSs), web-based authoring tools (WBATs), and web-based educational systems (WBESs). Some frequent tasks were analysis of social networks (SNs) and exploratory data analysis (EDA).

The issues that the researchers faced when using the tools, which represents the answer to RQ4, are listed in the "Issues" column. In the majority of the selected papers, the researchers did not discuss problems related to tool usage. The main exceptions are studies in which researcher presented their own tools and discussed missing or incomplete features that should be fully implemented in future – this was labelled as work in progress (WIP).

4. POTENTIAL LIMITATIONS

The findings might not be representative of the whole EDM community but only of the practitioners who presented their work at one of the EDM conferences. An important issue in the analysis was the lack of information about the used tools. There were various instances when researchers obviously used a graph tool, or at least it could be expected that they relied on such tools, but failed to report the information.

Moreover, we used a somewhat "relaxed" definition of a graph tool. This allowed for the inclusion of both general-purpose tools for graph manipulation and domain-specific tools that were developed for educational domain but also utilize a graph-based structure. The primary motive behind this choice was to provide a list of graph tools potentially usable in a wider range of studies, as well as a list of tools that illustrates how graphs were implemented or used in a more specific problem. The former tool category generally includes tools associated with the "U" usage (tools utilized without modification), while the latter tool category mostly covers tools associated with the "C" usage (new tools introduced by their authors).

On the other hand, we excluded graph-based tools that could be labelled as data mining tools or causal modelling tools. For instance, some popular predictive and/or explanatory models (decision trees, random forests, and Bayesian networks) are graph-based, while causal modelling usually assumes creation or discovery of causal graphs. As these tools are more often featured in EDM studies, we assumed that EDM researchers are more familiar with their usage, so the focus of the present study is on other less frequently used graph tools.

5. CONCLUSION

We hope that the collected information about the usage of graph tools within the EDM community may prove valuable for researchers considering the use of graphs to solve educational problems. For future work, we plan to include other publication series, even those that are not solely devoted to the EDM research. The results of such an attempt could demonstrate whether EDM practitioners from other regions of the world are more represented in the graph-based research than indicated by the results of the present study.

6. ACKNOWLEDGMENTS

The research presented in this paper was supported by the Ministry of Education, Science, and Technological Development of the Republic of Serbia under Grant III-44010: "Intelligent Systems for Software Product Development and Business Support based on Models".

7. REFERENCES

[1] Abbas, S. and Sawamura, H. 2008. Argument mining using highly structured argument repertoire. In *Proceedings of the First International Conference on EDM* (Montreal, Canada, June 20 - 21, 2008). EDM'08. 202-209.

[2] Baker, R.S.J.d. and Yacef, K. 2009. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining,* 1 (1), 3-17.

[3] Bayer, J., Bydzovska, H., Geryk, J., Obsivac, T. and Popelinsky, L. 2012. Predicting drop-out from social behaviour of students. In *Proceedings of the Fifth International Conference on EDM* (Chania, Greece, June 19 - 21, 2012). EDM'12. 103-109.

[4] Ben-Naim, D., Bain, M. and Marcus, N. 2009. A user-driven and data-driven approach for supporting teachers in reflection and adaptation of adaptive tutorials. In *Proceedings of the Second International Conference on EDM* (Cordoba, Spain, July 1 - 3, 2009). EDM'09. 21-30.

[5] Eagle, M. and Barnes, T. 2014. Exploring differences in problem solving with data-driven approach maps. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 76-83.

[6] Garcia-Saiz, D., Palazuelos, C. and Zorrilla, M. 2014. The predictive power of SNA metrics in education. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 419-420.

[7] Goslin, K. and Hofmann, M. 2013. Identifying and visualizing the similarities between course content at a learning object, module and program level. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 320-321.

[8] Gutierrez-Santos, S., Mavrikis, M., Poulovassilis, A. and Zhu, Z. 2014. Indicator visualisation for adaptive exploratory learning environments. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 377-378.

[9] Johnson, M.W. and Barnes, T. 2010. EDM visualization tool: Watching students learn. In *Proceedings of the Third International Conference on EDM* (Pittsburgh, PA, USA, June 11 - 13, 2010). EDM'10. 297-298.

[10] Johnson, M.W., Eagle, M.J., Joseph, L. and Barnes, T. 2011. The EDM Vis tool. In *Proceedings of the Fourth*

International Conference on EDM (Eindhoven, The Netherlands, July 6 - 8, 2011). EDM'11. 349-350.

[11] Johnson, M.W, Eagle, M. and Barnes, T. 2013. InVis: An interactive visualization tool for exploring interaction networks. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 82-89.

[12] Johnson, M.W., Eagle, M., Stamper, J. and Barnes, T. 2013. An algorithm for reducing the complexity of interaction networks. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 248-251.

[13] Kitchenham, B. and Charters, S. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering.* Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University.

[14] Lynch, C.F. 2014. AGG: Augmented graph grammars for complex heterogeneous data. In *Extended Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 37-42.

[15] Lynch, C.F. and Ashley, K.D. 2014. Empirically valid rules for ill-defined domains. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 237-240.

[16] Martinez-Maldonado, R., Yacef, K. and Kay, J. 2013. Data mining in the classroom: discovering groups' strategies at a multi-tabletop environment. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 121-128.

[17] McTavish, T.S. 2014. Facilitating graph interpretation via interactive hierarchical edges. In *Extended Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 59-61.

[18] Merceron, A., Schwarzrock, S., Elkina, M., Pursian, A, Beuster, L., Fortenbacher, A., Kappe, L. and Wenzlaff B. 2013. Visual exploration of interactions and performance with LeMo. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 396-397.

[19] Olson, R., Daily, Z., Malayny, J. and Szkutak, R. 2013. Project CASSI: A social-graph based tool for classroom behavior analysis and optimization. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 398-399.

[20] Petersen, K., Feldt, R., Mujtaba, S. and Mattsson, M. 2008. Systematic mapping studies in software engineering. In *Proceedings of the Twelfth International Conference on Evaluation and Assessment in Software Engineering* (Bari, Italy, June 26 - 27, 2008).

[21] Portillo-Rodriguez, J., Vizcaino, A., Piattini, M. and Beecham, S. 2012. Tools used in Global Software Engineering: A systematic mapping review. *Information and Software Technology.* 54 (Mar. 2012), 663-685.

[22] Rabbany Khorasgani R., Takaffoli, M. and Zaiane, O.R. 2011. Analyzing participation of students in online courses using social network analysis techniques. In *Proceedings of the Fourth International Conference on EDM* (Eindhoven, The Netherlands, July 6 - 8, 2011). EDM'11. 21-30.

[23] Ray, F, Brawner, K. and Robson, R. 2014. Using data mining to automate ADDIE. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 429-430.

[24] Romashkin, N., Ignatov, D. and Kolotova, E. 2011. How university entrants are choosing their department? Mining of university admission process with FCA taxonomies. In *Proceedings of the Fourth International Conference on EDM* (Eindhoven, The Netherlands, July 6 - 8, 2011). EDM'11. 230-233.

[25] Romero, C., Gutierrez, S., Freire, M. and Ventura, S. 2008. Mining and visualizing visited trails in web-based educational systems. In *Proceedings of the First International Conference on EDM* (Montreal, Canada, June 20 - 21, 2008). EDM'08. 182-186.

[26] Romero, C. and Ventura, S. 2010. Educational data mining: A review of the state of the art. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews,* 40 (6), 601-618.

[27] Rus, V., Banjade, R., Lintean, M., Niraula, N. and Stefanescu, D. 2013. SEMILAR: A semantic similarity toolkit for assessing students' natural language inputs. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 402-403.

[28] Sheshadri, V., Lynch, C. and Barnes, T. 2014. InVis: An EDM tool for graphical rendering and analysis of student interaction data. In *Extended Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 65-69.

[29] Smith, A., Wiebe, E., Mott, B. and Lester, J. 2014. SKETCHMINER: Mining learner-generated science drawings with topological abstraction. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 288-291.

[30] Vaculik, K., Nezvalova, L. and Popelinsky, L. 2014. Graph mining and outlier detection meet logic proof tutoring. In *Extended Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 43-50.

[31] Xu, B. and Recker, M.M. 2010. Peer production of online learning resources: A social network analysis. In *Proceedings of the Third International Conference on EDM* (Pittsburgh, PA, USA, June 11 - 13, 2010). EDM'10. 315-316.

[32] Yamakawa, O., Tagawa, T., Inoue, H., Yastake, K. and Sumiya T. 2011. Combining study of complex network and text mining analysis to understand growth mechanism of communities on SNS. In *Proceedings of the Fourth International Conference on EDM* (Eindhoven, The Netherlands, July 6 - 8, 2011). EDM'11. 335-336.

A predictive model for identifying students with dropout profiles in online courses

Marcelo A. Santana
Institute of Computing
Federal University of Alagoas
marcelo.almeida@nti.ufal.br

Evandro B. Costa
Institute of Computing
Federal University of Alagoas
evandro@ic.ufal.br

Baldoino F. S. Neto
Institute of Computing
Federal University of Alagoas
baldoino@ic.ufal.br

Italo C. L. Silva
Institute of Computing
Federal University of Alagoas
italocarlo@nti.ufal.br

Joilson B. A. Rego
Institute of Computing
Federal University of Alagoas
jotarego@gmail.com

ABSTRACT

Online education often deals with the problem related to the high students' dropout rate during a course in many areas. There is huge amount of historical data about students in online courses. Hence, a relevant problem on this context is to examine those data, aiming at finding effective mechanisms to understand student profiles, identifying those students with characteristics to drop out at early stage in the course. In this paper, we address this problem by proposing predictive models to provide educational managers with the duty to identify students whom are in the dropout bound. Four classification algorithms with different classification methods were used during the evaluation, in order to find the model with the highest accuracy in prediction the profile of dropouts students. Data for model generation were obtained from two data sources available from University. The results showed the model generated by using SVM algorithm as the most accurate among those selected, with 92.03% of accuracy.

Keywords

Dropout, Distance Learning, Educational Data Mining, Learning Management Systems

1. INTRODUCTION

Every year, the registration marks in E-learning modality has increased considerably, in 2013, 15.733 courses were offered, in E-learning or semi-presence modality. Furthermore, the institutions are very optimistic, 82% of researched places, believe that the amount of registration marks will have a considerable expansion in 2015 [1], showing the E-learning evolution and its importance as a tool for citizen's formation. The Learning Management Systems (LMS) [15] can be considered one of factors that has had an important role for popularization of this learning modality [1].

Despite the rapid growth of online courses, there has also been rising concern over a number of problems. One issue in particular that is difficult to ignore is that these online courses also have high dropout rates. Specifically, in Brazil, in 2013, according with the latest Censo, published by the E-learning Brazilian Association (ABED), the dropout average was about 19,06% [1].

Beyond the hard task on identifying the students who can have possible risk of dropping out, the same dropout also brings a huge damage to current financial and social resources. Thus, the society also loses when they are poorly managed, once the student fills the vacancy but he gives up the course before the end.

Online education often deals with the problem related to the high students' dropout rate during a course in many areas. There is huge amount of historical data about students in online courses. Hence, a relevant problem on this context is to examine those data, aiming at finding effective mechanisms to understand student profiles, identifying those students with characteristics to drop out at early stage in the course.

