April 2006

Knowledge Engineering for Intelligent Tutoring

Kevin R. Kardian
Worcester Polytechnic Institute

Knowledge Engineering for Intelligent Tutoring Systems: Assessing Semi-Automatic Skill Encoding Methods

A Major Qualifying Project Report:

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Kevin R. Kardian

Date: April 27, 2006

Approved:

Professor Neil T. Heffernan

Abstract

Building a mapping between items and their related knowledge components, while difficult and time consuming, is central to the task of developing intelligent tutoring systems (ITS). Improving performance on this task by creating a semi-automatic skill encoder would facilitate ITS development. The goal of this project is to explore text classification techniques to reduce the time required to correctly tag items. This work has received favorable peer-review and was accepted for publication at the 8th Annual Intelligent Tutoring System Conference.

Acknowledgements

This research was made possible by the US Dept of Education, Institute of Education Science, “Effective Mathematics Education Research” program grant #R305K03140, the Office of Naval Research grant #N00014-03-1-0221, NSF CAREER award to Neil Heffernan, and the Spencer Foundation. All the opinions in this article are those of the authors, and not those of any of the funders.

This work would not have been possible without the assistance of the 2004-2005 WPI/CMU Assistment Team including Mingyu Feng, Andrea Knight, Ken Koedigner at CMU, Abraao Lourenco, Michael Macasek, Goss Nuzzo-Jones, Kai Rasmussen, Leena Razzaq, Steven Ritter at Carnegie Learning, Carolyn Rose at CMU, Terrence Turner, Ruta Upalekar, and Jason Walonoski.

Table of Contents

Abstract ii

Acknowledgements iii

Table of Contents iv

1 Introduction 1

2 Related Background 3

2.1 Machine Learning 3

2.1.1 Text Classification 3

2.2 Intelligent Tutoring Systems 7

2.2.1 The Assistment System 8

2.2.1.1 The Assistment Builder 10

2.2.1.2 Transfer Models and Knowledge Components 12

2.3 Previous Study of Skill Encoding 16

2.4 Applications Related to Skill Encoding 17

3 Design Decisions 18

3.1 Mallet Text Classification 18

3.2 Data Set 19

4 Methodology 21

4.1 Implementation 21

4.2 Experiment 1: Direct Classification of Question Texts into Skills 22

4.3 Experiment 2: Using Hierarchical Classification to Improve Performance 23

5 Results and Analysis 25

5.1 Analysis of Experiment 1 25

5.1.1 Qualitative Analysis of Individual Skills 26

5.2 Analysis of Experiment 2 29

6 Conclusions and Future Work 31

Appendix A1 32

Appendix A2 35

References 58

1 Introduction

Research in the area of artificial intelligence as it applies to intelligent tutoring systems is becoming increasingly prevalent. The US Dept of Education funded the Assistment project to build cognitively valid diagnostic assessment systems, which tutor while they assess. Razzaq et al [1] report on the fact that students do indeed learn from computers and that intelligent tutoring systems can assess students' performance within reasonable accuracy.

One of the more difficult problems in creating intelligent tutoring systems involves knowledge engineering for a given domain. In order to accurately gauge performance, thousands of questions must be tagged with a fine grained mapping of knowledge components (KCs). A KC represents a set of concepts, skills, or strategies needed to solve items within a domain; a complete set of knowledge components is referred to as a transfer model. Unfortunately, building these matrices can be difficult and time consuming; this process may be made easier through computer assistance. It is evident that reducing the amount of work required in building these matrices would greatly benefit those who develop subject matter for intelligent tutoring systems.

Inspired by Rosé et al [2], the purpose of this paper investigate the possibility that the computer might be able to assist the author by suggesting what KCs can be paired with a given question by looking at the words in the question. Rosé reported that "even in cases where the predictions cannot be made with an adequate level of reliability, there are advantages to starting with automatic predictions and making corrections, in terms of reliability, validity, and speed of coding." Having this assistance might also help in maintaining a complete mapping of items to KCs, as new items need to be added and coded to the system by users who might not be the original coders. Furthermore, such assistance might lead to more accurate matrices, as the system may suggest a KC that individual users might have overlooked for a given item.

The goal of this project is to attempt to further investigate semi-automatic skill coding to an end of improving the time required to tag an item with one or more skills from a very large number of possible skills. This paper does not attempt to do any empirical analysis to measure coding time, and instead is first investigating the idea applying Rosé et al. idea to our dataset. Specifically, this paper explores the accuracy of calculating several of the most likely skills, instead of only one. Furthermore, the accuracy associated with each skill individually is discussed in an attempt to gain a better understanding of what makes an item more or less difficult to classify. The worth associated with imposing a hierarchical model, beginning with a substantially less specific skill set as a basis for a more specific skill classification, is also investigated.

The next chapter outlines the relevant background information that is associated with this project. Chapter 3 deals with the important design decisions that were considered before implementation. Chapter 4 outlines the methodology that was followed for the purposes of this project. Chapter 5 discusses the results of each of the experiments, and analyzes those results. Finally, the paper concludes with some comments and a description of further advances that can be made with this research in chapter 6.

2 Related Background

2.1 Machine Learning

Machine learning is the use of artificial intelligence algorithms to understand a set of information and use it to solve a problem. Various algorithms exist, each with its own domain of problems that it is capable of solving to varying degrees of effectiveness. As the processing capabilities of computers increase, so do machine learning algorithms change and improve. As a result, the types and complexities of problems being solved in this manner are increasing. These algorithms play an integral role in the completion of the research described in this paper. [3]

Learning algorithms can be broken into two portions for the purposes of this study: training and testing. In training, parameters are learned from a data set and represented in a model that is useful for addressing the particular problem, in this case categorizing questions. Once training is complete, testing takes this model and applies it to a different data set in order to solve the problem. The testing and training data sets should contain no common examples between this; this ensures that a machine learning algorithm's efficacy is not simply limited to a small set of test data. [3][4] The specific machine learning techniques employed in this project are described below.

2.1.1 Text Classification

Text classification, defined by Dumais et al [4] as "the assignment of natural language texts to one or more predefined categories based on their content", has many practical applications. These applications include but are not limited to sorting of email or files, searching by category, and topic identification. This last application is the primary focus of this project. In many cases, including our own, documents would need to be classified by hand, which can prove very time consuming. As a result, there is a distinct motivation for automatic and semi-automatic text classification algorithms. Furthermore, these classification algorithms should not be rule-based, since defining these rules can prove difficult and time consuming. Guthrie et al [5] stress the importance of a mathematical model for text classification that both represents the problem and can easily be used to calculate relevant solutions.

An alternative to rule-base classifiers are classifiers created using machine-learning algorithms. While the number of features associated with a set of documents can be extremely large, these types of classifiers are much easier and faster to make and change and offer much more flexibility. Such machine learning techniques include but are not limited to regression models, nearest neighbor classifiers, support vector machines (SVM), neural networks, and Naive Bayes. Naive Bayes classifiers are the classification technique that we employed for this project, and will be discussed in detail below. These types of classifiers are gauged in terms of their classification speed and accuracy. [4]

Each of these machine-learned classifiers creates a vector of features, words in this case, and assigns each of them a confidence that such a feature fits into one of the relevant categories. These feature vectors are often reduced in size by a process called feature selection. A small number of features that have a high degree of mutual information with a given category are selected in order to both improve classification accuracy and reduce the amount of time required to create the classifier. Dumais et al [4] selected 50 features for each possible category in some of their algorithms, and 300 features in others. In their testing, SVM proved to be most effective, but Naive Bayes was outlined as "surprisingly effective". Naive Bayes was selected for its ease of implementation; the fact that it offers an accuracy that is reasonably high is enough for the comparisons that we report on in this paper. Generally speaking, the classification speed depends only on summing the confidences associated with each of the features in a given document and finding which of the categories has the highest related confidence.

2.1.2 Naive Bayes

The Naïve Bayes algorithm is a text classification algorithm that is probabilistic in nature; the confidence associated with the feature vector in a Naïve Bayes classifier is actually a probability distribution. This algorithm was so named because it makes the so called "Naïve Bayes assumption", which concludes that each of the features, or words, are actually independent of each other. While this assumption is often incorrect, it greatly simplifies the generation of a classifier and still achieves reasonably accurate results. This sort of data representation has led to the term "bag of words" to describe such a classifier, since the context in which the words are found is irrelevant. There are actually several methods by which this assumption can be employed in text classification, and McCallum and Nigam [6] compared two such methods.

In the first method, the feature vectors were binary in nature, which means that the number of times that a word appeared was irrelevant its classification. The other method used feature vectors that weighted the probabilities for classification based on the number of times a given word appeared in the document. Testing revealed that this "multinomial" Naïve Bayes model was almost universally more effective at text classification, especially when the vocabulary used is large in size. Based on these results, the multinomial model is what we will be discussing below.

In Naïve Bayes text classification, the probability of each category is estimated by finding the probability associated with all of the words in a given document. Because of the assumption of independence, this probability is simply the sum of each of the weighted probabilities from the feature vector. After the probabilities for each category are calculated, they can be ordered to provide a ranking for the best guesses about the correct category for a given document. [4][6]

The advantages associated with the multinomial are inherent in the nature of real-world problem solving. Actual problems are not simply limited to the use of a very few, highly correlated words. Instead, they have many instances of some very common words that are used between categories. Furthermore, documents that repeat a word several times can be assumed to be more heavily related to the corresponding category. For example, assume that each instance of numbers appearing in the data is replaced with a universal tag that simply identifies it as a number. It would not be out of the ordinary for most of the documents to have at least one number, but documents with many numbers may be heavily correlated with a particular category. Multinomial Naïve Bayes classifiers are capable of utilizing this information. Finally, an algorithm that uses binary feature vectors explicitly uses information gained from words that do *not* appear in a document; one may argue that multinomial approaches are at a disadvantage because they do not consider the absence of certain features. However, this argument is defunct since the absent features simply result in a count of zero, which implicitly reduces the weight of the corresponding probability to zero. [6]

2.2 Intelligent Tutoring Systems

The goal of Intelligent Tutoring Systems (ITS) is to attempt to provide automatic and dynamic assistance to students within a computer program. As the development of these systems progresses, new and innovative techniques are being employed. One such system, developed by Koedinger et al [7], outlines several useful techniques for ITS creation.

The Practical Algebra Tutor (PAT) was created and delivered to over 500 high school students in Pittsburgh, PA. Many themes were described as integral to the design and development process. Among them was a heavy emphasis on a close relationship between educators and system developers. Curriculum developers' expertise on the relevant subject matter was combined with cognitive psychology and artificial intelligence techniques in order to produce a system that would work well for its particular setting. Furthermore, PAT was created with design goals that matched the goals of the education system. For example, the curriculum at the time focused on applying algebra skills to real-world problems. As a result, PAT included functionality to control tables and graphs as a tangible representation of those problems. Additionally, classroom teachers were consulted to attempt to harness the same strategies that were actually employed in their classrooms.

PAT was created not only to aid the learning process, but also to monitor the progress of the students. When student made mistakes, the system would be aware of what the student may have been thinking and provide relevant hints. In addition, their performance on each problem would be noted so that this data could be later used to identify strengths and weaknesses. This dual-purpose tutoring system proved effective, and is discussed later in the context of the Assistment System.

