Mitch Vierhus
CPE Senior Project
6 / 11 / 2015

MADmath: Word Problem Generator

High Level Overview

Word problems always add a degree of difficulty to obtaining the correct solution because of the intermediate step required of having to translate the question into a solvable math expression. This mental exercise helps develop a person's critical thinking skills and is important to practice because of their close relation to many science problems. The original idea for MADmath was to take the main idea behind the game MADLibs, where random adjectives, nouns, adverbs, and verbs are inserted into pre existing sentence skeletons, and use it to generate word problems. Settings control the words being randomly selected from to generate the problems, allowing the user to have their questions take on any theme they want to make the questions more entertaining. During the course of the project, the abilities of MADmath grew beyond that of simply generating random word problems as it also became a tool to help walk the user through the problem and even supply them with a way to double check most of their calculations.

Existing Products

There are already numerous websites with the ability to quickly generate word problems, however, many of these questions don't come with any form of explanation beyond your solution being correct or not. During implementation the main random question and problem generation, I came to a realization that the real utility of the program as a tool lies in how much it enables to user to do. Having the description of the problem and the correct mathematical expression to solve it available to the user if they get stuck and need help is one of the defining features that separates MADmath from the freely available word problems online.

Only being told if your answer is correct or not is of little help, but even those that supply explanations, just like the way teachers teach, the explanations for the solutions vary in depth and vocabulary. When a person is struggling to learn new material, having to learn new vocabulary and try to understand different explanations impedes their learning process. MADmath attempts to solve this problem by allowing the use of an easily configurable explanation to accompany the question as it it presented to the user. MADmath gives access to an unlimited number of varying word problems, while also supplying the user with consistent explanations and vocabulary, making learning the concept the only thing the user needs to concentrate on.

Not only does MADmath appeal to teachers in that it reduces the time needed to spend creating homework assignments, it also has the ability to save and solve functions defined by the user. MADmath has the capabilities to save most function, with its expression, and either present the user with a GUI calculator capable of solving it or the expression can have its variables randomly filled and be presented to the user as a problem to try and solve. The program isn't just a tool to help users generate a large number of example problems for practice, it also acts as a tool capable of double checking calculations for certain functions.

Ease of Use

Being a completely software based project allows for its distribution to be extremely cheap and fast because it only requires the uploading of the executable to some web server for download. Seeing as its main target audience is teachers and their students, putting the program up on the school's website for download would make it easily accessible to the student body. Because most schools already have websites, adding the ability to download a small simple executable from their servers would not drastically affect their upload costs and could be implemented for a negligible amount.

Another option would be to take the database accessing and problem generating components from the project and use them to develop a web server, capable of the same features, that uses a browser based web client. Either option would allow for the teachers to post their own configuration files for problems to a centralized location on the website; supplying the students with easy access to the problems and their specific explanations and terminology (the same being taught to them in the classroom).

The entire purpose of this program is for educational purposes and I think it would be inappropriate to consider making any sort of profit. The success and usefulness of the program comes from the number available and accurate files of functions and problems. By making the program itself and encouraging teachers to distribute and share their problems and explanations, good teaching practices and effective math problems and explanations will be freely available to the public.

Details of Implementation

MADmath is capable of generating random word problems and expressions for the user to solve and supplying them with tools for help when they are stuck. As the user is attempting to solve a problem, the program supplies an easy to use interface allowing them to request more details about the problem if they are stuck or confused. A feature is also present for writing and saving your own functions that can then either be used to generate solvable expressions to challenge the user or the user can use a GUI calculator capable of solving the given function.

MADmath is perfect for teachers who don't want to waste their time writing multiple examples and want to supply their students with consistent problems and explanations as they learn new material. The program is also a useful for a user looking for a tool to help them solve any sort math problem; the ability to save and call functions allows the user to develop their own functions to help them double check their calculations they work through a problem. With the ability to have a description attached to each of constants used and the problem or function as a whole, MADmath is also a great tool for teaching and solving science related math problems.

The main issue that impacted my approach were the pre existing word problem generators found around the web and deciding how I could build a better one. One of the more difficult challenges associated with my project was determining the best way to improve upon what the other generators offer. The main features that separates MADmath from the other available generators are: offering a description with the problems to help the user if they get stuck, showing the appropriate expression to solve the problem, the ability to add and search the database of problems, words, constants, and functions, an interface to double check the

user's work as they work on solving a problem, and the ability to use multiple windows to help break the problem down into its individual functions.

One of the major limiting factors that impacted my approach was the need to make it extremely configurable so each teacher could produce the problems and explanations that go along well with their teaching of the topic. At first I researched the best way to walk through a problem and attempted to produce my problems for each . Making it so that problems, functions, and constants can be easily added and used for generation made developing the system more challenging as error detection on the inputted data is required so the expression can be solved appropriately.