In this paper, we address this problem by proposing predictive models to provide educational managers with the duty of identifying students who are in the dropout bound. This predictive model took in consideration academic elements related with their performance at the initial disciplines of the course. Data from System Information course at Federal University of Alagoas (UFAL) were used to build this model, which uses a very known LMS, called Moodle.

A tool to support the pre-processing phase was used in order to prepare data for application of Data Mining algorithms. The Pentaho Data Integration [2] tool covers the extraction areas, transformation and data load (ETL), making easier the archive generation in the compatible format with the data mining software adopted, called WEKA[5].

Therefore, for what was exposed above, it justifies the needing of an investment to develop efficient prediction methods, assessment and follow up of the students with dropout risk, allowing a future scheduling and adoption of proactive measures aiming the decrease of the stated condition.

The rest of the paper is organized as follows. Section 2 presents some related work. Section 3 Environment for Construction of predictive model. Afterwards, we present the experiment settings in Section 4, and in Section 5 we discuss the results of the experiment. Section 6 presents some concluding remarks and directions of future work.

2. RELATED WORK

Several studies have been conducted in order to find out the reasons of high dropout indices in online courses. Among them, Xenos [18] makes a review of the Open University students enrolled in a computing course. In this studies, five acceptable reasons, that might have caused the dropout, were identified: Professional (62,1%), Academic (46%), Family (17,8%), Health Issues (9,5%), Personal Issues (8,9%). According to Barroso and Falcão (2004) [6] the motivational conditions to the dropout are classified in three groups: i) Economic - Impossibility of remaining in the course because of socio-economics issues; ii) Vocational - The student is not identified with the chosen course. iii) Institutional - Failure on initial disciplines, previous shortcomings of earlier contents, inadequacy with the learning methods.

Manhães et al.[14] present a novel architecture that uses EDM techniques to predict and identify those who are at dropout risk. The paper shows initial experimental results using real world data about of three undergraduate engineering courses of one the largest Brazilian public university. According to the experiments, the classifier Naive Bayes presented the highest true positive rate for all datasets used in the experiments.

A model for predicting students' performance levels is proposed by Erkan Er [9]. Three machine learning algorithms were employed: instance-based learning Classifier, Decision Tree and Naive Bayes. The overall goal of the study is to propose a method for accurate prediction of at-risk students in an online course. Specifically, data logs of LMS, called METU-Online, were used to identify at-risk students and successful students at various stages during the course. The experiment were realized in two phases: testing and training. These phases were conducted at three steps which correspond to different stages in a semester. At each step, the number of attributes in the dataset had been increased and all attributes were included at final stage. The important characteristic of the dataset was that it only contained time-varying attributes rather than time-invariant attributes such as gender or age. According to the author, these data did not have significant impact on overall results.

Dekker [8] in your paper presents a data mining case study demonstrating the effectiveness of several classification techniques and the cost-sensitive learning approach on the dataset from the Electrical Engineering department of Eindhoven University of Technology. Was compared two decision tree algorithms, a Bayesian classifier, a logistic model, a rule-based learner and the Random Forest. Was also considered the OneR classifier as a baseline and as an indicator of the predictive power of particular attributes. The experimental results show that rather simple classifiers give a useful result with accuracies between 75 and 80% that is hard to beat with other more sophisticated models. We demonstrated that cost-sensitive learning does help to bias classification errors towards preferring false positives to false negatives. We believe that the authors could get better results by making some adjustments to the parameters of the algorithms.

Jaroslav [7], aims to research to develop a method to classify students at risk of dropout throughout the course. Using personal data of students enriched with data related to social behaviours, Jaroklav uses dimensionality reduction techniques and various algorithms in order to find which of the best results managing to get the accuracy rates of up to 93.51%, however the best rates are presented at the end of the course. Whereas the goal is to identify early on dropout, the study would be more relevant if the best results were obtained results at the beginning of the course.

In summary, several studies investigating the application of EDM techniques to predict and identify students who are at risk dropout. However, those works share similarities: (i) identify and compare algorithm performance in order to find the most relevant EDM techniques to solve the problem or (ii) identify the relevant attributes associated with the problem. Some works use past time-invariant student records (demographic and pre-university student data). In this study, contribution to those presented in this section, makes the junction between two different systems, gathering a larger number of attributes, variables and time invariant. Besides being concerned with the identification and comparison of algorithms, identify the attributes of great relevance and solve the problem the predict in more antecedence the likely to dropout students.

3. ENVIRONMENT FOR CONSTRUCTION OF PREDICTIVE MODEL

This subsection presents an environment for construction for a predictive model for supporting educators in the task of identifying prospective students with dropout profiles in online courses. The environment is depicted in Figure 1.

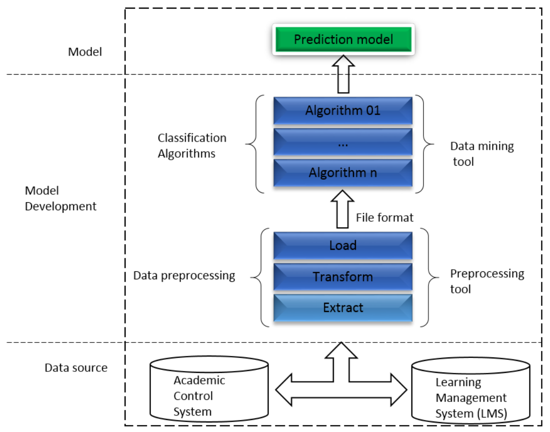


Figure 1: Environment for Construction of predictive model

The proposed environment in this work is composed by three layers: Data source, Model development and Model. The data sources are located in the first layer. Data about all students enrolled at the University are stored in two data sources: The first one contains students' personal data, for example: age, gender, income, marital status and grades from the academic control system used by the University. Information related with frequency of access, participation, use of the tools available, and grades of students related the activities proposed within the environment are kept in second data source.

In the second layer, the pre-processing [11] activity over the data is initiated. Sequential steps are executed in this layer in order to prepare them to data mining process. In the original data some information can not be properly represented in a expected format by data mining algorithm, data redundancy or even data with some kind of noise. These problems can produce misleading results or make the algorithm execution becomes computationally more expensive.

This layer is divided into the following stages: data extraction, data cleaning, data transformation, data selection and the choice of algorithm that best fits the model. Just below, will be displayed briefly each step of this layer.

Data extraction: The extraction phase establishes the connection with the data source and performs the extraction of the data.

Data cleaning: This routine tries to fill missing values, smooth out noise while identifying outliers, and correct data inconsistencies.

Data transformation: In this step, data are transformed and consolidated into appropriate forms for mining by performing summary or aggregation operations. Sometimes, data transformation and consolidation are performed before the data selection process, particularly in the case of data warehousing. Data reduction may also be performed to obtain a smaller representation of the original data without sacrificing its integrity.

Data selection: In this step, relevant data to the analysis task are retrieved from the database.

Choice of algorithm: An algorithm to respond with quality in terms of accuracy, which has students elusive profile, was considered the algorithm that best applies to the model.

Finally, the last layer is the presentation of the model. This layer is able to post-processing the result obtained in the lower layer and presenting it to the end-user of a most understandable way.

4. EXPERIMENT SETTINGS

The main objective of this present research is to build a predictive model for supporting educators in the hard task of identifying prospective students with dropout profiles in online courses, using Educational Data Mining (EDM) techniques [16]. This section is organized as follows: Section 4.1 describes the issue which drives our assessment. Section 4.2 shows which data were selected for to the data group utilized in the experiment and which algorithms were chosen for data mining execution. Section 4.3 indicates the employed tools during the execution of experiment. Finally, Section 4.4 shows every step in experiment execution, including data consolidation, data preprocessing and algorithms execution.

4.1 Planning

The research question that we would like to answer is:

RQ. Is our predictive model able to early identify the students with dropout risk?

In order to answer this question, EDM techniques with four different classification methods were used, aiming to get a predictive model which answers us with quality in precise ways which students have a dropout profile, taking in consideration only data about the initial disciplines of a specified course.

4.2 Subject Selection

4.2.1 Data Selection

The Federal University of Alagoas offers graduation courses, postgraduate courses and E-learning courses. In the on line courses, there are more than 1800 registered students[4].

An E-learning course is usually partitioned in semesters, where different disciplines are taught along these semesters. Each semester usually has five disciplines per semester, and each discipline has a duration between five to seven weeks. Anonymous data, from the Information Systems E-learning course, were selected from this environment, relative to first semester in 2013. Data of one discipline (Algorithm and Data Structure I), chosen based on its relevance, were analysed. Such discipline has about 162 students enrolled.

4.2.2 Machine Learning Algorithms Selection

In this work to predict student dropouts, four machine learning algorithms were used, using different classification methods. The methods used were: simple probabilistic classifier based on the application of Bayes' theorem, decision tree, support vector's machine and multilayer neural network.

These techniques have been successfully applied to solve various classification problems and function in two phases: (i) training and (ii) testing phase. During the training phase each technique is presented with a set of example data pairs (X, Y), where X represents the input and Y the respective output of each pair [13]. In this study, Y can receive one of the following values, "approved" or "reproved", that corresponds the student situation in discipline.

4.3 Instrumentation

The Pentaho Data Integration [2] tool was chosen to realize all preprocessing steps on selected data. Pentaho is a opensource software, developed in Java, which covers extraction areas, transform and load of the data [2], making easier the creation of an model able to : (i) extract information from data sources, (ii) attributes selection, (iii) data discretization and (iv) file generation in a compatible format with the data mining software.

For execution of selected classification algorithms (see Section 4.2.2), the data mining tool Weka was selected. Such algorithms are implemented on Weka software as NaiveBayes (NB), J48 (AD), SMO (SVM), MultilayerPerceptron (RN)

[17] respectively. Weka is a software of open code which contains a machine learning algorithms group to data's mining task [5].

Some features were taken in consideration for Weka [10] adoption, such as: ease of acquisition, facility and availability to directly download from the developer page with no operation cost; Attendance of several algorithms versions set in data mining and availability of statistical resources to compare results among algorithms.

4.4 Operation

The evaluation of experiment was executed on HP Probook 2.6 GHz Core-I5 with 8Gb of memory, running Windows 8.1.

4.4.1 Data's Preprocessing

Real-world data tend to be dirty, incomplete, and inconsistent. Data preprocessing techniques can improve data quality, thereby helping to improve the accuracy and efficiency of the subsequent mining process [11].

Currently, the data is spread in two main data sources: LMS Moodle, utilized by the University as assistance on E-learning teaching, including data which show the access frequency, student's participation using the available tools, as well as the student's success level related to proposed activities. Meanwhile, student's personal files as age, sex, marital status, salary and disciplines grades are kept in the Academic Control System (ACS), which is a Software designed to keep the academic control of the whole University [4].

Aiming to reunite a major data group and work only with relevant data to the research question that we want to answer, we decided to perform consolidation of these two data source in a unique major data source, keeping their integrity and ensuring that only relevant information will be used during data mining algorithms execution.

Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the data mining process [11].

To maintain the integrity and reliability between data, a mandatory attribute, with unique value and present between in both data sources, was chosen. Thus, the CPF attribute was chosen to make data unification between the two selected data sources, once it permits the unique identification among selected students.

In order to facilitate algorithms execution and comprehension of results,predicting the dropout in an early stage of the study. In order to achieve a high rate of accuracy and minimum of false negatives, i.e. students that have not been recognized to be in danger of dropout. Some attributes were transformed, as we can seen below:

- The corresponding attributes related with discipline grades were discretized in a five-group-value (A,B,C,D e E), depending on the discipline's achieved grades. The student with a grade higher or equal 9, was allocated for "A" group. Those ones who had their grades between 8,99 and 7 were allocated for "B" group. the "C" students are those that had a grade between 6,99 and 5, and those who had grades under 5,99 stayed at "D" group and finally those that doesn't have a grade associated were allocated in "E" group.