In general, the PAT system was shown to effectively assist students' learning. Standardized test scores among those who used the system showed significant improvement compared to those who did not. Furthermore, teachers expressed great satisfaction with the system, stating that it allows them to use their time to provide more individual assistance. [7]

2.2.1 The Assistment System

The Assistment System [1] is a web-based Intelligent Tutoring System (ITS) for mathematics that was designed to help balance the time spent on teaching students and testing their abilities. The goal was to design an ITS that assisted while it assessed, hence the name. The Assistment System is capable of using machine learning algorithms to help understand the strengths and weaknesses of the individual students. It is also capable of providing help to the students as they attempt to work through the problems.

As with the PAT system, the teachers who would be using this system for their classroom were involved heavily in the design process. Initially, teachers were asked questions about specific math problems related to breaking the problem down. The goal of these interviews was to identify the steps required to help a student through a problem. Each question was then converted to an Assistment, which consisted of the original question and several scaffolding questions. One or more of the scaffolds, or sub-problems, were presented to the students if they were having difficulty with the problem. Additionally, the students were given the option to ask for hints throughout the process, and forced hints, called buggy messages, were issued based on specific incorrect responses.

The Assistment System logs any and all student activity, and keeps teachers informed through an online reporting system. These reports are updated dynamically, so teachers are able to view student progress even as the students are completing problems. These reports were designed to answer one or more of the following questions:

- Which items are my students finding difficult?
- Which items are my students doing worse on compared to the state average? Which students are 1) doing the best, 2) spending the most time, 3) asking for the most hints etc.?
- Which of the approximately 80 skills that we are tracking are students doing the best/worst on?
- What are the exact actions that a given student took?

This functionality was made possible by combining ITS design practices with an online database application development. In general, teachers are pleased with the result and the system is being constantly improved by continuing to work closely with those teachers. By spending time in classrooms while the Assistment System is being used, developers refine the Assistments to communicate the material in a more effective fashion and correct errors that cause confusion.

As will be discussed later, this system has shown the potential to accurately anticipate how a student will score on actual standardized tests. Furthermore, the Assistment System demonstrates a means to identify the effectiveness of various tutoring methods (i.e. different ways to scaffold a question). Through these means, the system is capable of improving, satisfying a design goal of teaching students more effectively. [1]

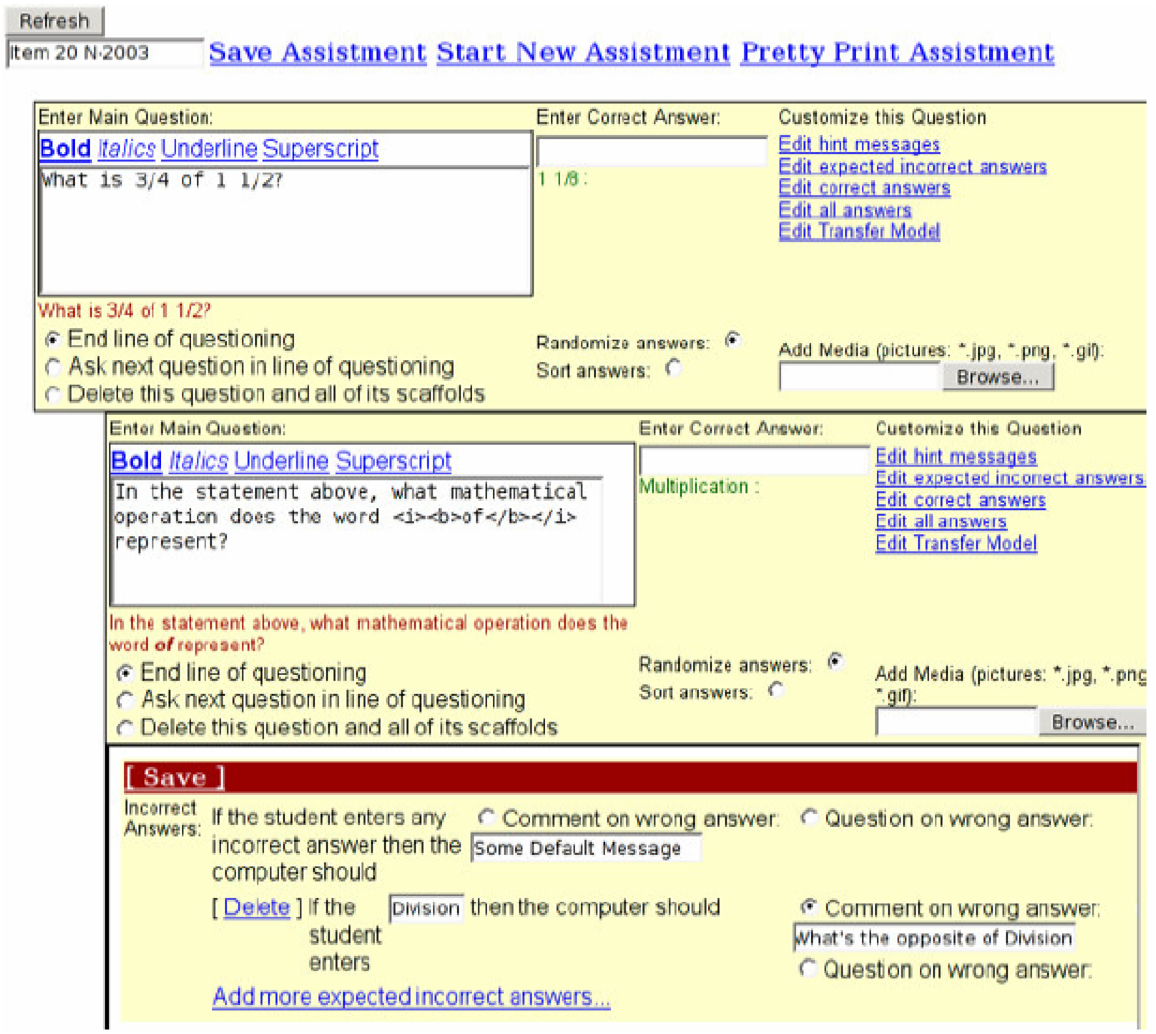
2.2.1.1 The Assistment Builder

The Assistment System employs a specialized builder as part of the system to aid in the "rapid development and deployment of Intelligent Tutoring Systems (ITS)". In general, ITS creation is prohibitively time-consuming and requires a range of skills, including programming ability in addition to experience with the relevant content. The goal of the Assistment builder is to allow teachers with no prior computer programming ability to create Assistments with little or no training. The builder adopts a system where each Assistment is a series of states in a graph, which are connected by a number of arcs. This architecture allows for feedback to a variety of answers based on expected actions on the parts of the students.

The builder, created for its simplicity and ease of use, represents each Assistment in XML (eXtensible Markup Language). Within this representation, each problem is made up of two primary components, an interface definition that controls the widgets that are displayed to the students, and a behavior definition that outlines how each state within the question relates to other states. When a student attempts to solve a given problem, his or her answers are compared to the behavior definition to determine what will be displayed next. The basic nature of the data representation is integral to the ability of the builder to facilitate Assistment creation. This architecture is not only useful for quickly creating and deploying new problems, but also for taking steps to correct errors in a timely and efficient manner.

Valuing speed of development over complexity, the Assistment builder provides a limited number of widgets for use by teachers when creating their interfaces. However, any given question can be branched to different subsequent branches based on the answer. Furthermore, both requested and forced hints can be added to any question. In this way the builder is able to achieve an effective level of complexity without increasing the difficulty to an unmanageable level. Like the Assistment System itself, the builder is run entirely through a web browser, which makes it still easier to use. New Assistments are propagated to a teacher's class instantly, and changes are updated as they are made.

When presented to a teacher, the builder starts with a blank skeleton to which they can add question text, images, and answers. Additionally, this skeleton allows for the addition of zero or more scaffolding questions that apply directly to the main question. The process of creating these scaffolds is exactly the same as creating the original questions. This initial screen is referred to as the *Main View*, and contains the minimum amount of information that is required for a question to be functional. Additional information can be added to an Assistment through other views, which include the *All Answer View*, *Correct Answer View*, *Incorrect Answer View*, *Hints View*, and *Transfer Model View*. Transfer models are of particular interest to this project, and will be discussed in more detail below. Any time after the fields in the *Main View* are populated, the Assistment can be saved and used immediately. At any later time, the builder can be revisited and the information in any of the views can be changed, once again resulting in immediate changes to the usable Assistment. Shown below is an example of the Assistment builder, in *Incorrect Answer View*, with the relevant fields populated.


Refresh

Item 20 N-2003 Save Assistment Start New Assistment Pretty Print Assistment

Enter Main Question: Bold Italics Underline Superscript

What is 3/4 of 1 1/2?

Enter Correct Answer: 1 1/8 :

Customize this Question: Edit hint messages, Edit expected incorrect answers, Edit correct answers, Edit all answers, Edit Transfer Model

What is 3/4 of 1 1/2?

[✓ End line of questioning] Ask next question in line of questioning; Delete this question and all of its scaffolds

Randomize answers: [✓] Sort answers:

Add Media (pictures: *.jpg, *.png, *.gif): Browse...

Enter Main Question: Bold Italics Underline Superscript

In the statement above, what mathematical operation does the word <i><b>of</b></i> represent?

Enter Correct Answer: Multiplication :

Customize this Question: Edit hint messages, Edit expected incorrect answers, Edit correct answers, Edit all answers, Edit Transfer Model

In the statement above, what mathematical operation does the word ***of*** represent?

[✓ End line of questioning] Ask next question in line of questioning; Delete this question and all of its scaffolds

Randomize answers: [✓] Sort answers:

Add Media (pictures: *.jpg, *.png, *.gif): Browse...

[Save]

Incorrect Answers: If the student enters any incorrect answer then the computer should: Comment on wrong answer: Some Default Message; Question on wrong answer:

[Delete] If the student enters Division then the computer should [✓ Comment on wrong answer:] What's the opposite of Division; Question on wrong answer:

Add more expected incorrect answers...