Attempting to make the program runnable on most platforms directed my approach and selection of libraries used for implementation. I chose to develop the program using Python because of high level language features and its ability to run cross-platform and its large number of available cross platform libraries, including: database management (sqlite3), GUI creation (PyQT4), expression evaluation (py_expression_eval).

Determining which libraries to ultimately use and getting them work appropriately was challenging because of the varying amount of documentation and support each API was receiving. The py_expression_eval library had the most issues and didn't have all its documented features working appropriately. To get the library working with my classes required me to go in a fix up a few of their incomplete features. Having to read through the libraries source code and make a few changes took longer than expected and ended up as one of the more significant challenges I faced during my senior projects implementation.

Design Decisions

At first I was considering using the standard python pickle or shelve libraries for memory management, but once the project became more about allowing the user to enter large amounts of customized problems and data, I decided to used a more powerful library to allowing me to interface with a MySQL database, sqlite3. Using pickle or shelves would have required the interface for searching and storing of data to be restricted to a dictionary, key and value pairs. Using sqlite3 to run and manage a MySQL database, gives MADmath the ability to execute optimized queries allowing for a much faster and organized form of searching through the potentially large amounts of stored data.

Before developing the GUI, I needed to decide on which Python API to use. I have limited experience using the standard Tkinter library, but its limiting features left me wanting more control and customization over the GUI. I eventually decided on using the PyQt.QtGui library because of the large amount of support it has from its active user community. The Qt library allows the program to take on the appearance of the OS environment that it is running on, giving it a familiar look to the user.

Developmental Challenges

When attempting to implement the py_expression_eval library into my problem solving code, I faced a lot of issues with multiple of its features not working as documented. This hurdle forced me to investigate the problems by reading through the source code and making changes to fix certain issues. Having limited experience debugging other people's source code, this

experience was great practice at understanding different code styles and ways of handling certain things. This obstacle really showed how much can be learned from reading other people's code and exposing yourself different ways of solving common coding problems.

Developing this program was my first exposure to creating a GUI based desktop program and using the QtGui library as a whole. I have had past experience developing an IOS app in Swift, but not being on running on a desktop environment seemed to limit the usefulness of the program. Its similarity to Swift and its GUI APIs made it quick to get used to and my familiarity with Python really sped up the development process. The ability to open multiple independent windows to give the user access to a countless number of tools made the potential functionality limitless. Being able to have multiple windows open and running simultaneously allowed me to think of different uses for the tools in MADmath and how they work together. Trying to design the most effective interface became exponentially more complex with the introduction of the multiple windows which a similar IOS app would be restricted from running.

Architecture

There are three main components that make up the program: DBManager, ProblemFactory, and ProblemSolver. These three main classes were responsible for saving new and loading previously added data, generating problems with random values and vocabulary, and solving the generated expressions respectively.

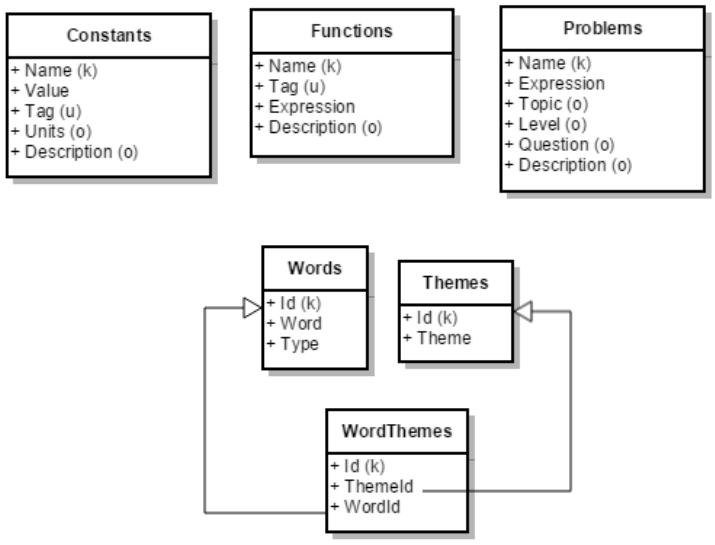

Figure 1: Back end Database schema

Figure 1 is a diagram showing how the data is stored within the mysql database. To implement the feature of being able to select a specific theme for the vocabulary being selected, I needed to use a third table because of their one to many relationship. The DBManager class is solely responsible for accessing the programs database and has custom methods to narrow accordingly.