- Every student was labelled as approved or reproved based on the situation informed by the academics registers. The final score of each discipline is composed by two tests, if the student did not succeed in obtaining the minimum average, he will be leaded to the final reassessment and final test.

- In the "City" attribute, some inconsistencies were found, where different data about the same city were registered in database. For instance, the instances of **Ouro Branco** and **Ouro Branco/AL** are related to same city. This problem was totally solved, with application of techniques for grouping attributes.

- The attribute "age" had to be calculated. For this, the student's birth date, registered in database, was taken in consideration.

When all the attributes were used the accuracy was low. That is why we utilized feature selection methods to reduce the dimensionality of the student data extracted from dataset. We improved the pre-processing method the data.

In order to preserve reliability of attributes for classification after the reduction. We use InfoGainAttributeEval algorithm that builds a rank of the best attributes considering the extent of information gain based on the concept of entropy.

After this procedure, we reduced the set of attributes from 17 to 13 most relevant. The list of the refined set of attributes in relevance ordercan be found in Table 1.

Table 1: Selected Attributes

Attributes	Description
AB1	First Evaluation Grade
Blog	Post count and blog view
Forum	Post count and forum views
Access	Access Count in LMS
Assign	Sent files count e viewed
City	City
Message	Count of sent messages
Wiki	Post count and wiki view
Glossary	Post count and glossary view
Civil status	Civil status
Gender	Gender
Salary	Salary
Status	Status on discipline

Taking in consideration that the main objective is to predict student's final situation with the earlier advance as possible inside the given discipline, to this study we will only use data until the moment of the first test.

The Figure 2 presents all the executed stages, during the preprocessing phase, in order to generate a compatible file with the mining software.

4.4.2 Algorithms Execution

The k-fold method was applied to make a assessment the model generalization capacity, with k=10 (10-fold cross validation). The cross validation method, consists in splitting of the model in k subgroups mutually exclusive and with the same size, from these subgroups, one subgroup is selected for test and the remaining k-1's are utilized for training. The average error rate of each training subgroup can be used as an estimate of the classifier's error rate. When Weka implements the cross validation, it trains the classifier k times to calculate the average error rate and finally, leads the build classifier back utilizing the model as a training group. Thus, the average error rate provides a better solution in terms of classifier's error accuracy reliability [12].

In order to get the best results of the algorithms without losing generalization, some parameters of SVM algorithms were adjusted.

The first parameter was set the parameter "C". This parameter is for the soft margin cost function, which controls the influence of each individual support vector; this process involves trading error penalty for stability [3].

The default kernel used by Weka tool is the polynomial we changed to the Gaussian setting the parameters Gamma. Gamma is the free parameter of the Gaussian radial basis function [3].

After several adjustments to the values of the two parameters mentioned above, which showed the best results in term of accuracy and lower false positive rate, was C = 9.0 and Gamma = 0.06 parameter.

For comparison of results related to selected algorithms, we used Weka Experiment Environment (WEE). The WEE allows the selection of one or more algorithms available in the tool as well as analyse the results, in order to identify, if a classifier is, statistically, better than the other. In this experiment, the cross validation method, with the parameter "k=10" [5], is used in order to calculate the difference on the results in each one of the algorithms related to a chosen standard algorithm (baseline).

5. RESULTS AND DISCUSSIONS

In this section, the results of the experiment, described in Section 4, are analyzed.

The WEE tool calculated the average accuracy of each classifier. Table 2 shows the result of each algorithms execution. The accuracy represents the percentage of the test group instance which are correctly classified by the model built during training phases. If the built model has a high accuracy, the classifier is treated as efficient and can be put into production [11].

Comparing the results among the four algorithms, we can verify that the accuracy oscillates around 85.5 to 92.03%. Furthermore, a classifier which has a high error rate to false positives is not suitable to our solution. In this case, we have considered the algorithm which has the lower false positive rates.

Table 2: Accuracy and rates

Classifiers	NB	AD	SVM	RN
Accuracy	85.50	86.46	92.03	90.86
True Positives	0.76	0.77	0.88	0.85
False Negatives	0.24	0.23	0.12	0.15
True Negatives	0.89	0.91	0.94	0.93
False Positives	0.11	0.09	0.06	0.07

As we can see on table 2 the algorithm SVM presented a low false positive rate and better accuracy. Therefore, only the best algorithm was considered to our solution. The Naive Bayes classifier had the worst result in terms of accuracy and a high false positive rate. The other ones had an error average of 8%, and then, we end up with 8% of the students with dropout risk not so correctly classified.

5.1 Research Question

As can be seen in table 2, in our experiment, the SVM algorithm obtained 92% of accuracy. According to Han J. *et al.* [11] if the accuracy of the classifier is considered acceptable, the classifier can be used to classify future data tuples for which the class label is not known. Thus, the results are pointing to the viability of model able to early identify a possible student's dropout, based on their failures in the initial disciplines.

5.2 Statistical Significance Comparision

We often need compare different learning schemes on the same problem to see which is the better one to use. This is a job for a statistical device known as the t-test, or Student's t-test. A more sensitive version of the t-test known as a paired t-test it was used. [17]. Using this value and desired significance level (5%), consequently one can say that these classifiers with a certain degree of confidence (100 - significance level) are significantly different or not. By using the t-test paired in the four algorithms, performed via Weka analysis tool, observed that the SVM algorithm is significantly respectful of others.

5.3 Threats to validity

The experiment has taken in consideration data from the Information System course and the Data Structure Algorithm discipline. However, the aforementioned discipline was chosen, based on its importance in the context of Information System course.

6. CONCLUSION AND FUTURE WORK

Understand the reasons behind the dropout in E-learning education and identify in which aspects can be improved is a challenge to the E-learning. One factor, which has been pointed as influencer of students' dropout, is the academic element related with their performance at the initial disciplines of the course.

This research has addressed dropout problem by proposing predictive models to provide educational managers with the duty to identify students whom are in the dropout bound.

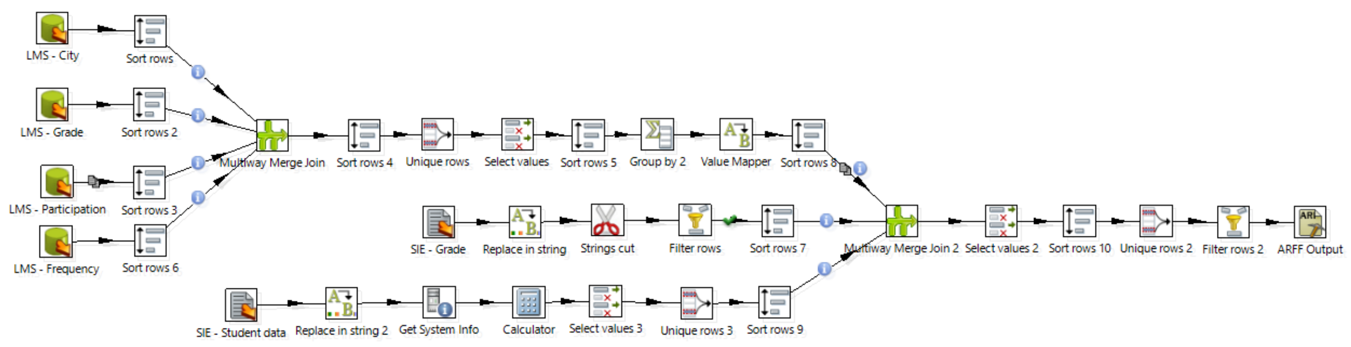


Figure 2: Steps Data Preprocessing

The adopted approach allowed us to perform predictions at an initial discipline phase. The preliminaries results has shown that prediction model to identify students with dropout profiles is feasible. These predictions can be very useful to educators, supporting them in developing special activities for these potential students, during the teaching-learning process.

As an immediate future work, some outstanding points still should be regarded to the study's improvement, as apply the same model in different institution databases with different teaching methods and courses, including new factors related to dropout as: professional, vocational and family data, execute some settings in algorithms' parameters in order to have the best achievements. Furthermore, a integrated software to LMS, to provide this feedback to educators, will be developed using this built model.

7. REFERENCES

[1] Abed - E-learning Brazilian Association. http://www.abed.org.br/. Accessed December 2014.

[2] Pentaho - Pentaho Data Integration. http://www.pentaho.com/. Accessed January 2015.

[3] SVM - support vector machines (svms). http://www.svms.org/parameters/. Accessed December 2014.

[4] UFAL - Federal University of Alagoas. http://www.ufal.edu.br/. Accessed January 2015.

[5] Weka - the University of Waikato. http://www.cs.waikato.ac.nz/ml/weka/. Accessed January 2015.

[6] M. F. Barroso and E. B. Falcao. University dropout: the case of ufrj physics institute. *IX National Meeting of Research in Physics Teaching*, 2004.

[7] J. Bayer, H. Bydzovská, J. Géryk, T. Obsívac, and L. Popelínsky. Predicting drop-out from social behaviour of students. In A. H. M. Y. Kalina Yacef, Osmar Zaiane and J. Stamper, editors, *Proceedings of the 5th International Conference on Educational Data Mining - EDM 2012*, pages 103–109, Greece, 2012.

[8] G. Dekker, M. Pechenizkiy, and J. Vleeshouwers. Predicting students drop out: A case study. In T. Barnes, M. C. Desmarais, C. Romero, and S. Ventura, editors, *EDM*, pages 41–50, 2009.

[9] E. Er. Identifying at-risk students using machine learning techniques: A case study with is 100. In *International Journal of Machine Learning and Computing*, pages 476–481, Singapore, 2012. IACSIT Press.

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

[11] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[12] S. B. Kotsiantis, C. Pierrakeas, and P. E. Pintelas. Preventing student dropout in distance learning using machine learning techniques. In V. Palade, R. J. Howlett, and L. C. Jain, editors, *KES*, volume 2774 of *Lecture Notes in Computer Science*, pages 267–274. Springer, 2003.

[13] I. Lykourentzou, I. Giannoukos, V. Nikolopoulos, G. Mpardis, and V. Loumos. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Comput. Educ.*, 53(3):950–965, Nov. 2009.

[14] L. M. B. Manhães, S. M. S. da Cruz, and G. Zimbrão. Wave: An architecture for predicting dropout in undergraduate courses using edm. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 243–247, New York, NY, USA, 2014. ACM.

[15] M. Pretorius and J. van Biljon. Learning management systems: Ict skills, usability and learnability. *Interactive Technology and Smart Education*, 7(1):30–43, 2010.

[16] C. Romero and S. Ventura. Educational data mining: A review of the state of the art. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(6):601–618, Nov 2010.

[17] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[18] M. Xenos, C. Pierrakeas, and P. Pintelas. A survey on student dropout rates and dropout causes concerning the students in the course of informatics of the Hellenic Open University. *Computers Education*, 39(4):361 – 377, 2002.