Figure 2.1: ***Incorrect Answer View*** **of the Assistment builder.**

The Assistment builder was created to improve ITS development speed at the cost of giving fewer options to the creators of these systems. However, this system has been instrumental in creating far more problems for intelligent tutoring than would have been created without it. [8]

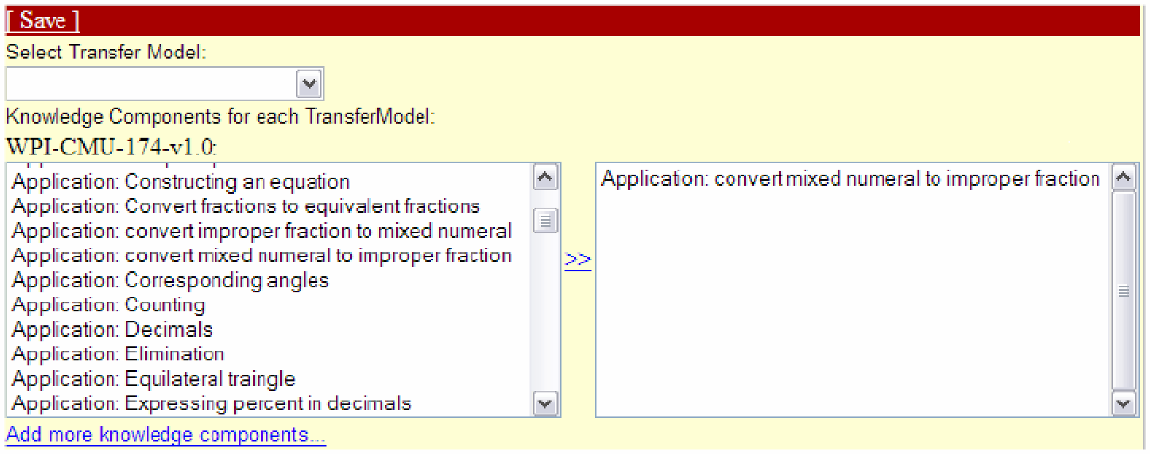
2.2.1.2 Transfer Models and Knowledge Components

Knowledge Components (KCs) are the tags to which individual questions correspond. Because the tags might represent concepts, skills, or strategies needed to solve problem, as opposed to simply procedural skills, the name Knowledge Components was assigned. These components are mapped to each question, and some questions may be tagged with more than one skill. In the psychometrics community, a complete mapping of a question set to its related KCs is referred to as a *Q-matrix* [9][10]. Croteau et al [11] refer to this mapping as a Transfer Model

Transfer Models are inherently developed independently from any individual tutor [8]. As a result, they are an unbiased representation of how different KCs correspond to the subject matter. In general, questions are associated with their related KCs at the time of creation, but in some cases tagging sessions must occur. In the Assistment Project [1], the team has made a focused effort to find more specific Transfer Models that provide a less general form of data. However, coding sessions have been shown to take approximately 48 person-hours to tag about 300 questions with their related KCs. This procedure, shown below in Figure 2.2, was done with paper cut-outs of all the items.

Figure 2.2: A Knowledge Component coding session.

When the session was over, there were 80-100 piles of items, and 20 hours of data-entry to build the Transfer Model. As a result, there is distinct motivation for some form of facilitated KC mapping. Below is an example of the view of a Transfer Model as it appears in the Assistment builder.


[Save]

Select Transfer Model:

Knowledge Components for each TransferModel:

WPI-CMU-174-v1.0:

Application: Constructing an equation
Application: Convert fractions to equivalent fractions
Application: convert improper fraction to mixed numeral
Application: convert mixed numeral to improper fraction
Application: Corresponding angles
Application: Counting
Application: Decimals
Application: Elimination
Application: Equilateral traingle
Application: Expressing percent in decimals

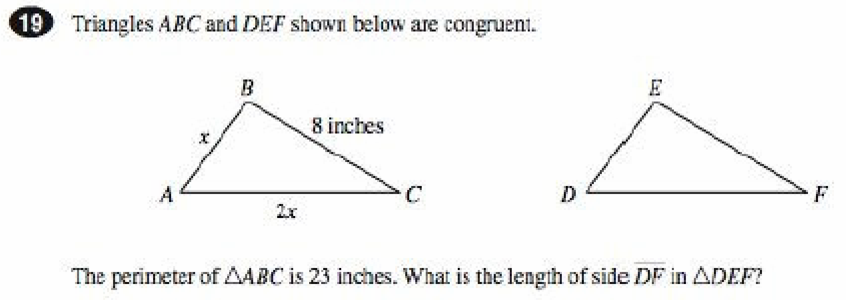
>>

Application: convert mixed numeral to improper fraction

Add more knowledge components...

Figure 2.3: A Transfer Model in the Assistment builder

The establishment of fine-grained transfer models was justified by the idea that specific examples may not be as "cut-and-dry" as originally anticipated. For example, consider the problem displayed below. This problem corresponds to several categories, including geometry, measurement, and number sense, but less refined Transfer Models would only classify this problem into the category of geometry. For more information on Transfer Models, consult Appendix A1. [12]


19 Triangles ABC and DEF shown below are congruent.

The perimeter of ΔABC is 23 inches. What is the length of side $\overline{DF}$ in ΔDEF?

Figure 2.4: An Assistment example.

2.3 Previous Study of Skill Encoding

This project aims to repeat and extend a study that was previously conducted on semi-automatic skill encoding. Rosé et al [2] explored both automatic and semi-automatic skill encoding with the same purpose as this project: to facilitate the tagging of questions in the Assistment System with their related Knowledge Components (KCs). The research performed in that paper not only aimed to study the effectiveness of various text classification models, but to understand the value associated with semi-automatic skill encoding. This paper demonstrated that improving the time required to tag questions did not necessarily have to rely on the algorithms actually making the decisions, but instead could provide suggestions to a user who made the final decision his or herself.

The study imposed a simple hierarchical classification model that is inherent in the Transfer Models studied. This model mapped each KC in the MCAS39 to a single KC in the MCAS5 (see Appendix A1). Each question was first classified into one of the MCAS5, and that result was assumed to be correct. Then the question was classified into one of the MCAS39 KCs that corresponded to the result of the top tier selection. While the first selection was not made with absolute certainty, the increased accuracy at the lower level made up for it. The results yielded by the hierarchical classification model were an improvement over basic classification.

In cases where fully automatic coding is unreliable, Rosé et al suggest employing a semi-automatic solution. A test was employed to quantify the potential time that is saved using such a method over manually encoding each question. Given 45 minutes of training, ten participants were required to map KCs to questions manually, while another ten were given an interface that first suggested a possible KC. Fifty percent of the time, the suggested KC was correct. Results indicated that offering a suggestion improved accuracy without decreasing time. Furthermore, it was speculated that a higher accuracy in the suggestion would both improve classification time and accuracy. [2]

2.4 Applications Related to Skill Encoding

Recently, Pardos et al [13] have studied the relationship between Transfer Models and the ability to accurately predict how students will perform on actual standardized tests. This study used Bayesian networks to attempt to attribute correct and incorrect answers on tutoring question to a particular Knowledge Component (KC). This information was then applied to an expected sampling of potential test questions in order to predict a result. Tests were run using several different Transfer Models, ranging from a model containing only a single KC to the April Transfer Model (see Appendix A1). The purpose was to assess whether a more fine-grained set of KCs made such predictions more accurate.

Evidence of this study indicated that fine-grained Transfer Models do indeed offer a better method of gauging student performance. While the MCAS39 Transfer Model displayed the greatest ability to predict scores, the April model showed promising results. Furthermore, this gap in performance was attributed to a difference in the sizes of the data sets that were used for training purposes. It was speculated that if the April Transfer Model had a greater representation in the training data, it could offer an improvement to predictions. Based on these results, it is increasingly important to facilitate the automated mapping of KCs to questions, as finer-grained models are inherently more time-consuming to work with. [13]

3 Design Decisions

This chapter outlines the specific choices that were made when creating the experiments in this project. In particular, it covers the text classification package that contained the machine learning algorithms employed by the experiments. It also elaborates on the specific decisions that were made in finding the correct dataset for our experiments. Each decision was weighed with respect to simplicity, feasibility, ease of use, and effectiveness. Without an effective basis on which to run the experiments, the process would have been overly time consuming and the results would have been difficult to assess in any meaningful light.

3.1 Mallet Text Classification

This project employs the Mallet toolkit [14] to assist in text classification. Mallet includes several utilities for manipulating data and supports multiple text classification algorithms. Mallet was chosen over several other text classification packages primarily due to its ease of use. Mallet was extremely well documented, which made it very simple to find relevant methods despite the large scope of the package as a whole. Furthermore, Mallet used a very straight-forward interface, which allowed for a clear subdivision of tasks. This was an important consideration because of the rapidly evolving world of machine learning algorithms; it is significant that new techniques can be inserted into this system without vast changes to other parts of the system.

All trials were run using a Naïve Bayes classification algorithm. This was chosen over the other algorithms suggested by Rosé et al [2] because neither Voted Perceptron nor Support Vector Machines are supported by Mallet. Other algorithms were considered, but implementing a hierarchical classification model proved simplest with Naïve Bayes. In Mallet, the Naïve Bayes trainer is implemented by splitting each question text into a feature vector, with one word per feature. Each feature is assigned a weight during training so that instances can be classified based on their comparison to the feature vectors derived from the training set.

3.2 Data Set

The question texts that were used as data came from two distinct sources. First, we started with about 280 released 8th grade math test items from the Massachusetts Comprehensive Assessments Systems (MCAS) state test. These items are available on the Massachusetts's Department of Education web site.[1] The rest of the instances are questions that were written at Worcester Polytechnic Institute and at Carnegie Mellon University by graduate students as part of the assembly of a tutoring system for the test items. For each "original" MCAS item, members of the Assistment Project [1] wrote between three and five scaffolding questions, which attempted to break down solving the item into solving a few easier questions.

The original data contained 1258 items, where many of the question texts were tagged with more than one skill. Due to the difficulty of knowing how to evaluate our classifier, we decided to focus only on questions tagged with a single skill. This exclusion left us with 878 question text instances.

Each of the skills, the distribution of which is shown below in Figure 3.1, was used in all data calculations. Note that the skill with the highest number of occurrences, which happened to be named "Pattern Finding", made up less than 10% of the total

[1] http://www.doe.mass.edu/mcas/testitems.html

number of instances, so we would hope to get classification accuracy at least higher than 10%.

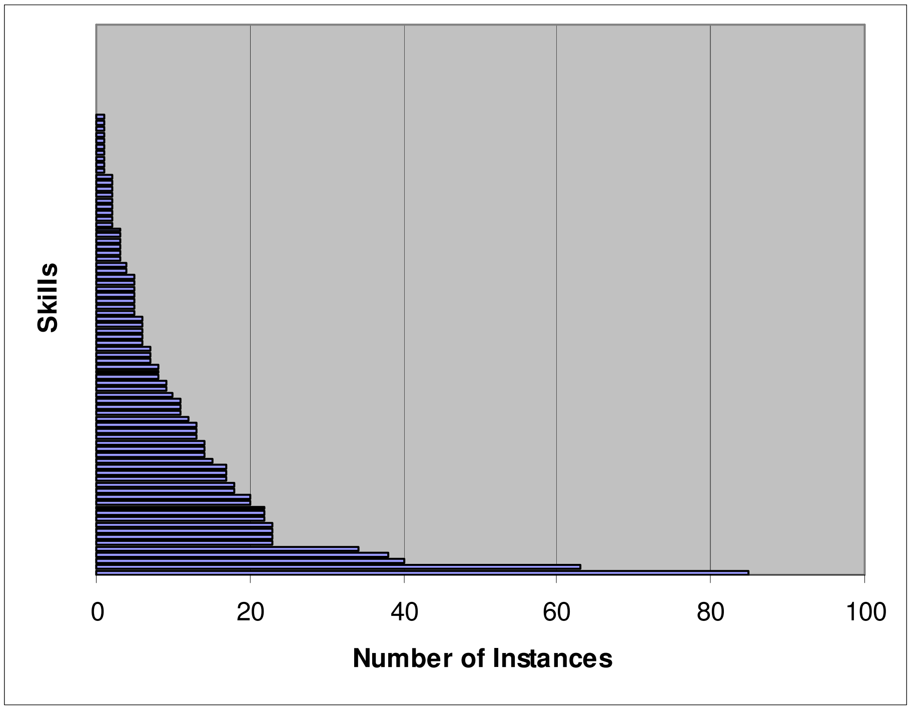


Figure 3.1: Distribution of skills based on number of instances.