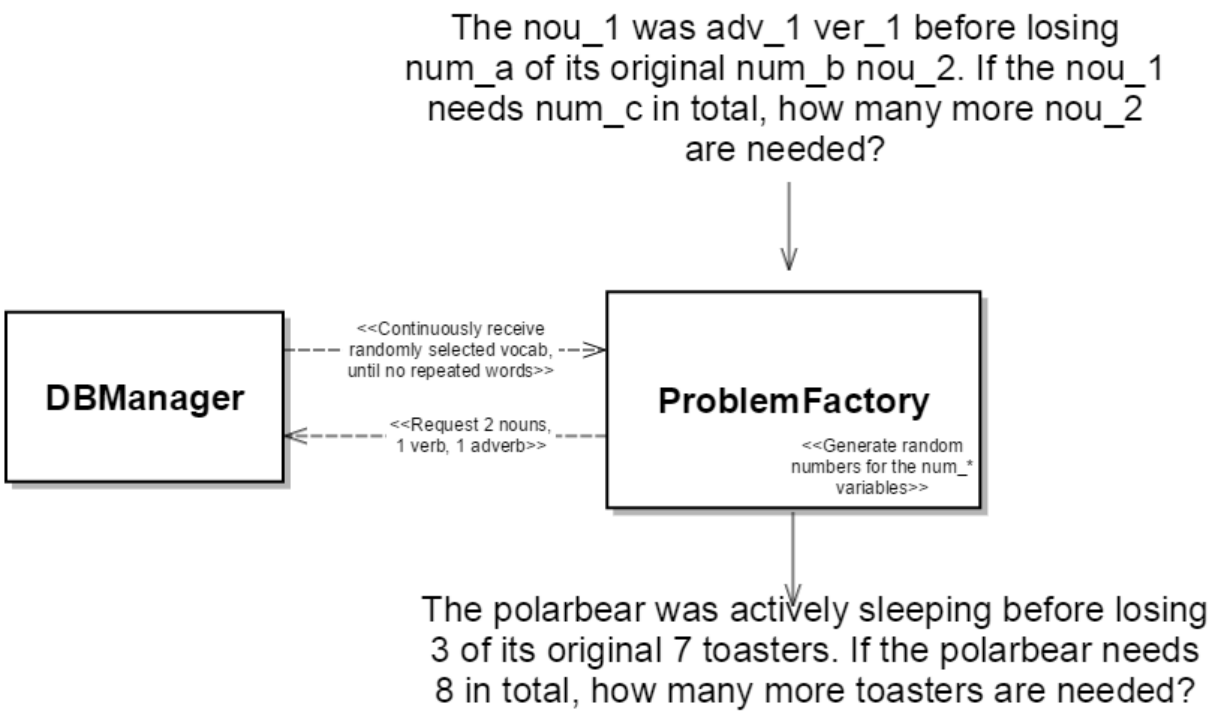

Figure 2: ProblemFactory randomly filling in a word problem

The ProblemFactory component of the program has the responsibility of interacting with the DBManager to request problems and all their necessary components from the database. Once they were retrieved and loaded into the program, The ProblemSolver class was able to substitute in the constants and variables and ultimately solve the expression to check the user's answer.

Next Steps

Python and the libraries I selected to use, attempt to stay backwards compatible as newer versions are released, pretty much eliminating the need for maintenance unless the addition of more features is being pursued. As discussed earlier, MADmath is limited only by the number of users and the content they share and make available and encourages the sustainable use of resources. Also, with time being a valuable resource for teachers, MADmath eliminates their need to sit down and reword different word problems for each assignment/test.

An upgrade to the design that I was working on, but was unable to complete in time, was enabling the use of sub function calls. Implementing this feature would require a bit reworking of code, but would enable for the creation of more complex functions and example problems. This feature was tough to implement because it requires multiple checks needing to be down recursively as functions are passed as the arguments of other functions.

Another upgrade that would be effective to introduce would be the ability to run a local web server allowing for a browser based web client use of the program. This feature would make it possible for teachers to set up their own personal hot spots that they could use to go

over example problems in class. This functionality would further the program's usefulness as a tool and allow for teachers to think of more creative ways of using it effectively.

Appendix

Included project source code files:

- ConfigParser.py - parses .csv configuration files and interactswith DBManager to store them into the programs database
- DBManager.py - responsible for interacting with the sqlite3 library and tts mysql database
- Problem.py - contains the code for the structural classes used in loading and generating problems from the database. Also contains the code for the ProblemFactory static class
- ProblemSolver.py - contains the wrapper code Iwrotefor thepy_expression_eval library used.
- *.pyw - classes responsible for using the Qt interface to generate the GUI
- *.png - contain the MADmath logo
- *.csv - example import files