Automatic Age Detection Using Text Readability Features

Avar Pentel
Tallinn University,Tallinn, Estonia
+372 51 907 739
pentel@tlu.ee

ABSTRACT

In this paper, we present the results of automatic age detection based on very short texts as about 100 words per author. Instead of widely used n-grams, only text readability features are used in current study. Training datasets presented two age groups - children and teens up to age 16 and adults 20 years and older. Logistic Regression, Support Vector Machines, C4.5, k-Nearest Neighbor, Naïve Bayes, and Adaboost algorithms were used to build models. All together ten different models were evaluated and compared. Model generated by Support Vector Machine with Adaboost yield to f-score 0.94, Logistic regression to 0.93. A prototype age detection application was built using the best model.

Keywords

Automatic age detection, readability features, logistic regression, support vector machines, Weka.

1. INTRODUCTION

One important class of information in user modeling is related to user age. Any adaptive technology can use age prediction data. In educational context automatic tutoring systems and recommendation systems, can benefit on age detection.

Automatic age detection has also utilities in crime prevention. With widespread of social media, people can register accounts with false age information about themselves. Younger people might pretend to be older in order to get access to sites that are otherwise restricted to them. In the same time older people might pretend to be younger in order to communicate with youngster. As we can imagine, this kind of false information might lead to serious threats, as for instance pedophilia or other criminal activities.

But besides serious crime prevention, automatic age detection can by used by educators as indirect plagiarism detector. While there are effective plagiarism detection systems, they do not work when parents are doing pupils homework or students are using somebody else's original work, which is not published anywhere. There are closed communities where students can buy homework's for any topic.

Full scale authorship profiling is not an option here, because large amount of author texts is needed. Some authors [1] argue, that at least 10000 words per author is needed, other that 5000 [2]. But if we think about business purpose of this kind of age detector, especially when the purpose is to avoid some criminal acts, then there is no time to collect large amount of text written by particular user.

When automatic age detection studies fallow authorship profiling conventions then it is related to second problem – the features, widely used in authorship profiling, are semantic features. Probability that some sequence of words, even a single word, occur in short text is too low and particular word characterizes better the context [3] than author. Some authors use character n-grams frequencies to profile users, but again, if we speak about texts that are only about 100 words long, these features can also be very context dependent.

Semantic features are related to third problem - they are costly. Using part of speech tagging systems to categorize words and/or large feature sets for pattern matching, takes time and space. If our goal is to perform age detection fast and online then it is better to have few features that can be extracted instantly on client side.

In order to avoid all three previously mentioned shortcomings, we propose other set of features. We call them readability features, because they are previously used to evaluate texts readability. Texts readability indexes are developed already before computerized text processing, so for example Gunning Fog index [4] takes into account complex (or difficult) words, those containing 3 or more syllables and average number of words per sentence. If sentence is too long and there are many difficult words, the text is considered not easy to read and more education is needed to understand this kind of text. Gunning Fog index is calculated with a formula (1) below:

$$GunningFogIndex = 0.4 \times \left[\left(\frac{words}{sentences} \right) + 100 \times \left(\frac{complexwords}{words} \right) \right] \quad (1)$$

We suppose that authors reading skills and writing skills are correlated and by analyzing author's text readability, we can infer his/her education level, which at least to the particular age is correlated with actual age of an author. As readability indexes work reliably on texts with about 100 words, these are good candidates for our task with short texts.

As a baseline we used n-gram features in pre testing. Comparing readability features with n-gram features, we found that with wider age gap between young and adult groups, readability features making better classifiers if using short texts [5]. Now we continue this work with larger dataset and with readability features only.

Using best fitting model, we created an online prototype age detector.

Section 2 of this paper surveys the literature on age prediction. In Section 3 we present our data, features, used machine learning algorithms, and validation. In Section 4 we present our classification results and prototype application. We conclude this paper in Section 5 by summarizing and discussing our study.

2. RELATED WORKS

In this section we review related works on age- and other author-specific profiling. There are no studies that dealing particularly with effect of text sizes in context of age detection. In previous section we mentioned that by literature for authorship profiling 5000 to 10000 words per author is needed [1,2]. Luyckx and

Daelemans [6] reported a dramatic decrease of the performance of the text categorization, when reducing the number of words per text fragment to 100. As authorship profiling and authors age prediction is not the same task, we focus on works that dealing particularly with user age.

The best-known age based classification results are reported by Jenny Tam and Craig H. Martell [7]. They used age groups 13-19, 20-29, 30-39, 40-49 and 50-59. All age groups were in different size. As features word and character n-grams were used. Additionally they used emoticons, number of capital letters and number of tokens per post as features. SVM model trained on youngest age group against all others yield to f-score 0,996. Moreover this result seems remarkable, while no age gap between two classes was used.

However we have to address to some limitations of their work that might explain high f-scores. Namely they used unbalanced data set (465 versus 1263 in training data set and 116 versus 316 in test set). Unfortunately their report gave only one f-score value, but no confusion matrices, ROC or Kappa statistics. We argue, that with unbalanced data sets, single f-score value is not sufficient to characterize the models accuracy. In such test set – 116 teenagers versus 316 adults - the f-score 0.85 (or 0.42 depending of what is considered positive result) will simply be achieved by model that always classifies all cases as adults. Also, it is not clear if reported f-score is weighted average of two classes' f-scores or presenting only one class f-score. Secondly it is not clear if given f-score was result of averaging cross validation results.

It is worth of mentioning, that Jane Lin [8], used the same dataset two years earlier in her postgraduate thesis supervised by the Craig Martell, and she achieved more modest results. Her best average f-score in teens versus adult's classification with SVM model was 0.786 as compared to Tam's and Martell reported 0.996. But besides averaged f-scores, Jane Lin also reported lowest and highest f-scores, and some of her highest f-scores were indeed 0.996 as reported in Tam and Martell paper.

Peersman et al [9] used large sample 10,000 per class and extracted up to 50,000 features based on word and character n-grams. Report states, that they used posts average of 12,2 tokens. Unfortunately it is not clear if they combined several short posts from the same author, or used single short message as a unique instance in feature extraction. They tested three datasets with different age groups –11-15 versus 16+, 11-15 versus 18+ and 11-15 versus 25+. Also experimentations carried out with number of features, and training set sizes. Best SVM model and with largest age gap, largest dataset and largest number of features yield to f-score 0.88.

Santosh, et al [10,11] used word n-grams as content-based features and POS n-grams as style based features. They tested three age groups 13-17, 23-27, and 33-47. Using SVM and kNN models, best classifiers achieved 66% accuracy.

Marquart [12] tested five age groups 18-24, 25-34, 35-49, 50-64, and 65-xx. Used dataset was unbalanced and not stratified. He also used some of the text readability features as we did in current study. Besides of readability features, he used word n-grams, HTML tags, and emoticons. Additionally he used different tools for feature extraction like psycholinguistic database, sentiment strength tool, linguistic inquiry word count tool, and spelling and grammatical error checker. Combining all these features, his model yield to modest accuracy of 48,3%.

Dong Nguyen and Carolyn P. Rose [13] used linear regression to predict author age. They used large dataset with 17947 authors with average text length of 11101 words. They used as features word unigrams and POS unigrams and bigrams. Text was tagged using the Stanford POS tagger. Additionally they used linguistic inquiry word count tool to extract features. Their best regression model had r^2 value 0.551 with mean absolute error 6.7.

As we can see, most of previous studies are using similar features, word and character n-grams. Additionally special techniques were used like POS tagging, Spell Checker, and Linguistic inquiry word count tool to categorize words. While text features extracted by this equipment are important, they are costly to implement in real life online systems. Similarly large feature sets up to 50,000 features, most of which are word n-grams, means megabytes of data. Ideally this kind of detector could work using client browser resources (JavaScript), and all feature extraction routines and models have to be as small as possible.

Summarizing previous work in the following table (1), we don't list all possible features. So for example features that are generated using POS tagging or features generated some word databases are all listed here as word n-grams. Last column gives f-score or the accuracy (with %) according to what characteristic was given in paper. Most of papers reported many different results, and we list in this summary table only the best result.

Table 1. Summary of previous work

Authors	Used feature types				training dataset size	avg. words per author	separation gap (year)	result f-score or accuracy (%)
	readability	word n-grams	char n-grams	emoticons				
Nguyen (2011)		x			17947*	11101	0	55.1%
Marquardt (2014)	x	x		x	7746	N/a	0	47.3%
Peersman (2011)		x	x		20000	12.2**	9	0.917
Lin (2007)		x		x	1728*	343	0	0.786
Tam & Martell (2009)		x	x	x	1728*	343	0	0.996***
Santosh (2014)		x			236600*	335	5	66%
This Study	**x**				**500**	**93**	**4**	**0.94**

*unbalanced datasets

**12.2 words was reported average message length, but it is not clear if only one message per user was used or user text was composed form many messages.

***not enough data about this result

3. METHODOLOGY

3.1 Sample & Data

We collected short written texts in average 93 words long from different social media sources like Facebook, Blog comments, and Internet forums. Additionally we used short essay answers from school online feedback systems and e-learning systems, and e-mails. No topic specific categorization was made. All authors were identified and their age fall between 9 and 46 years. Most authors in our dataset were unique, but we used multiple texts from the same author only in case, when the texts were written in different age. All texts in the collections were written in the same language (Estonian). We chose balanced and stratified datasets with 500 records and with different 4-year age gaps.

3.2 Features

In current study we used in our training dataset different readability features of a text. Readability features are quantitative data about texts, as for instance an average number of characters in the word, syllables in the word, words in the sentences, commas in the sentence and the relative frequency of the words with 1, 2,.., n syllable. All together 14 different features were extracted from each text plus classification variable (to which age class text author belongs).

In all features we used only numeric data and normalized the values using other quantitative characteristics of the text.

Used Feature set with explanations is presented in Table 2:

Table 2. Used features with calculation formulas and explanations

Feature	Explanation
Average number of Characters in Word	$= \frac{NumberOfCharactersInText}{NumberOfWordsInText}$ We excluded all white space characters when counting number of all characters in text
Average number of Words in Sentence	$= \frac{NumberOfWordsInText}{NumberOfSentencesInText}$
Complex Words to all Words ratio	$= \frac{NumberOfComplexWordsInText}{NumberOfWordsInText}$ Complex word is loan from Cunning Fog Index, where it means words with 3 or more syllables. As Cunning Fog index was designed for English, and Estonian language has as average more syllables per word, we raised the number of syllables according to this difference to five. Additionally we count the word complex if it has 13 or more characters.
Average number of Complex Words in Sentence	$= \frac{NumberOfComplexWordsInText}{NumberOfSentencesInText}$
Average number of Syllables per Word	$= \frac{NumberOfSyllablesInText}{NumberOfWordsInText}$
Average number of Commas per Sentence	$= \frac{NumberOfCommasInText}{NumberOfSentencesInText}$
One Syllable Words to all Words ratio	$= \frac{NumberOfWordsWith1syllableInText}{NumberOfWordsInText}$
Similarly as previous feature, we extracted 7 features for words containing 2, 3, 4 to 8 and more syllables.	$= \frac{NumberOfWordsWith\_N-SyllableInText}{NumberOfWordsInText}$ Novel syllable counting algorithm was designed for Estonian language, which is only few lines length and does not include any word matching techniques

3.3 Data Preprocessing

We stored all the digitalized texts in the local machine as separate files for each example. A local program was created to extract all previously listed 14 features from each text file. It opened all files one by one; extracted features form each file, and stored these values in a row of a comma-separated file. In the end of every row it stored data about the age group. A new and simpler algorithm was created for syllable counting. Other analogues algorithms for Estonian language are intended to exact division of the word to syllables, but in our case we are only interested on exact number of syllables. As it turns out, syllable counting is possible without knowing exactly where one syllable begins or ends.