Each instance was assigned one or more tags from the "April" transfer model, which contains 78 different skills. Each skill within the April model can also be mapped to exactly one skill from the MCAS5 transfer model, which contains five more general skills. More information on these transfer models can be found in Appendix A1.

4 Methodology

This chapter discusses the steps taken during this project. First discussed is the implementation methodology, wherein the individual programs and methods written to run the experiments are elaborated upon. After implementation was complete, two separate experiments were run to assess the system's performance. The methods employed in these experiments are explained below.

4.1 Implementation

As discussed earlier, this project was implemented using the Mallet text classification toolkit. Since this package was written in Java, it was simplest to implement my system in Java as well. There were several files generated for this project, each of which can be found in Appendix A2.

The first and most important file created was the classification engine, which took as input a data set text file, split the data into a training set and a test set, trained a classifier, and classified the data. For the most part, this file involved simple calls to the trainer and classifier objects within Mallet, but the modifications that were made are significant and will be discussed here. First, Mallet does not support hierarchical classification, so that aspect of the system needed to be implemented by hand. This was done using a method similar to that used by Rosé et al [2]. Each item was classified into one of the more general categories first, the result of which was assumed to be correct. The items were then classified into one of the subcategories for that assumed result. The second modification to the classification engine was the inclusion of a simple method to assess whether one of the top N skills predicted by the classifier was indeed the correct classification.

In order to make this system as flexible as possible, an abstract class was created to handle the specifics of any given transfer model, as well as the mapping between more and less specific transfer models. This class deals with retrieving data sets from the text files, as well as randomizing, splitting, and storing all of the items. Each subclass is expected to implement methods for encoding and decoding each of the skills; for more detail, see Appendix A2. For this project, the only subclass that was implemented for transfer models was one handling encoding and decoding the April transfer model.

Additionally, a suite of files was created for manipulating text files in order to convert data from the data frames fetched from the Assistments database into a single file representing a complete dataset.

Finally, files within the builder were changed to incorporate the classifiers. In the builder, transfer models were simply displayed in alphabetical order to the user. After modifications, the same transfer model was displayed with a variable number of Knowledge Components (KCs) suggested. The suggested KCs are included at the top of the list; the rest of the KCs are still presented in alphabetical order.

4.2 Experiment 1: Direct Classification of Question Texts into Skills

The first experiment was a study of the advantages associated with selecting more than one Knowledge Component (KC) from the April transfer model for each given item. For this experiment, the items were classified using 90% of the data for the training set. The total data set was divided at random in every trial, and the accuracy was gathered for the top N choices for each instance in the testing set. Each test was run a total of five times, and the averages and standard deviations were calculated.

This experiment also included a qualitative analysis of the classification accuracies associated with each of the individual skills. We classified the items using 50% of the data for the training set and outlined the individual classification accuracies for each of the 78 skills. A different ratio of training set to testing set was used in this part of the experiment because a 90% split resulted in too many of the skills having no instances present in the testing set, which made gathering data on those skills impossible. This experiment was also performed five times, and the data were averaged and compared to the number of instances representing each skill in the data set.

4.3 Experiment 2: Using Hierarchical Classification to Improve Performance

Our second experiment was based on an assessment of our hierarchical classification model, which would be compared to the performance of the classifier described in Experiment #1. Our method is based upon Rosé et al [2], who had the insight that by first classifying items in a small number of categories and later classifying those into categories that are nested with the broader categories a higher accuracy and could be achieved. To do this, we used a 90% training split, as in the first experiment. Our hierarchical model first selects a broad category from the MCAS5 and then classifies it into one of the April transfer model skills. As stated earlier, the broad category is not selected with absolute certainty, but the results for selecting a single skill hierarchically are expected to be an improvement over selecting one of April transfer model skills directly. Each trial was run five times, and the averages were calculated and compared to those of a direct classification.

5 Results and Analysis

This chapter outlines the results of the experiments and explains how those results impact the development of system that supports semi-automatic skill encoding. For each experiment, the results were considered based on their potential to save time for the user, not necessarily based on their overall accuracy.

5.1 Analysis of Experiment 1

The results of the direct classification of each item to a single Knowledge Component (KC) are shown below in Table 5.1. N represents the threshold for assessing whether the correct KC could be found in one of the top N choices made by the classifier.

Table 5.1: Accuracies for Top N Choices

If we ask MALLET to pick one skill from the list of 78, 40% of the time it will pick the correct skill.

N	Accuracy	Standard Deviation
1	0.4096	0.0084
2	0.5194	0.0239
3	0.5663	0.0212
4	0.6005	0.0234
5	0.6369	0.0125
6	0.6738	0.0099
7	0.6875	0.0160
8	0.7098	0.0125
9	0.7130	0.0103
10	0.7303	0.0142

Though we are mainly interested in comparing improvements with different methods, the correct way to interpret this 41% accuracy is that this is the probability that you would classify any item correctly if you took that item at random from our data set. Note that this number is different, and higher, than the value you would expect to get if you first picked a skill at random and then selected an item at random from that skill.

From a human-computer interface perspective, it is reasonable to suggest several choices from the user. If it were possible to limit the choices presented to a user when they are tagging new items with reasonable accuracy, one could expect a significant decrease in the amount of time taken to enter new items. With as many as 78 different skills to choose from, narrowing down the selection can be accomplished effectively without only presenting one recommended skill to the user. This section discusses the possible improvements that stem from selecting the top two or more skill choices for a given item.

The accuracies shown in Table 5.1 indicate the chance that the correct classification for a given item is one of the top N choices generated. These data indicate a substantial improvement over the initial accuracy by adding one or two additional skill selections. However, if the top five skills or more are selected, each additional skill selection seems to add between two and three percent accuracy. The goal is to narrow the selection of choices to as few as possible while still providing an accuracy that is high enough to assist the user in tagging items. Based on these results, it is apparent that the greatest benefits are achieved through selecting a small number of top choices.

5.1.1 Qualitative Analysis of Individual Skills

Some skills have a more reliable representation than others; it is obvious that skills with very few instances in the data set do not yield high classification accuracy. Despite the improving the ratio of testing set to training set in this part of the experiment, some of the skills still did not have a single instance in the testing set. As a result, any skill with fewer than seven instances was not considered at all in the analysis; this was chosen as the cutoff because it is the lowest number that still had skills with at least one instance in the testing set in each of the five trials. The remaining skills are shown below in Figure 5.1; the average standard deviation for this set was 0.16869.

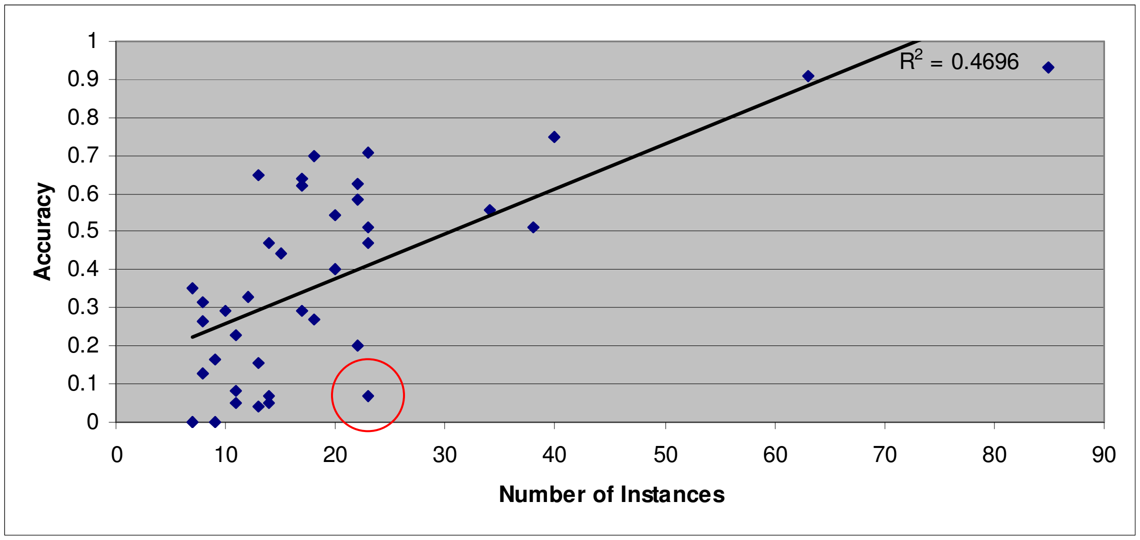


Figure 5.1: Accuracy vs. Number of Instances

In this part of the experiment, the average accuracy over the entire data set was approximately 42%. After all skills with less than seven instances were eliminated from the data, the weighted average was approximately 51%.

A better understanding of why some skills can be classified with higher accuracy than others may provide opportunities for improvement of the classification accuracy of the entire data set. The most general observation regarding the data was that a higher accuracy is related to a greater number of instances.

Skills with 17 instances or less, in general, did not show an accuracy that was lower than the accuracy for the entire data set. Conversely, the skills with the greatest number of instances showed relatively high accuracies in testing. The three skills with the greatest number of instances, Pattern Finding, Probability, and Symbolization-Articulation, yielded results of 0.9294, 0.9071, and 0.7510 respectively, well above those of the entire data set (about 42%). The fact that we observed a correlation between number of instances and classification accuracy is not surprising.

Not all skills with high instances counts showed such promising results, however. Equation Solving, circled in red in Figure 5.1, which ranked eighth for greatest number of instances, showed consistently poor performance. This is indicative that something about the question text in these types of questions makes them difficult to differentiate from other skills. An inspection of these question texts revealed a wide variety of actual question topics that are associated with this skill. Furthermore, many words that appear in these question texts can be logically associated with other skill sets and indeed appear in question texts from other skills. This association may have contributed to the comparatively low accuracy when classifying this skill.

Furthermore, some skills with comparatively small numbers of instances showed surprisingly high classification accuracy. In particular, the *"Linear Area Volume Conversion"* skill achieved an accuracy of 65%, which was about 23% above the accuracy for the total data set, while having fewer than 15 instances. The following are several examples from the data set for this skill:

1. Two rectangles, ABCD and WXYZ, are shown above. The measure of each side of WXYZ is 5 **times** the measure of each corresponding side of ABCD. Which statement is true of the areas of these two rectangles?
2. If we assumed that the length of an edge of the blue **cube** equals to 1, then, based on the problem condition, what is the length of an edge of the red **cube** going to be?
3. Now, let's find the surface **area** of a red **cube**. What is the surface **area** of the red **cube** if we know that the length of an edge of the red **cube** equals to 2?
4. Which of the following operations will give us the number of **times** the surface **area** of the red **cube** is larger that the surface **area** of the blue **cube**?

5. Which of the squares shown above has sides that are twice as long as the sides of **square** A shown on the left?
6. The original problem states that the **area** of **square** A is 4 **square** units. So, what is the **area** of **square** B?

Note that there are keywords associated with these question texts, which may have accounted for the increased accuracy. Each of the thirteen instances for this skill contained one or more of the words "times", "area", "square", or "cube".

Similar trends are seen for the Inequality Solving skill, which had produced an accuracy of about 62% with only 17 instances. All of the 17 items associated with this skill actually contain the word "inequality". In summary, many of the skills that were associated with high accuracy had appeared to have keywords that we often used in the question text, which makes sense.

5.2 Analysis of Experiment 2

The results of the hierarchical classification experiment are reported below in Table 5.2. Once again, N represents the number of choices selected from the classifier to be compared against the correct KC. These results are listed alongside the results from the first experiment, and the higher of the two in any given row appears in bold. The standard deviation reported corresponds to the hierarchical classification; see Table 5.1 for standard deviations associated with direct classification.

Table 5.2: Basic Classification vs. Hierarchical Classification

N	Basic Classification	Hierarchical Classification	Standard Deviation
1	0.4096	**0.4519**	0.0136
2	0.5194	**0.5207**	0.0166
3	0.5663	**0.5722**	0.0194
4	**0.6005**	0.5745	0.0235
5	**0.6369**	0.6137	0.0385

From Table 5.2, we see that the hierarchal classification had higher accuracy when asked to pick a single best KC. However, we decided to test the application of hierarchy for providing the user with two or more KCs. A hierarchical classification is more effective when the best item is selected, and is even an improvement when the top two choices are selected. However, when the top three or more KCs for a given item are selected, a direct classification into the April transfer model is either (approximately) as accurate as or more accurate than a hierarchical classification. Furthermore, it is evident that the rate of improvement in the performance of the hierarchical classification model decreases sharply, relative to the basic classification, if more than the top three options are selected. This is probably due to the accuracy of the initial selection from the MCAS5 transfer model; the effectiveness of hierarchical classification can be severely limited by the accuracy of the top tier when many options are selected from the lower tier possibilities. This suggests that a hierarchical model would serve as an effective part of a semi-automatic skill coder, but would work most effectively if supplemented in some way. For instance, we could use the best one or two guesses from the hierarchal classifier, and then pick three or four choices from the basic classifier. An alternative approach would be to use the confidence of the initial classifier that classified all items onto one of five categories to inform selection of the best classification at the next classification hierarchal of 78 skills. This would enable the top five choices to come from different parts of the MCAS5.

6 Conclusions and Future Work

In conclusion, it appears that we can use text-based Naïve Bayes classification somewhat effectively with an accuracy rate of about 40% when picking one of 78 skills, and an accuracy of about 51% when picking one of 39 adequately represented skills. This appears to be the basis for an effective aid for the people responsible for coding these items. We think it is reasonable that we could provide coders the top five skills as suggestions, and it turns out that in two-thirds of the cases the system could suggest the correct coding. We speculate our surprising accuracy might be related to the fact that having 78 skills means that you can divide up these instances in large groups of highly distinct item types. We also investigated using hierarchal classification, and got some improvements. These results will be published in the 8th International Conference on Intelligent Tutoring Systems. [15]

Rosé et al [2] employed a form of feature selection that replaced certain symbols and numbers with tags corresponding to their context. The figures replaced include but are not limited to fractions, monetary values, percentages, and dates. For the future, we are considering implementing similar feature selection and testing how classification accuracies are affected. Additional work can also be done in quantifying the value of presenting more than one KC to the user. There is an inherent threshold that has yet to be discovered regarding the potential time-saving benefits of presenting multiple KCs. Finally, investigations into more powerful text classification algorithms are already underway, and could prove beneficial further improving classification time.

Appendix A1

MCAS5 Transfer Model[2]:

- Data Analysis Statistics Probability
- Geometry
- Measurement
- Number Sense Operations
- Patterns Relations Algebra

MCAS39 Transfer Model[2]:

- Understanding data generation techniques
- Understanding data presentation techniques
- Understanding data representation techniques
- Understanding concept of probabilities
- Understanding polygon geometry
- Understanding and applying congruence and similarity
- Understanding line intersection angle formation
- Understanding and applying Pythagorean Theorem
- Using geometry tools
- Understanding plane translations
- Identifying 3d figures
- Translating 3d to 2d
- Using appropriate units of measurement
- Converting from one measure to another
- Using measurement formulas and techniques
- Using ratio and proportion
- Representing and understanding rate of change
- Understanding number representations
- Estimating and computing various numbers
- Using estimates over exacts
- Using appropriate operations with rational numbers
- Using common irrational numbers
- Using ratios and proportions
- Representing numbers in scientific notations
- Applying number theory
- Understanding absolute value
- Applying concept of powers and roots
- Applying properties of operators
- Using inverse operations
- Understanding patterns
- Understanding linear growth
- Evaluating algebraic expressions
- Understanding variable identity
- Creating and using symbolic expressions
- Understanding line-slope concept
- Understanding variable roles
- Setting up and solving equations
- Understanding co-variation
- Modeling co-variation

[2] http://www.doe.mass.edu/frameworks/math/2000/final.pdf

April Transfer Model[3]:

- 360 Degrees in Circle
- Adding Decimals
- Addition
- Algebraic Manipulation
- Area
- Area Concept
- Area of Circle
- Circle Graph
- Circumference
- Combinatorics
- Comparing Fractions
- Compounding Interest
- Congruence
- Conversion of fractions decimals percents
- Discount
- Divide Decimals
- Divisibility
- Division
- Equation Concept
- Equation Solving
- Equilateral Triangle
- Equivalent Fractions Decimals Percents
- Evaluating Functions
- Exponents
- Finding Percents
- Fraction Decimals Percents
- Fraction Division
- Fraction Multiplication
- Fractions
- Graph Shape
- Graph Types
- Histogram
- Increasing Percent (Sales Tax)
- Inducing Functions
- Inequality Solving
- Integers
- Interpreting Linear Equations
- Interpreting Number Line
- Isosceles Triangle
- Knowing English and Metric Terms
- Least Common Multiple
- Linear Area Volume Conversion
- Making Sense of Expressions and Equations
- Mean
- Meaning of PI
- Measurement
- Measurement Use Ruler
- Median
- Mode
- Multiplication
- Multiplying Decimals
- Multiplying Positive Negative Numbers
- Number Line
- Number Theory
- Of Means Multiply
- Order of Operations
- Ordering Numbers
- Ordering Fractions
- Ordering Decimals
- Pattern Finding
- Percent Of
- Percents
- Perimeter
- Plot graph
- Point Plotting
- Prime Number
- Probability
- Properties of Geometric Figures
- Properties of Solids
- Proportion
- Pythagorean Theorem
- Qualitative Graph Interpretation
- Range
- Rate
- Rate with Distance and Time
- Reading graph
- Reciprocal
- Reduce Fraction
- Rounding
- Scale
- Scientific Notation

[3] www.assistment.org

- Similar Triangles
- Simple Calculation
- Slope
- Square Root
- Statistics
- Statistics Concept
- Stem and Leaf Plot
- Substitution
- Subtracting Decimals
- Subtraction
- Sum of Interior Angles more than 3 Sides
- Sum of Interior Angles Triangle
- Supplementary Angles
- Surface Area
- Surface Area and Volume
- Symbolization Articulation
- Transformations/Rotations
- Transversals
- Triangle Inequality
- Understanding Line Slope Intercept
- Unit Conversion
- Venn diagram
- Volume
- X Y Graph
- Long division