In order to illustrate our new syllable counting algorithm, we give some examples about syllables and related rules in Estonian language. For instance the word *rebane* (fox) has 3 syllables: *re – ba – ne*. In cases like this we can apply one general rule – when single consonant is between vowels, then new syllable begins with that consonant.

When in the middle of word two or more consecutive consonants occur, then usually the next syllable begins with last of those consonants. For instance the word *kärbes* (fly) – is split as *kär-bes*, and *kärbsed* (flies) is split as *kärb-sed.* The problem is that this and previous rule does not apply to compound words. So for example, the word *demokraatia* (democracy) is split before two consecutive consonants as *de-mo-kraa-tia*.

Our syllable counting algorithm deals with this problem by ignoring all consecutive consonants. We set syllable counter on zero and start comparing two consecutive characters in the word, first and second character, then second and third and so on. General rule is, that we count a new syllable, when the tested pair of characters is vowel fallowed by consonant. The exception to this rule is the last character. When the last character is vowel, then one more syllable is counted.

Implemented syllable counting algorithm as well as other automatic feature extraction procedures can be seen in section 4.3 and in the source code of the prototype application.

3.4 Machine Learning Algorithms and Tools

For classification we tested six popular machine-learning algorithms:

- Logistic regression
- Support Vector Machine
- C4.5
- k-nearest neighbor classifier
- Naive Bayes
- AdaBoost.

Motivation of choosing those algorithms is based on literature [14,15]. The suitability of listed algorithms for given data types and for given binary classification task was also taken in to account. Last algorithm in the list – Adaboost – is actually not classification algorithm itself, but an ensemble algorithm, which is intended for use with other classifying algorithms, in order to make a weak classifier stronger. In our task we used Java implementations of listed algorithms that are available in freeware data analysis package Weka [16].

3.5 Validation

For evaluation we used 10 fold cross validation on all models. It means that we partitioned our data to 10 even sized and random parts, and then using one part for validation and other 9 as training dataset. We did so 10 times and then averaged validation results.

3.6 Calculation of final f-scores

Our classification results are given as weighted average f-scores. F-score is a harmonic mean between precision and recall. Here is given an example how it is calculated. Let suppose we have a dataset presenting 100 teenagers and 100 adults. And our model classifies the results as in fallowing Table 3:

Table 3. Example illustrating calculation of f-scores

Classified as =>	teenagers	adults
teenagers	88	12
adults	30	70

When classifying teenagers, we have 88 true positives (teenagers classified as teenagers) and 30 false positives (adults classified as teenagers). We also have 12 false negatives (teenagers classified as not teenagers) and 70 true negatives (adults classified as not teenagers). In following calculations we use abbreviations: TP = true positive; FP = false positive; TN = true negative; FN = false negative.

Positive predictive value or precision for teenagers' class is calculated by formula 2.

$$precision = \frac{TP}{TP + FP} = \frac{88}{88 + 30} = 0.746 \qquad (2)$$

Recall or sensitivity is the rate of correctly classified instances (true positives) to all actual instances in predicted class. Calculation of recall is given by formula 3.

$$recall = \frac{TP}{TP + FN} = \frac{88}{88 + 12} = 0.88 \qquad (3)$$

F-score is harmonic mean between precision and recall and it is calculated by formula 4.

$$f - score = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \qquad (4)$$

Using data in our example the f-score for teenager class will be 0.807, but if we do the same calculations for adult class then the f-score will be 0.769.

Presenting our results, we use a single f-score value, which is an average of both classes' f-score values.

4. RESULTS

4.1 Classification

Classification effect was related to placement of age separation gaps in our training datasets. We generated 8 different datasets by placing 4-year separation gap in eight different places. We generated models for all datasets, and present the best models' f-scores on figure 1. As we can see, our classification was most effective, when the age separation gap was placed to 16-19 years.

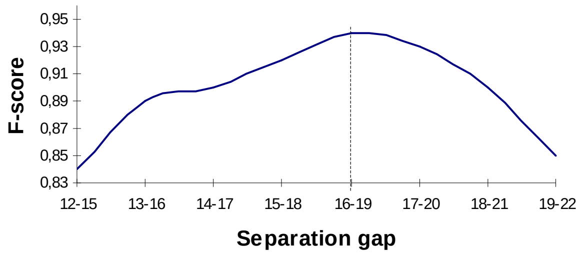


Figure 1. Effect of the position of separtion gap

With a best separation gap (16-19) between classes, Logistic regression model classified 93,12% of cases right, and Support Vector Machines generated model classified 91,74% of cases. Using Adaboost algorithm combined with classifier generated by Support Vector Machine yield to 94.03% correct classification and f-score 0.94. Classification models built by other algorithms performed less effectively as we can see in Table 4.

Results in fallowing table are divided in to two blocks. In the left side there are the results of the models generated by listed algorithms. In the right side there are the results of the models generated by Adaboost algorithm and the same algorithm listed in the row.

Table 4. Averaged F-scores of different models

	F-score	
		Using Adaboost
Logistic Regression	0.93	0.93
SVM (standardized)	0.92	0.94
KNN (k = 4)	0.86	0.86
Naïve Bayes	0.79	0.84
C4.5	0.75	0.84

As we can see in the table above, the best performers were classifiers generated by Logistic Regression algorithm and Support Vector Machine (with standardized data). In the right section of the table, where the effect of Adaboost algorithm is presented, we can see that Adaboost here cannot improve results with Logistic regression classifier, and kNN, but it improves results of SVM, Naïve Bayes and most significantly on C4.5. As Adaboost is intended to build strong classifiers out of weak classifiers, than the biggest effect on C4.5 is expectable. Two best performing classifiers remained still the same after using Adaboost, but now Support Vector Machine outperformed Logistic Regression by 0.91 percent points.

4.2 Features with highest impact

As there is relatively small set of readability features, we did not used any special feature selection techniques before generating models, and evaluating features on the basis of SVM model with standardized data. The strongest indicator of an age is the average number of words in sentence. Older people tend to write longer sentences. They also are using longer words. Average number of characters per word is in the second place in feature ranking. Best predictors of younger age group are frequent use of short words with one or two syllables.

In following Table (5), coefficients of standardized SVM model are presented.

Table 5. Features with highest impact in standardized SVM model

Coefficient	Feature
1.3639	Words in sentence
0.8399	Characters in word
0.258	Complex words in sentence
-0.2713	Ratio of words with 4 syllables
-0.3894	Commas per sentence
-0.7451	Ratio of words with 1 syllable
-0.762	Ratio of words with 2 syllables

4.3 Prototype Application

As the difference between performance of models generated by Adaboost with SVM and Logistic Regression is not significant, but as from the point of view of implementation, models without Adaboost are simpler, we decided to implement in our prototype application Logistic Regression model, which performed best without using Adaboost.[1] We implemented feature extraction routines and classification function in client-side JavaScript. Our prototype application uses written natural language text as an input, extracts features in exactly the same way we extracted features for our training dataset and predicts author's age class (Fig. 2.).

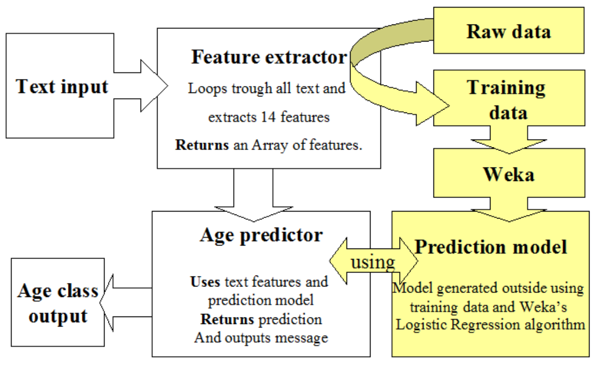

Text input
Feature extractor
Loops trough all text and
extracts 14 features
Returns an Array of features.
Raw data
Training data
Weka
Age predictor
Uses text features and prediction model
Returns prediction
And outputs message
using
Prediction model
Model generated outside using training data and Weka's Logistic Regression algorithm
Age class output

Figure 2. Application design

Our feature extraction procedure (Figure 3.) consists 3 stages:

1. Text input is split to sentences, and to words, and all excess white space chars are removed. Some simple features, number of characters, number of words, number of sentences, are also calculated in this stage.
2. In second stage syllables in words are counted.
3. All calculated characteristics are normalized using other characteristics of the same text. For example number of characters in text divided to number of words in text.

[1] http://www.tlu.ee/~pentel/age_detector/

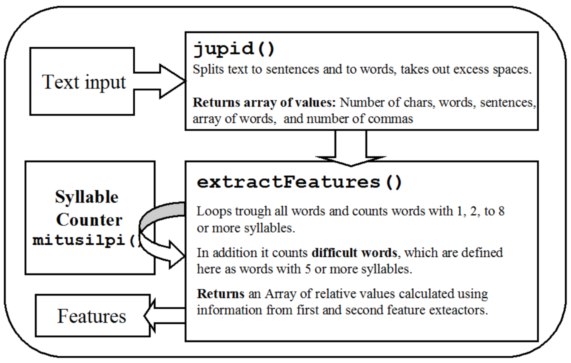

Text input
jupid()
Splits text to sentences and to words, takes out excess spaces.
Returns array of values: Number of chars, words, sentences, array of words, and number of commas
Syllable Counter
mitusilpi()
extractFeatures()
Loops trough all words and counts words with 1, 2, to 8 or more syllables.
In addition it counts difficult words, which are defined here as words with 5 or more syllables.
Returns an Array of relative values calculated using information from first and second feature exteactors.
Features

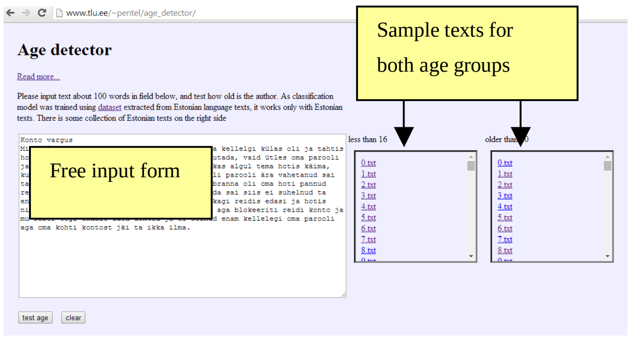
Figure 3. Feature Extractor

A new and simpler algorithm (5) was created for syllable counting. Other analogues algorithms for Estonian language are intended to exact division of the word to syllables, but in our case we are only interested on exact number of syllables. As it turns out, syllable counting is possible without knowing exactly where one syllable begins or ends. Unfortunately this is true only for Estonian (and maybe some other similar) language.

```
function number_of_syllables(w){              (5)

v="aeiouõäöü"; /* all vowels in Estonian lang. */

counter=0;

w=w.split('');/* creates char array of word */

wl=w.length; /* number of char’s in word */

  for(i=0; i < wl - 1; i++){

   if(v.indexOf(w[i])!=-1 && v.indexOf(w[i+1])==-1)

      counter++;

 /*

if char is vowel and next char is not, then count a
syllable (there are some exceptions to this rule, which
are easy to program).

*/

   }

  if( v.indexOf(w[wl-1]) != -1) counter++;

// if last char in the word is vowel, count new syllable

  return counter;

}
```

Implemented syllable counting algorithm as well as other automatic feature extraction procedures can be seen in the source code of the prototype application.[2]

Finally we created simple web interface, where everybody can test prediction by his/her free input or by copy-paste. As our classifier was trained on Estonian language, sample Estonian texts are provided on website for both age groups (Fig. 4.).