Appendix A2

```
/*
 * File Name: ClassifierTester.java
 * Author: Kevin Kardian
 * This file was used for testing the various classifier trainers.
 */

package classification;

import java.util.ArrayList;
import mapping.AprilMapping;
import mapping.TMMapping;
import edu.umass.cs.mallet.base.classify.Classification;
import edu.umass.cs.mallet.base.classify.Classifier;
import edu.umass.cs.mallet.base.classify.ClassifierTrainer;
import edu.umass.cs.mallet.base.classify.NaiveBayesTrainer;
import edu.umass.cs.mallet.base.pipe.CharSequence2TokenSequence;
import edu.umass.cs.mallet.base.pipe.FeatureSequence2FeatureVector;
import edu.umass.cs.mallet.base.pipe.Pipe;
import edu.umass.cs.mallet.base.pipe.SerialPipes;
import edu.umass.cs.mallet.base.pipe.Target2Label;
import edu.umass.cs.mallet.base.pipe.TokenSequence2FeatureSequence;
import edu.umass.cs.mallet.base.pipe.TokenSequenceLowercase;
import edu.umass.cs.mallet.base.pipe.TokenSequenceRemoveStopwords;
import
edu.umass.cs.mallet.base.pipe.iterator.ArrayDataAndTargetIterator;
import edu.umass.cs.mallet.base.types.Instance;
import edu.umass.cs.mallet.base.types.InstanceList;

public class ClassifierTester {

      static ClassifierTrainer trainer = new NaiveBayesTrainer();
      //modify this to test different algorithms
      static TMMapping data;
      //general transfer model to control the data
      static Pipe [] pipelist = new Pipe[5];
      //a set of pipes for hierarchical classification
      static double ratio = .9;
      //the portion of the data to be used for training

      private static int decodeMCAS5(String target)
      //This method ensures that the indices used remain consistent
throughout the program
      {
            if (target.equals("D")) return 0;
            else if (target.equals("G")) return 1;
            else if (target.equals("N")) return 2;
            else if (target.equals("M")) return 3;
            else if (target.equals("P")) return 4;
            return -1;
      }

      private static Pipe newPipe()
      //This method is used to instantiate a new pipe (with the same
arguments) for each classifier
```

```
      {
            Pipe instancePipe = new SerialPipes (new Pipe[] {
                        new Target2Label (),
                        new CharSequence2TokenSequence (),
                        new TokenSequenceLowercase (),
                        new TokenSequenceRemoveStopwords (),
                        new TokenSequence2FeatureSequence(),
                        new FeatureSequence2FeatureVector(),
                  });
            return instancePipe;
      }

      private static boolean labelling_correct(Classification result,
String correctLabel, int topx)
      //This method checks to see if one of the topx labels for a
classification is correct
      {
            for (int i=0; i<topx; i++)
            {
                  if
(result.getLabeling().getLabelAtRank(i).toString().equals(correctLabel)
)
                        return true;
            }
            return false;
      }

      private static Classifier[] trainsubs(InstanceList list)
      //This method trains each of the subclassifiers
      {
            //Each classifier requires its own pipe for controlling
data
            InstanceList [] subs = new InstanceList[5];
            for (int i=0; i<5; i++)
            {
                  pipelist[i] = newPipe();
                  subs[i]=new InstanceList(pipelist[i]);
            }

            //Parse the list of instances to allocate each instance to
its corresponding broad category
            for (int i=0; i<list.size(); i++)
            {
                  String mydata =
list.getInstance(i).getData().toString();
                  String mytarget =
list.getInstance(i).getTarget().toString();
                  String mylabel =
list.getInstance(i).getTarget().toString().substring(0,1);
                  subs[decodeMCAS5(mylabel)].add(mydata, mytarget,
null, null);
            }

            //Create classifiers from their corresponding instances
            Classifier [] subclass = new Classifier[5];
            for (int i=0; i<5; i++)
            {
                  subclass[i] = trainer.train(subs[i]);
            }

            return subclass;
      }

      private static ArrayList hier_classify(ArrayList broad_results,
InstanceList list, Classifier[] subclass)
      //This method classifies each instance to subcategories, assuming
that the broad categories into which
      //    the instances have already been placed are correct
      {
            ArrayList new_results = new ArrayList();
            for (int i=0; i<broad_results.size(); i++)
            {
                  Classification result =
(Classification)broad_results.get(i);
                  String label =
result.getLabeling().getBestLabel().toString();
                  String data =
result.getInstance().getData().toString();
                  String correct =
((Instance)list.get(i)).getTarget().toString();
                  int index = decodeMCAS5(label);
                  Instance myInst = new Instance(data, correct, null,
null, pipelist[index]);
                  new_results.add(subclass[index].classify(myInst));
            }
            return new_results;
      }

      public static void main(String[] args)
      {
            //instantiating objects
            data = new AprilMapping();
            InstanceList broadlist = new InstanceList(newPipe());
            InstanceList speclist = new InstanceList(newPipe());

            //populating objects with relevant data
            broadlist.add (new ArrayDataAndTargetIterator
(data.getDataList(), data.getBroadTargetList()));
            speclist.add (new ArrayDataAndTargetIterator
(data.getDataList(), data.getSpecTargetList()));
            InstanceList [] blists = broadlist.splitInOrder (new
double[] {ratio, 1-ratio});
            InstanceList [] slists = speclist.splitInOrder (new
double[] {ratio, 1-ratio});

            //training classifiers
            Classifier broadclass = trainer.train(blists[0]);
            Classifier specclass = trainer.train(slists[0]);
            Classifier [] subclass = trainsubs(slists[0]);

            //running classifiers
            ArrayList spec_results = specclass.classify(slists[1]);
            ArrayList broad_results = broadclass.classify(blists[1]);
            ArrayList hier_results = hier_classify(broad_results, 
slists[1], subclass);

            double average;
            int basiccorrect = 0;
            int hiercorrect=0;
            int total = 0;

            for (int i=0; i<spec_results.size(); i++)
            {
                  //tracking correct classifications
                  Classification result = 
(Classification)spec_results.get(i);
                  String correctLabel = data.getTarget(data.getSize()-
spec_results.size()+i);
                  if (labelling_correct(result, correctLabel, 3))
                        basiccorrect++;
                  total++;
            }

            //outputting results
            System.out.println("Basic Classification:");
            average = (double)basiccorrect/total;
            System.out.println("accuracy: " + average);
            System.out.println("correct: " + basiccorrect);
      }

}
```

```
/*
 * File Name: ClassifierBuilder.java
 * Author: Kevin Kardian
 * This file was used creating serialized classifier objects
 * for use in the builder.
 */

package classification;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

import mapping.AprilMapping;
import mapping.TMMapping;

import edu.umass.cs.mallet.base.classify.Classifier;
import edu.umass.cs.mallet.base.classify.ClassifierTrainer;
import edu.umass.cs.mallet.base.classify.NaiveBayesTrainer;
import edu.umass.cs.mallet.base.pipe.CharSequence2TokenSequence;
import edu.umass.cs.mallet.base.pipe.FeatureSequence2FeatureVector;
import edu.umass.cs.mallet.base.pipe.Pipe;
import edu.umass.cs.mallet.base.pipe.SerialPipes;
import edu.umass.cs.mallet.base.pipe.Target2Label;
import edu.umass.cs.mallet.base.pipe.TokenSequence2FeatureSequence;
import edu.umass.cs.mallet.base.pipe.TokenSequenceLowercase;
import edu.umass.cs.mallet.base.pipe.TokenSequenceRemoveStopwords;
import
edu.umass.cs.mallet.base.pipe.iterator.ArrayDataAndTargetIterator;
import edu.umass.cs.mallet.base.types.InstanceList;

public class ClassifierBuilder {

      static ClassifierTrainer trainer = new  NaiveBayesTrainer();
      static TMMapping data;

      public static Pipe newPipe()
      //This method is used to instantiate a new pipe (with the same
arguments) for each classifier
      {
            Pipe instancePipe = new SerialPipes (new Pipe[] {
                        new Target2Label (),
                        new CharSequence2TokenSequence (),
                        new TokenSequenceLowercase (),
                        new TokenSequenceRemoveStopwords (),
                        new TokenSequence2FeatureSequence(),
                        new FeatureSequence2FeatureVector(),
                  });
            return instancePipe;
      }

      public static void main(String[] args) throws
FileNotFoundException, IOException {
            //instantiate objects
            data = new AprilMapping();
            InstanceList speclist = new InstanceList(newPipe());
            //populate objects with relevant data
            speclist.add (new ArrayDataAndTargetIterator
(data.getDataList(), data.getSpecTargetList()));
            InstanceList [] slists = speclist.splitInOrder (new
double[] {1, 0});

            //train the classifier
            Classifier sclass = trainer.train(slists[0]);

            //output the classifier to a file
            ObjectOutputStream s = new ObjectOutputStream(new
FileOutputStream("NaiveBayes-April"));
            s.writeObject(sclass);
            s.flush();
            s.close();
      }
}
```

```
/*
 * File Name: TMMapping.java
 * Author: Kevin Kardian
 *
 * This file is an abstract class for retrieving a dataset
 * from a text file and storing it in memory.  Subclasses
 * of this file will need to implement encode() and decode(),
 * which are used to create a specific hierarchical mapping.
 */

package mapping;

import java.util.ArrayList;
import java.util.Random;
import java.util.StringTokenizer;

import fileUtils.FReader;

public abstract class TMMapping {

            protected FReader dataset;
            //Controls the file from which the dataset is read.
            protected ArrayList data;
            //Stores the question texts
            protected ArrayList broad_targets;
            //Stores the broad knowledge components
            protected ArrayList spec_targets;
            //Stores the specific knowledge components
            protected int[] lookup;
            //A lookup array that is randomized to preserve the
            //original data order

            public int getSize()
            //Returns the size of the dataset.
            {
                  return data.size();
            }

            protected TMMapping(String sourcefile)
            //Constructor for a new mapping.
            //Gets all data and targets from the sourcefile and
            //stores it in random order.
            {
                  dataset = new FReader(sourcefile);
                  populate();
                  randomize();
            }

            private void populate()
            //Gets all data from the dataset and stores it in the
            //appropriate member variables.
            {
                  data = new ArrayList();
                  broad_targets = new ArrayList();
                  spec_targets = new ArrayList();
                  String line = dataset.readline();
                  while (line!=null)
                  {
                        StringTokenizer tok = new StringTokenizer(line,
"\t");
                        tok.nextToken();
                        data.add(tok.nextToken());
                        String target = tok.nextToken();
                        broad_targets.add(target.substring(0,1));
                        spec_targets.add(target);
                        line = dataset.readline();
                  }
            }

            private void randomize()
            //Sets lookup to a randomly order list that corresponds
            //to indices of the ArrayList member variables.
            {
                  Random rand = new Random();
                  lookup = new int[getSize()];
              for (int i = 0; i < getSize(); i++)
                  lookup[i] = i;

              for (int i = 0; i < getSize(); i++)
              {
                  // choose randomly from remaining elements
                  int r = i + rand.nextInt(getSize() - i);
                  int swap = lookup[r];
                  lookup[r] = lookup[i];
                  lookup[i] = swap;
              }
            }

            public ArrayList getDataList()
            //Returns the member variable data sorted by lookup.
            {
                  ArrayList sortedData = new ArrayList();
                  for (int i=0; i<data.size(); i++)
                  {
                        sortedData.add(data.get(lookup[i]));
                  }
                  return sortedData;
            }

            public ArrayList getSpecTargetList()
            //Returns the member variable spec_targets sorted by
lookup.
            {
                  ArrayList sortedTargets = new ArrayList();
                  for (int i=0; i<spec_targets.size(); i++)
                  {
                        sortedTargets.add(spec_targets.get(lookup[i]));
                  }
                  return sortedTargets;
            }

            public ArrayList getBroadTargetList()
            //Returns the member variable broad_targets sorted by
lookup.
```

```
            {
                  ArrayList sortedTargets = new ArrayList();
                  for (int i=0; i<broad_targets.size(); i++)
                  {

      sortedTargets.add(broad_targets.get(lookup[i]));
                  }
                  return sortedTargets;
            }

            public String getTarget(int index)
            //Returns the specific knowledge component at the given
index.
            {
                  return spec_targets.get(lookup[index]).toString();
            }

            public abstract String encode (int value);
            public abstract int decode (String target);
            //Tracks the encoding and decoding scheme as it applies to
            //a specific transfer model.

}
```

```
/*
 * File Name: AprilMapping.java
 * Author: Kevin Kardian
 *
 * This file is a subclass of TMMapping, and creates a mapping
 * that is specific to the April transfer model.
 *
 * Note: the Xs in the encoding and decoding scheme correspond
 * to deprecated knowledge components that are simply included
 * as place holders.
 */

package mapping;


public class AprilMapping extends TMMapping {

      public AprilMapping ()
      {
            super("simpledataset-april.txt");
      }

      public String encode(int value) {
            switch (value) {
                  case 1: return "X.360-Degrees-in-Circle";
                  case 2: return "N.Adding-Decimals";
                  case 3: return "N.Addition";
                  case 4: return "P.Algebraic-Manipulation";
                  case 5: return "M.Area";
                  case 6: return "M.Area-Concept";
                  case 7: return "M.Area-of-Circle";
                  case 8: return "D.Circle-Graph";
                  case 9: return "M.Circumference";
                  case 10: return "D.Combinatorics";
                  case 11: return "N.Comparing-Fractions";
                  case 12: return "N.Compounding_Interest";
                  case 13: return "G.Congruence";
                  case 14: return "X.Conversion-of-fractions-decimals-
percents";
                  case 15: return "N.Discount";
                  case 16: return "N.Divide-Decimals";
                  case 17: return "N.Divisibility";
                  case 18: return "N.Division";
                  case 19: return "P.Equation-Concept";
                  case 20: return "P.Equation-Solving";
                  case 21: return "X.Equilateral-Triangle";
                  case 22: return "N.Equivalent-Fractions-Decimals-
Percents";
                  case 23: return "P.Evaluating-Functions";
                  case 24: return "N.Exponents";
                  case 25: return "N.Finding-Percents";
                  case 26: return "N.Fraction-Decimals-Percents";
                  case 27: return "N.Fraction-Division";
                  case 28: return "N.Fraction-Multiplication";
                  case 29: return "N.Fractions";
                  case 30: return "P.Graph_Shape";
                  case 31: return "X.Graph-Types";
                  case 32: return "D.Histogram";
                  case 33: return "X.Increasing_Percent_(Sales_Tax)";
                  case 34: return "P.Inducing_Functions";
                  case 35: return "P.Inequality-Solving";
                  case 36: return "N.Integers";
                  case 37: return "P.Interpreting-Linear-Equations";
                  case 38: return "D.Interpreting-Numberline";
                  case 39: return "G.Isosceles-Triangle";
                  case 40: return "X.Knowing_English_and_Metric_Terms";
                  case 41: return "N.Least-Common-Multiple";
                  case 42: return "G.Linear-Area-Volume-Conversion";
                  case 43: return "P.Making-Sense-of-Expressions-and-
Equations";
                  case 44: return "D.Mean";
                  case 45: return "M.Meaning-of-PI";
                  case 46: return "M.Measurement";
                  case 47: return "M.Measurement-Use-Ruler";
                  case 48: return "D.Median";
                  case 49: return "D.Mode";
                  case 50: return "N.Multiplication";
                  case 51: return "N.Multiplying-Decimals";
                  case 52: return "X.Multiplying-Positive-Negative-
Numbers";
                  case 53: return "X.Number-Line";
                  case 54: return "N.Number-Theory";
                  case 55: return "N.Of-Means-Multiply";
                  case 56: return "N.Order-of-Operations";
                  case 57: return "N.Ordering_Numbers";
                  case 58: return "X.Ordering-Fractions";
                  case 59: return "N.Ordering_Decimals";
                  case 60: return "P.Pattern-Finding";
                  case 61: return "N.Percent-Of";
                  case 62: return "N.Percents";
                  case 63: return "M.Perimeter";
                  case 64: return "D.Plot_graph";
                  case 65: return "P.Point-Plotting";
                  case 66: return "N.Prime-Number";
                  case 67: return "D.Probability";
                  case 68: return "P.Properties-of-Geometric-Figures";
                  case 69: return "P.Properties-of-Solids";
                  case 70: return "N.Proportion";
                  case 71: return "G.Pythagorean-theorem";
                  case 72: return "P.Qualitative-Graph-Interpretation";
                  case 73: return "D.Range";
                  case 74: return "N.Rate";
                  case 75: return "N.Rate-with-Distance-and-Time";
                  case 76: return "D.Reading_graph";
                  case 77: return "N.Reciprocal";
                  case 78: return "N.Reduce-Fraction";
                  case 79: return "N.Rounding";
                  case 80: return "N.Scale";
                  case 81: return "N.Scientific-Notation";
                  case 82: return "G.Similar-Triangles";
                  case 83: return "N.Simple-Calculation";
                  case 84: return "X.Slope";
                  case 85: return "N.Square-Root";
                  case 86: return "D.Statistics";
                  case 87: return "X.Statistics-Concept";
                  case 88: return "D.Stem-and-Leaf-Plot";
                  case 89: return "P.Substitution";
                  case 90: return "N.Subtracting-Decimals";
                  case 91: return "N.Subtraction";
                  case 92: return "G.Sum-Of-Interior-Angles-more-than-
3-Sides";
                  case 93: return "G.Sum-of-Interior-Angles-Triangle";
                  case 94: return "G.Supplementary_Angles";
                  case 95: return "M.Surface-Area";
                  case 96: return "M.Surface-Area-and-Volume";
                  case 97: return "P.Symbolization-Articulation";
                  case 98: return "G.Transformations/Rotations";
                  case 99: return "G.Transversals";
                  case 100: return "G.Triangle-Inequality";
                  case 101: return
"P.Understanding_Line_Slope_Intercept";
                  case 102: return "M.Unit-Conversion";
                  case 103: return "D.Venn-Diagram";
                  case 104: return "M.Volume";
                  case 105: return "X.X-Y-Graph";
                  case 106: return "X.long_division";
                default: return null;
            }
      }

      public int decode(String target) {
            if (target.equals("X.360-Degrees-in-Circle")) return 1;
            else if (target.equals("N.Adding-Decimals")) return 2;
            else if (target.equals("N.Addition")) return 3;
            else if (target.equals("P.Algebraic-Manipulation")) return
4;
            else if (target.equals("M.Area")) return 5;
            else if (target.equals("M.Area-Concept")) return 6;
            else if (target.equals("M.Area-of-Circle")) return 7;
            else if (target.equals("D.Circle-Graph")) return 8;
            else if (target.equals("M.Circumference")) return 9;
            else if (target.equals("D.Combinatorics")) return 10;
            else if (target.equals("N.Comparing-Fractions")) return 11;
            else if (target.equals("N.Compounding_Interest")) return
12;
            else if (target.equals("G.Congruence")) return 13;
            else if (target.equals("X.Conversion-of-fractions-decimals-
percents")) return 14;
            else if (target.equals("N.Discount")) return 15;
            else if (target.equals("N.Divide-Decimals")) return 16;
            else if (target.equals("N.Divisibility")) return 17;
            else if (target.equals("N.Division")) return 18;
            else if (target.equals("P.Equation-Concept")) return 19;
            else if (target.equals("P.Equation-Solving")) return 20;
            else if (target.equals("X.Equilateral-Triangle")) return
21;
            else if (target.equals("N.Equivalent-Fractions-Decimals-
Percents")) return 22;
            else if (target.equals("P.Evaluating-Functions")) return
23;
            else if (target.equals("N.Exponents")) return 24;
            else if (target.equals("N.Finding-Percents")) return 25;
            else if (target.equals("N.Fraction-Decimals-Percents"))
return 26;
            else if (target.equals("N.Fraction-Division")) return 27;
            else if (target.equals("N.Fraction-Multiplication")) return
28;
            else if (target.equals("N.Fractions")) return 29;
            else if (target.equals("P.Graph_Shape")) return 30;
            else if (target.equals("X.Graph-Types")) return 31;
            else if (target.equals("D.Histogram")) return 32;
            else if (target.equals("X.Increasing_Percent_(Sales_Tax)"))
return 33;
            else if (target.equals("P.Inducing_Functions")) return 34;
            else if (target.equals("P.Inequality-Solving")) return 35;
            else if (target.equals("N.Integers")) return 36;
            else if (target.equals("P.Interpreting-Linear-Equations"))
return 37;
            else if (target.equals("D.Interpreting-Numberline")) return
38;
            else if (target.equals("G.Isosceles-Triangle")) return 39;
            else if
(target.equals("X.Knowing_English_and_Metric_Terms")) return 40;
            else if (target.equals("N.Least-Common-Multiple")) return
41;
            else if (target.equals("G.Linear-Area-Volume-Conversion"))
return 42;
            else if (target.equals("P.Making-Sense-of-Expressions-and-
Equations")) return 43;
            else if (target.equals("D.Mean")) return 44;
            else if (target.equals("M.Meaning-of-PI")) return 45;
            else if (target.equals("M.Measurement")) return 46;
            else if (target.equals("M.Measurement-Use-Ruler")) return
47;
            else if (target.equals("D.Median")) return 48;
            else if (target.equals("D.Mode")) return 49;
            else if (target.equals("N.Multiplication")) return 50;
            else if (target.equals("N.Multiplying-Decimals")) return
51;
            else if (target.equals("X.Multiplying-Positive-Negative-
Numbers")) return 52;
            else if (target.equals("X.Number-Line")) return 53;
            else if (target.equals("N.Number-Theory")) return 54;
            else if (target.equals("N.Of-Means-Multiply")) return 55;
            else if (target.equals("N.Order-of-Operations")) return 56;
            else if (target.equals("N.Ordering_Numbers")) return 57;
            else if (target.equals("X.Ordering-Fractions")) return 58;
            else if (target.equals("N.Ordering_Decimals")) return 59;
            else if (target.equals("P.Pattern-Finding")) return 60;
            else if (target.equals("N.Percent-Of")) return 61;
            else if (target.equals("N.Percents")) return 62;
            else if (target.equals("M.Perimeter")) return 63;
            else if (target.equals("D.Plot_graph")) return 64;
            else if (target.equals("P.Point-Plotting")) return 65;
            else if (target.equals("N.Prime-Number")) return 66;
            else if (target.equals("D.Probability")) return 67;
            else if (target.equals("P.Properties-of-Geometric-
Figures")) return 68;
```

```
            else if (target.equals("P.Properties-of-Solids")) return 
69;
            else if (target.equals("N.Proportion")) return 70;
            else if (target.equals("G.Pythagorean-theorem")) return 71;
            else if (target.equals("P.Qualitative-Graph-
Interpretation")) return 72;
            else if (target.equals("D.Range")) return 73;
            else if (target.equals("N.Rate")) return 74;
            else if (target.equals("N.Rate-with-Distance-and-Time")) 
return 75;
            else if (target.equals("D.Reading_graph")) return 76;
            else if (target.equals("N.Reciprocal")) return 77;
            else if (target.equals("N.Reduce-Fraction")) return 78;
            else if (target.equals("N.Rounding")) return 79;
            else if (target.equals("N.Scale")) return 80;
            else if (target.equals("N.Scientific-Notation")) return 81;
            else if (target.equals("G.Similar-Triangles")) return 82;
            else if (target.equals("N.Simple-Calculation")) return 83;
            else if (target.equals("X.Slope")) return 84;
            else if (target.equals("N.Square-Root")) return 85;
            else if (target.equals("D.Statistics")) return 86;
            else if (target.equals("X.Statistics-Concept")) return 87;
            else if (target.equals("D.Stem-and-Leaf-Plot")) return 88;
            else if (target.equals("P.Substitution")) return 89;
            else if (target.equals("N.Subtracting-Decimals")) return 
90;
            else if (target.equals("N.Subtraction")) return 91;
            else if (target.equals("G.Sum-Of-Interior-Angles-more-than-
3-Sides")) return 92;
            else if (target.equals("G.Sum-of-Interior-Angles-
Triangle")) return 93;
            else if (target.equals("G.Supplementary_Angles")) return 
94;
            else if (target.equals("M.Surface-Area")) return 95;
            else if (target.equals("M.Surface-Area-and-Volume")) return 
96;
            else if (target.equals("P.Symbolization-Articulation")) 
return 97;
            else if (target.equals("G.Transformations/Rotations")) 
return 98;
            else if (target.equals("G.Transversals")) return 99;
            else if (target.equals("G.Triangle-Inequality")) return 
100;
            else if 
(target.equals("P.Understanding_Line_Slope_Intercept")) return 101;
            else if (target.equals("M.Unit-Conversion")) return 102;
            else if (target.equals("D.Venn-Diagram")) return 103;
            else if (target.equals("M.Volume")) return 104;
            else if (target.equals("X.X-Y-Graph")) return 105;
            else if (target.equals("X.long_division")) return 106;
            return 0;
      }
}
```

```
/*
 * File Name: FReader.java
 * Author: Kevin Kardian
 *
 * This file is a class to create a layer of abstraction between
 * java file input and my programs.
 */

package fileUtils;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class FReader {

      private FileReader input;
      //Stores the file input object
      private String fname;
      //Stores the filename as a string
      private BufferedReader buf;
      //An input buffer for use by the java file input methods

      public FReader (String file)
      //Constructor for the FReader class.
      {
            fname = file;
            open();
      }

      void open()
      //Opens the file corresponding to the filename for reading.
      {
            try {
                  input = new FileReader(fname);
            } catch (FileNotFoundException e) {
                  System.err.println("File not found");
                  e.printStackTrace();
            }
            buf = new BufferedReader(input);
      }

      public void close()
      //Closes the file input object.
      {
            try {
                  input.close();
            } catch (IOException e) {
                  System.err.println("Error closing file");
                  e.printStackTrace();
            }
      }

      public String readline()
      //Reads a single line from the file input object.
      {
            String result = null;
            try {
                  result = buf.readLine();
            } catch (IOException e) {
                  System.err.println("Error reading from file");
                  e.printStackTrace();
            }
            return result;
      }

}
```

```
/*
 * File Name: FWriter.java
 * Author: Kevin Kardian
 *
 * This file is a class to create a layer of abstraction between
 * java file output and my programs.
 */

package fileUtils;

import java.io.FileWriter;
import java.io.IOException;

public class FWriter {
      FileWriter output;
      //Stores the file output object
      String file;
      //Stores the filename as a string

      public FWriter(String fname)
      //Constructor for the FWriter class.
      {
            file = fname;
            create();
      }

      FWriter(String fname, boolean append)
      //Constructor that can specify whether or not to append to
      //an existing file.
      {
            file = fname;
            if (append) open();
            else create();
      }

      void create()
      //Creates a new file with the given filename.
      {
            try {
                  output = new FileWriter(file);
            } catch (IOException e) {
                  System.err.println("Could not create the file");
                  e.printStackTrace();
            }
      }

      void open()
      //Opens an existing file with the given filename.
      {
            try {
                  output = new FileWriter(file, true);
            } catch (IOException e) {
                  System.err.println("File not found");
                  e.printStackTrace();
            }
      }
	public void close()
	//Closes the file output object.
	{
		try {
			output.close();
		} catch (IOException e) {
			System.err.println("Error closing file");
			e.printStackTrace();
		}
	}

	public void write(String out)
	//Writes a string to the output object.
	{
	  try {
			output.write(out);
		} catch (IOException e) {
			System.err.println("Error writing to file");
			e.printStackTrace();
		}
	}

	public void writeline(String out)
	//Writes a string to the output object with a new line
	//character at the end.
	{
		write(out+"\n");
	}

}
```

```
/*
 * File Name: combineFrames_april.java
 * Author: Kevin Kardian
 *
 * This file is used for combining the dataframes from the april
 * transfer model into on dataset file.
 *
 * Throughout the file, vary the number of calls to nextToken()
 * according to the format of the original dataframes.
 *
 * NOTE: A complete subclass of TMMapping is required before
 * the data frames can be combined accurately.
 */

package dataUtils;

import java.util.ArrayList;
import java.util.StringTokenizer;
import mapping.AprilMapping;
import mapping.TMMapping;
import fileUtils.FReader;
import fileUtils.FWriter;

public class combineFrames_april
{

	static int index;

	static String questionText(Integer ID)
	//This method gets the question text from relevant files
	{
		FReader questions = new FReader("frame1-8thgrade.txt");
		//the file containing dataframe 1 (or equivalent)
		String line = questions.readline();
		while (line!=null)
		{
			StringTokenizer tok = new StringTokenizer(line,
"\t");
			if (tok.nextToken().equals(ID.toString()))
			{
				String result = tok.nextToken();
				while (tok.hasMoreElements())
				{
					result = tok.nextToken();
				}
				return result;
			}
			line = questions.readline();
		}
		return null;
	}

	static String getcode (StringTokenizer t)
	//This method gets the corresponding code from the TMMapping file
	{
		while (t.hasMoreTokens())
		{
```

```
                  String temp = t.nextToken();
                  index++;
                  if (temp.equals("1"))
                  {
                        TMMapping temp2 = new AprilMapping();
                        return temp2.encode(index);
                  }
            }
            return "";
      }

      public static void main(String[] args)
      {
            ArrayList collected = new ArrayList();
            //a list of the questions that have been seen, to avoid
repeating
            FReader coding  = new FReader("frame2-april.txt");
            //the file containing dataframe 2 (or equivalent)
            FWriter merge = new FWriter("dataset-april.txt");
            //the destination file
            String line = coding.readline();
            line = coding.readline();
            while (line!=null)
            {
                  StringTokenizer tok = new StringTokenizer(line,
"\t");
                  Integer id = Integer.valueOf(tok.nextToken());
                  String text = questionText(id);
                  if ((text!=null) && (!collected.contains(text)))
                  {
                        collected.add(text);
                        tok.nextToken();
                        tok.nextToken();
                        index = 0;
                        String thecode = getcode(tok);
                        while (!thecode.equals(""))
                        {
                              String result = "";
                              result += id.toString();
                              result += "\t";
                              result += text.trim();
                              result += "\t";
                              result += thecode;
                              if (!text.trim().equals(""))
                                    merge.writeline(result);
                              thecode = getcode(tok);
                        }
                  }
                  line = coding.readline();
            }

            coding.close();
            merge.close();
            System.out.println("done");
      }
}
```

```
/*
 * File Name: extractEncoding.java
 * Author: Kevin Kardian
 *
 * This file creates a textfile called temp.txt.
 * It parses a file in q-matrix form and creates the new file
 * in a manner that is consistent with the java code that
 * needs to be written for a subclass of TMMapping.
 */

package dataUtils;

import java.util.StringTokenizer;

import fileUtils.FReader;
import fileUtils.FWriter;

public class extractEncoding {

      public static void main(String[] args) {
            FReader temp = new FReader("frame2-april.txt");
            String line = temp.readline();
            temp.close();
            StringTokenizer tok = new StringTokenizer(line, "\t");

            //NOTE: modify the number of nextToken() calls here
            //depending on the number of tabs that need to be
            //skipped before the knowledge components are listed
            tok.nextToken();
            tok.nextToken();
            tok.nextToken();
            tok.nextToken();
            FWriter dest = new FWriter("temp.txt");
            int i = 1;
            while (tok.hasMoreElements())
            {
                  String result = "case ";
                  result += Integer.toString(i++);
                  result += ": return \"X.";
                  result += tok.nextToken().replace(' ', '_');
                  result += "\";";
                  dest.writeline(result);
            }
            dest.close();
            System.out.println("done");
      }

}
```

```
/*
 * File Name: extractDecoding.java
 * Author: Kevin Kardian
 *
 * This file parses a result file from extractEncoding.java and
 * creates a new file in a manner that is consistent with the
 * java code that needs to be written for a subclass of TMMapping.
 */

package dataUtils;

import java.util.StringTokenizer;

import fileUtils.FReader;
import fileUtils.FWriter;

public class extractDecoding {

      public static void main(String[] args) {
            FReader original = new FReader("encodeApril.txt");
            FWriter dest = new FWriter("reverseApril.txt");
            String line = original.readline();
            StringTokenizer tok = new StringTokenizer(line, "\"");
            tok.nextToken();
            String result = "if (target.equals(\"";
            result += tok.nextToken();
            result += "\")) return 1;";
            dest.writeline(result);
            line = original.readline();
            int i=2;
            while (line!=null)
            {
                  tok = new StringTokenizer(line, "\"");
                  tok.nextToken();
                  result = "else if (target.equals(\"";
                  result += tok.nextToken();
                  result += "\")) return ";
                  result += Integer.toString(i++);
                  result += ";";
                  dest.writeline(result);
                  line = original.readline();
            }
            original.close();
            dest.close();
            System.out.println("done");
      }

}
```

```
/*
 * File Name: simplifyDataset.java
 * Author: Kevin Kardian
 *
 * This file takes any dataset and removes any items with the
 * same question text.  This is used to remove items that have
 * more than one knowledge component associated with them.
 */

package dataUtils;

import java.util.ArrayList;
import java.util.StringTokenizer;

import fileUtils.FReader;
import fileUtils.FWriter;

public class simplifyDataset {

      public static void main(String[] args) {
            ArrayList first = new ArrayList();
            ArrayList second = new ArrayList();
            FReader input1  = new FReader("dataset-april.txt");
            FWriter output = new FWriter("simpledataset-april.txt");

            String line = input1.readline();
            while (line!=null)
            {
                  StringTokenizer tok = new StringTokenizer(line,
"\t");
                  Integer id = Integer.valueOf(tok.nextToken());
                  if (first.contains(id))
                        second.add(id);
                  else first.add(id);
                  line = input1.readline();
            }

            input1.close();
            FReader input2 = new FReader("dataset-april.txt");
            line = input2.readline();
            while (line!=null)
            {
                  StringTokenizer tok = new StringTokenizer(line,
"\t");
                  Integer id = Integer.valueOf(tok.nextToken());
                  if (!second.contains(id))
                        output.writeline(line);
                  line = input2.readline();
            }

            input2.close();
            output.close();
            System.out.println("done");
      }

}
```