Please input text about 100 words in field below, and test how old is the author. As classification model was trained using dataset extracted from Estonian language texts, it works only with Estonian texts. There is some collection of Estonian texts on the right side

```
Konto vargus
Mi                   a kellelgi külas oli ja tahtis
ho                   utada, vaid ütles oma parooli
ja                   kas algul tema hotis käima,
ku                   li parooli ära vahetanud sai
ta                   branna oli oma hoti pannud
re                   da sai siis ei suhelnud ta
en                   kagi reidis edasi ja hotis
ni                   aga blokeeriti reidi konto ja
mu                   d enam kellelegi oma parooli
aga oma kohti kontost jäi ta ikka ilma.
```

Figure 4. Prototype application at http://www.tlu.ee/~pentel/age_detector/

5. DISCUSSION & CONCLUSIONS

Automatic user age detection is a task of growing importance in cyber-safety and criminal investigations. One of the user profiling problems here is related to amount of text needed to perform reliable prediction. Usually large training data sets are used to make such classification models, and also longer texts are needed to make assumptions about author's age. In this paper we tested novel set of features for authors age based classification of very short texts. Used features, formerly known as text readability features, that are used by different readability formulas, as Gunning Fog, and others, proved to be suitable for automatic age detection procedure. Comparing different classification algorithms we found that Logistic Regression and Support Vector Machines created best models with our data and features, giving both over 90% classification accuracy.

While this study has generated encouraging results, it has some limitations. As different readability indexes measure how many years of education is needed to understand the text, we can not assume that peoples reading, or in our case writing, skills will continuously improve during the whole life. For most people, the writing skill level developed in high school will not improve further and therefore it is impossible to discriminate between 25 and 30 years old using only those features as we did in current study. But these readability features might be still very useful in discriminating between younger age groups, as for instance 7-9, 10-11, 12-13. The other possible utility of similar approach is to use it for predicting education level of an adult author.

In order to increase the reliability of results, future studies should also include a larger sample. The value of our work is to present suitability of a simple feature set for age based classification of short texts. And we anticipate a more systematic and in-depth study in the near future.

[2] http://www.tlu.ee/~pentel/age_detector/source_code.txt

6. REFERENCES

[1] Burrows, J. 2007. All the way through: testing for authorship in different frequency strata. Literary and Linguistic Computing. 22, 1, pp. 27–47. Oxford University Press.

[2] Sanderson, C., and Guenter, S. 2007. Short text authorship attribution via sequence kernels, Markov chains and author unmasking: an investigation. EMNLP'06. Association for Computational Linguistics. pp. 482–491. Stroudsburg, PA, USA.

[3] Rao, D. et al. 2010. Classifying latent user attributes in twitter, SMUC '10 Proceedings of the 2nd international workshop on Search and mining user-generated contents. pp. 37-44.

[4] Gunning, R. 1952. The Technique of Clear Writing. New York: McGraw–Hill

[5] Pentel, A. 2014. A Comparison of Different Feature Sets for Age-Based Classification of Short Texts. Technical report. Tallinn University, Estonia. www.tlu.ee/~pentel/age_detector/Pentel_AgeDetection2b.pdf

[6] Luyckx, K. and Daelemans, W. 2011. The Effect of Author Set Size and Data Size in Authorship Attribution. Literary and Linguistic Computing, Vol-26, 1.

[7] Tam, J., Martell, C. H. 2009. Age Detection in Chat. International Conference on Semantic Computing.

[8] Lin, J. 2007. Automatic Author profiling of online chat logs. Postgraduate Thesis.

[9] Peersman, C. et al. 2011. Predicting Age and Gender in Online Social Networks. SMUC '11 Proceedings of the 3rd international workshop on Search and mining user-generated contents, pp 37-44, ACM New York, USA.

[10] Santohs, K. et al. 2013. Author Profiling: Predicting Age and Gender from Blogs. CEUR Workshop Proceedings, Vol-1179.

[11] Santosh, K. et al. 2014. Exploiting Wikipedia Categorization for Predicting Age and Gender of Blog Authors. UMAP Workshops 2014.

[12] Marquart, J. et al. 2014. Age and Gender Identification in Social Media. CEUR Workshop Proceedings, Vol-1180.

[13] Nguyen, D. et al. 2011. Age Prediction from Text using Linear Regression. LaTeCH '11 Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities. pp 115-123, Association for Computational Linguistics Stroudsburg, PA, USA.

[14] Wu, X. et al. 2008. Top 10 algorithms in data mining. Knowledge and Information Systems. vol 14, 1–37. Springer.

[15] Mihaescu, M. C. 2013. Applied Intelligent Data Analysis: Algorithms for Information Retrieval and Educational Data Mining, pp. 64-111. Zip publishing, Columbus, Ohio.

[16] Weka. Weka 3: Data Mining Software in Java. Machine Learning Group at the University of Waikato. http://www.cs.waikato.ac.nz/ml/weka/

Mining an Online Judge System to Support Introductory Computer Programming Teaching

Rodrigo Elias Francisco
Instituto Federal Goiano
Campus Morrinhos
Morrinhos – GO – Brazil
+55 (64) 34137900
rodrigo.francisco@ifgoiano.edu.br

Ana Paula Ambrosio
Instituto de Informatica
Universidade Federal de Goiás
Goiânia – GO – Brazil
+55 (62) 35211181
apaula@inf.ufg.br

ABSTRACT

Computer programming is an activity which requires a set of cognitive processes that naturally develop through practice, writing algorithmic solutions. Students learn a lot from their mistakes, but for this they need feedback on their workouts. Marking students' work outs is very time consuming, which often limits a teacher's capacity to offer close guidance individually. The PROBOCA project aims to build a tool, based on the BOCA online judge, suited for the purpose of learning computer programming by practice. In addition to a problem database organized by theme and difficulty, the system provides functionalities to support the teacher in the classroom. One of the main endeavors is to develop a procedure for estimating the degree of difficulty of a certain problem. This "nominal" parameter may then be compared to the difficulty level as perceived by each student. The result is a valuable indicator of those students that are experiencing challenges. This paper presents the preliminary specification of PROBOCA´s architecture and functional requirements along with its current state of development.

Keywords

Online Judge; Computer Programming Education.

1. INTRODUCTION

The learning process of Computer Programming (CP) usually involves practicing the resolution to many problems. Students write code to implement algorithms that meet exercise requirements and teachers should review those codes and present their feedback to the students. As this is a time consuming task, some teachers are installing and using online judge systems as automatic reviewing and scoring tools.

Online judge refers to software tools originally designed for programming competitions. They usually run on a server, and contestants access it online. Basically, its role is to present the contestant teams with a list of problems, to which they should respond by uploading program codes that satisfy the criteria of each problem. The tool then evaluates the answer code by using a set of predefined inputs and comparing the program results to predefined outputs. If the output of the answer code exactly matches the expected output for the corresponding input, the answer code is considered correct. Otherwise it is considered incorrect. No indication of where the program went wrong is given. Although helpful as a teaching assistant, these tools were not designed for use in a classroom and therefore lack some features that are important for academic management.

The BOCA software [1] is an online judge system used in programming marathons in Brazil. It is freely available for institutions to download and install. This particular system allows teachers to register problems and to track their students' work. However this system is neither easy to handle nor has an exercise database (DB), needed to facilitate the generation of problem lists.

This project proposes to extend the BOCA online judge to make it more suitable for use in introductory programming teaching. The resulting system, called PROBOCA, complements the online judge functionality with features that improve its ease-of-use, enhancing teacher productivity and course effectiveness.

One of the introduced features provides automatic classification of problem difficulty, based on submitted solutions and students' evaluation of degree of difficulty encountered in solving a given problem. This will aid teachers while composing the exercise lists, allowing them to better gauge the list complexity. It also lets students organize the order of problems to solve, tackling the easier problems first before turning to more complex ones.

Another additional feature is the production of student reports based on the submitted solutions and the students' behavior while accessing the tool. Student evaluation of program difficulty, number of submissions for the same exercise, time between submissions, order of submissions, among other information gathered by the tool, reveal information about student behavior and his ease in solving the proposed problems.

2. EXERCISES ON PROGRAMMING EDUCATION

Aiming at automatic correction of program code and student monitoring and evaluation during the programming process, several initiatives have been developed.

Within this line of research, the study of Chaves et al [2] aims to explore resources of online judges looking to integrate them into the Moodle system. The goal is to provide a Virtual Learning Environment (VLE) with automatic evaluation feature, allowing the teacher to monitor students´ problem solving. The authors defined an architecture containing a module for integration with online judges (MOJO). The Integration Module integrated the VLE structure to URI Online Judge [3] and SPOJ Brazil [4].

In [5], the authors have developed a prototype intended to be an educational online judge. They argue that online judge tools are suited to competition and have few educational features. They also criticize the way online judges provide feedback to students only indicating whether the answer is right/wrong or if there were errors at compilation/runtime. Their project, called JOnline, aims to add the following didactic features: Presenting tips in

Portuguese to fix compilation errors found in the submitted source code; presenting the test cases that generate erroneous results; organization of the problems by topic; difficulty level deduced from a poll conducted by the system and a resource for collaborative programming that allows two students to co-write one common code.

Automatic generation of exercise lists based on user defined criteria requires the classification of problems according to these criteria. In [6], the authors conclude that there is a strong relationship between the difficulty of a problem and the total number of lines of code and amount of flow control in the program (IF, WHILE, FOR ...). New students have difficulty in reading and interpreting problem statements. This problem has been related to the difficulty in dealing with abstraction [9].

3. BOCA

Designed for use in programming marathons, the BOCA online judge system has vocabulary and requirements contextualized by Competition and Teams. The main interest in this system is that it was designed to enable its users, the competitors, to interact with a set of problems. Figure 1 shows a use case diagram with the different functions provided by BOCA. Its software allows registration of problems for a given competition. That is done by submitting a PDF file stating the problem to be tackled and the files containing the input and output data sets for the corresponding test cases. The registered problems are associated with a single competition. If needed, to reuse the problems for another competition, the files must be inserted again. BOCA does not include a database to store the problems.

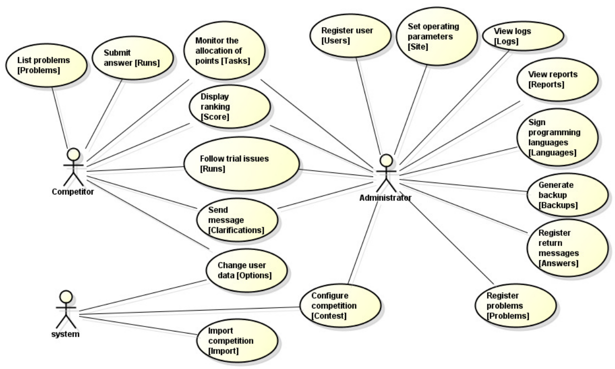

List problems [Problems]
Submit answer [Runs]
Monitor the allocation of points [Tasks]
Register user [Users]
Set operating parameters [Site]
View logs [Logs]
View reports [Reports]
Display ranking [Score]
Sign programming languages [Languages]
Follow trial issues [Runs]
Competitor
Administrator
Generate backup [Backups]
Send message [Clarifications]
Register return messages [Answers]
Change user data [Options]
system
Configure competition [Contest]
Register problems [Problems]
Import competition [Import]

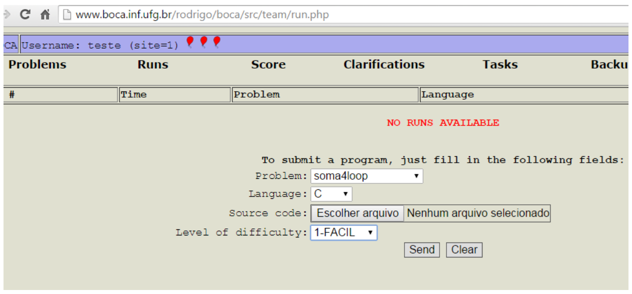
Figure 1. BOCA´s use case diagram

In the system, colored balloons represent the points gained by the competitors. This feature is part of the programming marathons context where correctly answering a problem yields a balloon to its team. Besides allowing teams to monitor their submissions and the related results, BOCA also provides a screen displaying the teams ranking including the team's overall score, and a list of the solved exercises using balloons to identify those successfully done. It also produces information on students' interaction with the problems containing the time spent and the number of resolution attempts.

BOCA has been used in CP introductory classes at our institute for some years, with good results. Using this system enhances the practice of problem solving by providing automatic feedback to students. We have observed that the number of problems solved by students using BOCA is significantly higher when compared to traditional exercise lists. The burden on the teacher is significantly reduced and the students get feedback. Teachers are then able to focus on helping the students that fail to solve the problems even after several attempts. This strategy allows students to tackle a larger number of exercises, thus increasing their potential to master cognitive skills required for programming. However, in spite of these advantages, some drawbacks were revealed.

Looking at the process of reviewing students' answers two issues were identified. First, the system assesses a submitted program by comparing its output, as generated by the student's code in response to a given input data set, to the registered expected output. For the two outcomes to be identical, exact formatting of the program's output is required. At the beginning students incurred in many errors just for using a slightly different format. This situation marks the problem's result as wrong whereas the program's logic maybe right, causing frustration among students as the system does not point out where the error is. Students, however, managed to adapt and began paying more attention to output formatting. Second, there is no cross-answers plagiarism identification, in other words no control over cheats. It was observed that some students simply copied their colleagues' program and submitted them as their own, thus considered to have solved the problem without any real comprehension of the solution.

Figure 2 shows the BOCA code submission screen. It was modified to include the "level of difficulty" selection where the students evaluate how hard it was to solve that problem they are submitting.



```
Username: teste (site=1)
```

#	Time	Problem	Language

```
NO RUNS AVAILABLE

To submit a program, just fill in the following fields:
              Problem: soma4loop
             Language: C
          Source code: Nenhum arquivo selecionado
  Level of difficulty: 1-FACIL
```

Figure 2. BOCA's submission functionality answers

Regarding the class management functionality, it was noted that each installed instance of BOCA supports one active competition at a time, meaning a single class per instance. Thus, if a teacher wants to have several competitions running at the same time i.e. different groups doing the same or different sets of exercises, he must install multiple instances of BOCA. Each instance is independent. So even if two competitions are composed of the same problems (e.g. for different classes), these problems must be separately registered for each instance. As each competition is independent, the system does not group the results of the different competitions. This feature would be interesting in case of multiple exercise lists delivered to the same class. It should also be noted that the system is not trivial to install and manage, which ends up discouraging the teachers from adopting it. On top of that, the teacher needs to add each student to the system, which proves to be quite tedious, especially for large classes. To overcome this drawback, teachers have implemented programs that automatically generate registration ids and corresponding passwords based on student university registration number. As BOCA does not allow password modification, students often know other students password as they are generated following certain patterns. This facilitates copying other students' solutions.

This shows that, although BOCA has several advantages, there are problems that hinder the use of the system in the classroom. To solve this, beyond what has already been presented, we aim to tackle the following requirements: automatic generation of lists of problems based on subjects and level of difficulty, measurement of the student experience with problem solving by subject, and rank problems by levels of difficulty. We consider the last item is important for didactic purposes, not finding a technique or algorithm to use, this is the focus of this work.

4. PROBOCA

From an architectural viewpoint, the proposed system builds upon BOCA that is considered an internal component and offers its structure, PHP source code and PostgreSQL DB to be reused.

In order to avoid further complication of the development process, the "Competitions and Teams" context is kept, with some adaptations. The terms "*Competition*" and "*Team*" are used to loosely indicate *exercise list* and *student* respectively. BOCA´s user interface allows for such extrapolation which is already in practice by the teachers who use this system in their classroom.

Adapting BOCA to support CP teaching brought the need to introduce new functional requirements, as presented in the use case diagram in Figure 3.

PROBOCA required some changes to the BOCA DB structure. It was necessary to modify the internal tables in order to link the stored problems to given course's syllabus and certain difficulty levels as well as to include more detailed data about the students. Furthermore, the original competition-based structure will be altered to adapt it to the concept of multiple exercise lists.

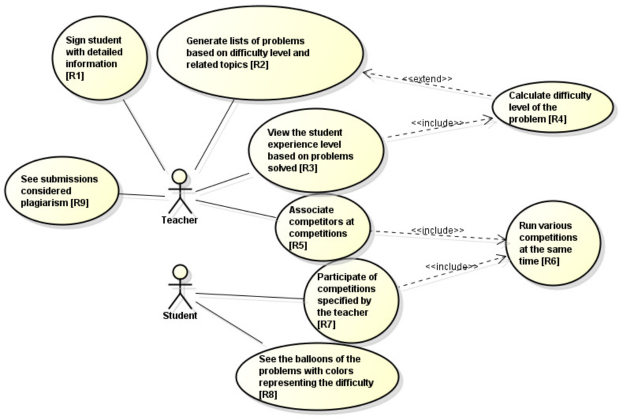
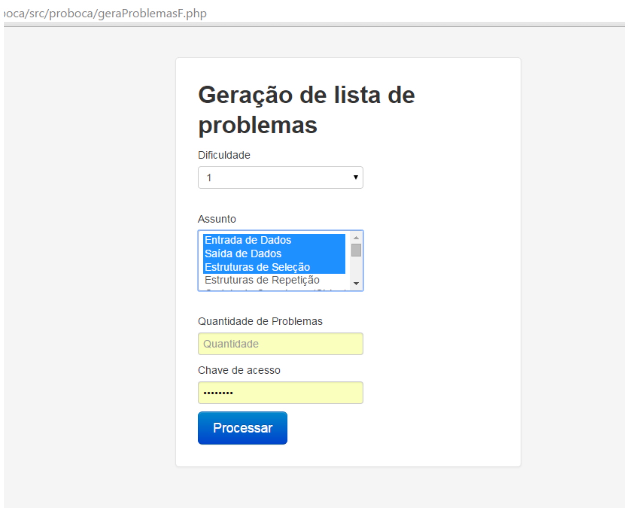

Sign student with detailed information [R1]
Generate lists of problems based on difficulty level and related topics [R2]
<<extend>>
Calculate difficulty level of the problem [R4]
<<include>>
View the student experience level based on problems solved [R3]
See submissions considered plagiarism [R9]
Teacher
Associate competitors at competitions [R5]
<<include>>
Run various competitions at the same time [R6]
<<include>>
Student
Participate of competitions specified by the teacher [R7]
See the balloons of the problems with colors representing the difficulty [R8]

Figure 3. Use Case Diagram with New Requirements

Following requirement R1, student registration is now done by the student himself using a token which is handed out by the teacher. Besides saving a password, other information is collected during registration that permits class wide statistical analysis.

R2 was intended to simplify the task of competition generation, which is performed by the teacher. To achieve this goal, a database of problems was implemented. Currently, to generate a competition, as shown in figure 4, the user indicates three parameters, namely a difficulty level (1-easy; 2-medium; 3-difficult); the desired component syllabus´ elements for the given problem set and finally the quantity of problems to compose the list. The system then analyzes which problems best fit the user-defined parameters and generates a competition list, from among the problems available in its DB. The difficulty level for each problem is estimated automatically by the system based on data obtained from solutions submitted by students and other parameters as described in section 6.


oca/src/proboca/geraProblemasF.php

Geração de lista de problemas

Dificuldade
1

Assunto
Entrada de Dados
Saída de Dados
Estruturas de Seleção
Estruturas de Repetição

Quantidade de Problemas
Quantidade

Chave de acesso
....

Figure 4. Generate problem lists\competitions

R3's purpose is to provide teachers with a report of every student´s experience level based on his interaction with the system. Using the collected data, it should be possible to present information about a student's experience regarding the syllabus elements addressed in each problem, thus allowing the teacher to detect where students show more difficulty. This requirement depends on R4, which measures the difficulty level of a given problem. A simple mechanism to measure the degree of difficulty of a given problem was developed and tested.

For R3 to be reliable, it is necessary that a plagiarism identification strategy (R9) be also implemented in the system.

R5 is aimed at associating a student to multiple competitions without the need to register each time. Thus, the data from different competitions can be associated to the student, allowing the integration of results obtained for the different lists presented to a given class. This requirement has a dependency upon R6, which aims to adapt BOCA to judge problems from several competitions. R7 is also related to R5, with the goal of allowing a student to participate in several competitions - analog to answering several problem lists.

R8 aims to display the problems' balloons, color coded to indicate the problem's difficulty level. Currently, balloon colors are registered by the system administrator along with the problems and have no relation to difficulty or content, unless the classification and corresponding color attribution is done by the teacher when uploading the problems.

5. IMPLEMENTATION

The functionalities of detailing the student information and providing automatic generation of exercise lists are already implemented and tested. Requirement R4, estimating the difficulty level of a given problem, is currently in progress.

Within this requirement, the first executed task was to create 74 exercises using BOCA´s problem-registration functionality. These exercises cover the syllabus of the "Introduction to programming" course (CS1) and range from debutant programming instructions (Hello World style) up to coding with matrixes. The inserted problems populated an initial database to be used for testing, but additional problems are expected to be easily included. An estimated level of difficulty for each problem was supplied by the teacher inserting the problem in the database.

To submit an answer, the student uploads one code file per problem solved. Along with the file upload, the student is asked to evaluate the problem´s difficulty, by indicating one of three choices: Easy, Medium or Difficult, based on his experience when solving the problem (Figure 4).

Success was obtained in computing several parameters that are needed to calculate problem difficulty. They include different measures, such as counting the number of repetition and selection structures used in the solution and the number of topics involved in the problem. Problem topics were defined using the introductory programming course syllabus. They include, input/output; attribution; selection; repetition; vectors; strings; matrices and functions. In this list, topics appear in the order they are taught and therefore in increasing level of complexity for novel students. Since programming is incremental, more complex topics usually base upon lesser complex topics. Several tests have been conducted to define and validate the mathematical function that will calculate the difficulty of the problem, but this is still an open issue.

Based on the calculated difficulty and the topics involved in the problem, colored balloons are associated to the problem. Different colors represent different topics, and within each topic, the level of difficulty is represented by the intensity of the color. For example, blue represents problems whose most complex topic is selection. Light blue balloons are associated to easy problems using selection. Medium blue balloons are associated to medium difficulty problems and dark blue balloons are associated to problems that demand more from students.