References

[1] Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Koedinger, K. R., Junker, B., Ritter, S., Knight, A., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar. R, Walonoski, J.A., Macasek. M.A., Rasmussen, K.P. (2005). The Assistment Project: Blending Assessment and Assisting. In C.K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.) *Proceedings of the 12th Artificial Intelligence In Education*, 555-562. Amsterdam: ISO Press

[2] Rosé, C, Donmez, P., Gweon, G., Knight, A., Junker, B., Cohen, W., Koedinger, K., & Heffernan, K. (2005). Automatic and Semi-Automatic Skill Coding With a View Towards Supporting On-Line Assessment. In C.K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.) *Proceedings of the 12th Artificial Intelligence In Education*, 555-562. Amsterdam: ISO Press.

[3] Mitchell, T. Machine Learning. McGraw-Hill Companies, Inc., 1997. p. 1-19.

[4] Dumais, S., Platt, J., Heckerman, D., Sahami, M. (1998). Inductive Learning Algorithms and Representations for Text Categorization. *Proceedings of the 7th international conference on Information and knowledge management,* November 2-7, 1998, Bethesda, Maryland. p. 148-155.

[5] Guthrie, L., Walker, E., and Guthrie, J. (1994). Document classification by machine: theory and practice. In *Proceedings of the 15th Conference on Computational*

Linguistics - Volume 2 (Kyoto, Japan, August 05 - 09, 1994). International Conference On Computational Linguistics. Association for Computational Linguistics, Morristown, NJ. p. 1059-1063.

[6] McCallum, A., Nigam, K. (1998). A comparison of event models for naive bayes text classification. In AAAI-98 Workshop on Learning for Text Categorization, 1998.

[7] Koedinger, K. R., Anderson, J. R., Hadley, W. H. and Mark, M. A. (1995). Intelligent tutoring goes to school in the big city. *Proceedings of the 7thWorld Conference on Artificial Intelligence and Education*, Charlottesville, NC, AACE, p. 421-428.

[8] Turner, T.E., Macasek, M.A., Nuzzo-Jones, G., Heffernan, N..T, Koedinger, K. (2005). The Assistment Builder: A Rapid Development Tool for ITS. In C.K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.) *Proceedings of the 12th Artificial Intelligence In Education*, 929-931. Amsterdam: ISO Press.

[9] Tatsuoka, K. (1995) Architecture of knowledge structures and cognitive diagnosis: A statistical pattern classification approach. In P. Nichole, S. Chipman & R. Brenonn (Eds.), Alternative Diagnostic Assessment. Hillsdale, NJ: Erlbaum, 1995.

[10] Barnes,T., D. Bitzer, & M. Vouk. (2005) Experimental analysis of the qmatrix method in knowledge discovery. *Proceedings of the 15th International*

Symposium on Methodologies for Intelligent Systems 2005, May 25-28, 2005, Saratoga Springs, NY.

[11] Croteau, E., Heffernan, N. T., Koedinger, K. R. (2004) Why Are Algebra Word Problems Difficult? Using Tutorial Log Files and the Power Law of Learning to Select the Best Fitting Cognitive Model. In J.C. Lester, R.M. Vicari, & F. Parguacu (Eds.) *Proceedings of the 7th International Conference on Intelligent Tutoring Systems.* Berlin: Springer-Verlag., p. 240-250.

[12] Upalekar, R. S. Tools to Help Build Models That Predict Student Learning. Masters thesis. Worcester Polytechnic Institute, 2006. http://www.wpi.edu/Pubs/ETD/browse/by_department/all.html

[13] Pardos, Z., Heffernan, N. T., Anderson, B. (in press) Using Fine-Grained Skill Models to Fit Student Performance with Bayesian Networks. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems.* Publisher and page numbers to be determined.

[14] McCallum, A. & Kachites (2002). "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu.

[15] Kardian, K., Heffernan, N. T. (in press) Knowledge Engineering for Intelligent Tutoring Systems: Assessing Semi-Automatic Skill Encoding Methods

Proceedings of the 8th International Conference on Intelligent Tutoring Systems. Publisher and page numbers to be determined.