The task of providing teachers with a report on student achievements (R3) has not been tackled yet. Ideas in this sense include: (1) comparing student's perceived problem difficulty and mean problem difficulty. If students perceive problems as harder or easier than the mean this could indicate that they are at a lesser or higher level of programming competency; (2) comparing successive submission of the same problem. This may show if students adopt a trial-and-error approach; (3) mean time taken to solve problems; (4) mean number of submission per problem; (5) score when compared to the other students; among others.

6. Estimating Problem Difficulty

In a sense, "difficulty" expresses the lack of ability, or amount of skill/effort needed to accomplish something. Obviously, the perception of difficulty is an individual matter that is related to many aspects, including the time at which the question was posed. Nonetheless, this work investigates the possibility of characterizing a programming problem in a way such that calculated parameters may correlate to a determined "Difficulty-level" as expressed by students.

In their work [6], Alvarez and Scott present the program control flow and number of lines of code as variables that correlate to the difficulty of a problem. The experiments undertaken in the current study corroborate this information.

Also, other works [7, 8] show the need to deal with abstraction to solve problems. A highly abstract problem is one that implies a greater level of generalization. Thus, a problem that involves many different topics can become more abstract and hence more difficult. It is fairly safe to assume that in order to measure the difficulty of a problem, it is necessary to make a detailed analysis of the topics that are addressed within the problem. That proposition helps in formulating the hypothesis below.

6.1 Problem Difficulty Mechanism

On one hand, the survey of the student´s opinion of the submitted problem's difficulty provided the first estimate. For each of the 74 registered exercises, this information was stored, along with the respective answer in the DB. Although information was collected for all students, statistics were calculated only considering those that completed at least 95% of the exercises (70 out of 74). This excluded students that dropped out the course in order to prevent their partial answers from skewing the results. After filtering, the mean difficulty "Mean_Dif" was then calculated for each exercise based on the answers of the resulting 62 students. "Median_Dif" and "Mode_Dif" were also calculated.

In addition, C source code was written to correctly workout each of the registered problems. A PHP program was also developed in order to analyze the programs´ internal structures and to extract some measures from each program. The measures include counting:

- Lines of code, represented by variable N_LC;
- Repetition structures used in the solution, N_RP;
- Selection structures, N_SL;
- Edges in a graph that represents the algorithm, N_EDG;
- Edges in a graph that represent repetition structures in the algorithm, N_EDG_RP;
- Height of a tree, with each sub-block of internally nested code representing a node, MX_HT and
- Number of topics involved in the problem, N_TPC. This last count was obtained manually.

To verify if a combination of variables obtained better results, the following formulae were also tested against the Mean_Dif variable to verify their correlation. Table 2 shows the results. Mean_Dif correlated positively with all measured variables, being the best correlation associated to f4, r = .76, p = .000, that improves on the individual correlations with N_TPC (number of topics) and MX_HT (maximum tree height).

Table 1. Correlation between measured variables and student evaluation

		Median_ Dif	Mean _Dif	Mode_ Dif
N_RP	*Pearson Correlation*	.49	.52	.44
(Repetitions)	*Sig. (2-tailed)*	.000	.000	.000
	N	74	74	74
N_SL	*Pearson Correlation*	.16	.34	.19
(Selections)	*Sig. (2-tailed)*	.171	.003	.108
	N	74	74	74
N_LC	*Pearson Correlation*	.44	.62	.45
(Lines of	*Sig. (2-tailed)*	.000	.000	.000
Code)	*N*	74	74	74
N_TPC	*Pearson Correlation*	.57	**.69**	.59
(Topics)	*Sig. (2-tailed)*	.000	.000	.000
	N	74	74	74
N_EDG	*Pearson Correlation*	.42	.58	.41
(Edges)	*Sig. (2-tailed)*	.000	.000	.000
	N	74	74	74
MX_HT	*Pearson Correlation*	.52	**.67**	.56
(Tree Height)	*Sig. (2-tailed)*	.000	.000	.000
	N	74	74	74
N_EDG_RP	*Pearson Correlation*	.53	.60	.51
(Rep. Edges)	*Sig. (2-tailed)*	.000	.000	.000
	N	74	74	74

$$f1 = \frac{N\_RP+N\_SL+N\_LC+N\_TPC+N\_EDG+MX\_HT+N\_EDG\_RP}{7}$$

$$f2 = N\_EDG \cdot 0.3 + MX\_HT \cdot 1.4 + N\_EDG\_RP \cdot 1.4 + N\_LC \cdot 0.2$$

$$f3 = \frac{N\_TPC+MX\_HT+N\_LC}{3} + \frac{N\_EDG\_RP+N\_EDG+N\_RP + N\_SL}{8}$$

$$f4 = \frac{N\_TPC + MX\_HT}{2}$$

Table 2 shows the results. Mean_Dif correlated positively with all measured variables, being the best correlation associated to f4, r = .76, p= .000, that improves on the individual correlations with N_TPC and MX_HT.

Table 2. Correlation of student perceived difficulty and developed formulae

		Mean_Dif
f1	*Pearson Correlation*	.66
	Sig. (2-tailed)	.000
	N	74
f2	*Pearson Correlation*	.68
	Sig. (2-tailed)	.000
	N	74
f3	*Pearson Correlation*	.67
	Sig. (2-tailed)	.000
	N	74
f 4	*Pearson Correlation*	**.76**
	Sig. (2-tailed)	.000
	N	74

We also verified correlations between teacher evaluations of the problems' difficulty. For this we asked five teachers to read each problem and attribute a level of difficulty in the range 1-5. Since we had only five evaluations we chose to work with the median difficulty obtained (Median_Prof_Dif).

Correlating this variable to the measured variables we obtained the results presented in table 3. Best correlations were obtained with N_RP and N_EDG_RP, *r* = .62, *p* = .000, both related to the number of repetition structures found in the solution codes.

Table 3. Correlation between measured variables and teacher evaluation

		Median_Prof_Dif
N_RP	*Pearson Correlation*	**.62**
(Repetitions)	*Sig. (2-tailed)*	.000
	N	74
N_SL	*Pearson Correlation*	.20
(Selections)	*Sig. (2-tailed)*	.093
	N	74
N_LC	*Pearson Correlation*	.56
(Lines of Code)	*Sig. (2-tailed)*	.000
	N	74
N_TPC	*Pearson Correlation*	.50
(Topics)	*Sig. (2-tailed)*	.000
	N	74
N_EDG	*Pearson Correlation*	.53
(Edges)	*Sig. (2-tailed)*	.000
	N	74
MX_HT	*Pearson Correlation*	.50
(Tree Height)	*Sig. (2-tailed)*	.000
	N	74
N_EDG_RP	*Pearson Correlation*	**.62**
(Rep. Edges)	*Sig. (2-tailed)*	.000
	N	74

Table 4 correlates the teacher defined difficulty with the proposed formulae defined above. Positive correlation was found with all formulae, being the best correlation associated to f2, *r* = .63, *p*=.000. Furthermore, a positive correlation was found between the teacher defined difficulty and mean student perceived difficulty, *r* = .69, *p* = .000.

Table 4. Correlation of teacher defined difficulty and developed formulae

		Mean_Dif
f1	*Pearson Correlation*	.59
	Sig. (2-tailed)	.000
	N	74
f2	*Pearson Correlation*	**.63**
	Sig. (2-tailed)	.000
	N	74
f3	*r**	.59
	Pearson Correlation	.000
	Sig. (2-tailed)	74
	N	
f4	*Pearson Correlation*	.55
	Sig. (2-tailed)	.000
	N	74

Although student perceived difficulty and teacher defined difficulty are correlated, there are differences that can be verified by the correlations found with the measured variables and proposed formulae. This could be explained by the fact that students evaluate difficulty based on the knowledge they have when developing the solution. As they do not know what comes ahead, they cannot base their evaluation on the overall knowledge of programming. Teachers on the other hand, have this overall view of the domain and evaluate difficulty accordingly. It must be observed that the teachers that did the evaluation are new to teaching and have not yet acquired a more critical understanding of the difficulties students encounter when learning to program. After developing algorithmic reasoning, people tend to forget how they thought before, and find that many concepts are obvious when in fact, for beginners, they are not.

7. CONCLUSION

PROBOCA is a system under development whose goal is to adapt the BOCA software for use in teaching programming. While BOCA itself was developed for programming marathons, it is already in use, as a support tool, in programming introductory courses. As a learning aid, BOCA has several limitations, especially relative to class administration and student monitoring. In addition, as a system specifically developed for competitions, it lacks mechanisms that facilitate the creation of exercise lists, such as a question bank, and analysis of student performance.

PROBOCA supports the persistence of problems registered by teachers in the database and provides greater access to students' related information. Unlike other "Online Judge" systems that are available exclusively online and are managed by their creators, PROBOCA can be downloaded and installed by the teacher, giving the teacher control over the problems stored in the database. Since teachers are responsible for introducing the problems, this solution has the additional advantage that it is language free, i.e., it is not limited to teachers and students that speak the language in which the problem specifications were written as is the case of online systems administered by third parties.

In addition to implementing an environment that facilitates the use of BOCA in teaching programming, PROBOCA also aims to provide teachers and students with information that will help students in their learning process. One of the important features of PROBOCA is an automatic evaluation of problem difficulty. This gives students direction in the path to follow when choosing which exercises to solve first, allowing them to solve easier exercises before more complex ones, diminishing student frustration at not solving problems. It also allows teachers to better gauge the level of difficulty of exercise lists. As shown by the collected data, teacher and student evaluation regarding problem difficulty do not match, and this may lead to distortions when teaching programming. Future work could include developing a system that will automatically suggest problems to students based on their performance and calculated problem difficulty.

This work shows the approach being taken to calculate the difficulty of the problems. This approach differs from others by treating the algorithms submitted by students in the form of graphs and trees to identify properties that could be correlated with the difficulty of problems. For this, data mining using correlations was undertaken.

Part of the system has already been implemented, and has been successfully used in the classroom. The success of these tasks shows the feasibility of the project and encourages further work.

REFERENCES

[1] BOCA. *BOCA Online Contest Administrator.* Available in <http://www.ime. usp.br/~cassio/boca/> Access on March 20th, 2015

[2] Chaves, J. O. et al. *Uma Ferramenta de Auxílio ao Processo de Ensino Aprendizagem para Disciplinas de Programação de Computadores (2013).* TISE 2013

[3] URI. *URI Online Judge.* Available in <http:// www.urionlinejudge.com.br/> Access on May 20th, 2014.

[4] SPOJ. *SPOJ Brasil.* Available in <http://br.spoj.com/embed/info/> Access on May 20th, 2014.

[5] Santos, J. C. S., Ribeiro, A. R. L. *JOnline - proposta preliminar de um juiz online didático para o ensino de programação (2007).* SBIE 2011.

[6] Alvarez, A. and Scott, T. A. (2010). *Using student surveys in determining the difficulty of programming assignments.* Journal of Computing Sciences in Colleges, 26(2):157–163.

[7] Gomes, A. et al. (2008). *Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. Revista Portuguesa de Pedagogia.* 42(2).

[8] Mendonça, A. et al. *Difficulties in solving ill-defined problems: A case study with introductory computer programming students.* In Frontiers in Education Conference, 2009. FIE'09. 39th IEEE, pages 1–6. IEEE.

[9] Mendonça, A., de Oliveira, C., Guerrero, D., and Costa, E. (2009). *Difficulties in solving ill-defined problems: A case study with introductory computer programming students.* In Frontiers in Education Conference, 2009. FIE'09. 39th IEEE, pages 1–6. IEEE.