**Appendix**

*#Digital Aristotle*

*####### Modules Begin*

**import** **json**

**from** **matplotlib.pylab** **import** \*

**import** **numpy** **as** **np**

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

**import** **matplotlib.cm** **as** **cm**

**import** **matplotlib.colors** **as** **colors**

**import** **random**

**import** **time**

**from** **itertools** **import** chain

**import** **nltk**

**import** **os.path**

*####### Modules End*

*###### Data Components Begin*

**class** **Serializable**(object):

*'''*

*Creates a class that allow for the creation of a*

*JSON database while still being able to be used in Python*

*'''*

@classmethod

**def** fromJsonFile(cls,fname):

f = open("Math6")

document = json.loads(f.read().encode('utf8'))

**return** cls.fromJsonObj(document)

@classmethod

**def** fromJsonObj(cls,obj):

**pass**

**def** toJSONObj(self):

**pass**

**def** \_\_str\_\_(self):

**return** (json.dumps(self.toJSONObj(), indent=2))

**class** **Book**(Serializable):

*'''*

*Creates a class that will contain major components*

*(book's name, chapters, isbn, and publish\_date) of*

*the database based on the math book*

*'''*

*# str [Chapter] str str*

**def** \_\_init\_\_(self,name,chapters,publish\_date,isbn):

self.name = name

self.chapters = chapters

self.publish\_date = publish\_date

self.isbn = isbn

@classmethod

**def** fromJsonObj(cls,dictionary):

*'''*

*Creates a dictionalry with components of the book*

*'''*

**return** cls(dictionary['name'],[Chapter.fromJsonObj(obj) **for** obj **in** dictionary['chapters']],dictionary['publish\_date'],dictionary['isbn'])

**def** toJSONObj(self):

*'''*

*Adds information to dictionary based on dictionary*

*values created in the "fromJSONObj" function*

*'''*

**return** {'name' : self.name, 'chapters' : [obj.toJSONObj() **for** obj **in** self.chapters], 'publish\_date': self.publish\_date, 'isbn': self.isbn}

**class** **Chapter**(Serializable):

*'''*

*Makes a class which will contain the a chapter's name,*

*SubChapter object, and the number of pages in the entire chapter*

*'''*

*# str [Sub\_Chapter]*

**def** \_\_init\_\_(self,name,subchapters,pages):

self.name = name

self.subchapters = subchapters

self.pages = set(pages)

*'''*

*Next two functions do the same JSON creation as in the "Book"*

*class, except it uses dictionary values for the chapter's name.*

*subchter, and pages*

*'''*

**def** toJSONObj(self):

**return** {'name' : self.name, 'subchapters' : [x.toJSONObj() **for** x **in** self.subchapters], 'pages':list(self.pages)}

@classmethod

**def** fromJsonObj(cls,dictionary):

**return** cls(dictionary['name'],[SubChapter.fromJsonObj(obj) **for** obj **in** dictionary['subchapters']],dictionary['pages'])

**class** **SubChapter**(Serializable):

*'''*

*Makes a class which will contain a subchapter's name, the*

*text in the subchapter, and the amount of pages in the SubChapter*

*'''*

*# str [Paragraph]*

**def** \_\_init\_\_(self,name,text,pages):

self.name = name

self.text = text

self.pages = set(pages)

*'''*

*The next two fucntions do the same JSON database creation as in the*

*"Book" class, except it uses dictionary values for the SubChapter's*

*name, text in the SubChapter, and the number of pages in the subchpter*

*'''*

@classmethod

**def** fromJsonObj(cls,dictionary):

**return** cls(dictionary['name'],

dictionary['text'],

dictionary['pages'])

**def** toJSONObj(self):

**return** {'name' : self.name, 'text' : self.text, 'pages': list(self.pages)}

*###### Data Components End*

*####### Data Creation Begin*

**def** createNewMathOut():

*'''*

*Function to use classes in "Data components" to create the JSON database*

*by searching through each page of the book, and using the font to determine*

*where the type of object, chapter, subchapters, or text, the information belongs*

*'''*

f = open("LinedMath.json")

book = Book('A First Book in Algebra',[],"","")

document = json.loads(f.read().encode('utf8'))

chapter = None

subchapter = None

**for** (page\_num, page) **in** enumerate(document):

page\_num += 1

**for** line **in** page: *# Searches by each line of text on a single page*

**if** line[u'font'] == 5: *# chapter*

chapter = Chapter(line[u'data'], [], [page\_num])

book.chapters.append(chapter)

subchapter = None

**elif** line[u'font'] == 3: *# Subchapter*

subchapter = SubChapter(line[u'data'], '', [page\_num])

**if** chapter **is** None:

chapter = Chapter('No Name Chapter', [], [page\_num])

chapter.subchapters.append(subchapter)

**elif** subchapter **is** **not** None:

subchapter.text += u'**\n**' + line[u'data']

subchapter.pages.add(page\_num)

chapter.pages.add(page\_num)

**else**:

subchapter = SubChapter("No Name", line[u'data'], [page\_num])

**if** chapter **is** None:

chapter = Chapter('No Name Chapter', [], [page\_num])

chapter.subchapters.append(subchapter)

fout = open("newMathOut.json",'w')

string = json.dumps(book.toJSONObj())

fout.write(string) *# Creates the database as a text file*

**def** readBook():

*'''*

*Function used to search through the JSON in python*

*database created in the fucntions*

*'''*

f = open('newMathOut.json')

book = Book.fromJsonObj(json.loads(f.read()))

**return** book

*####### Data Creation End*

*####### Search Data Begin (Functions)*

**def** levenshtein(source, target):

*'''*

*Function used to compare two strings for their similarity by comparing*

*the individual characters in the strings, and finding what is needed to*

*make them the same (i.e. insertion, substitution, or deletion). Returns*

*a number based on the changes required; the smaller the number, the closer*

*the two strings are to eachother.*

*'''*

**if** len(source) < len(target):

**return** levenshtein(target, source)

**if** len(target) == 0:

**return** len(source)

source = np.array(tuple(source))

target = np.array(tuple(target))

previous\_row = np.arange(target.size + 1)

**for** s **in** source:

*# Insertion*

current\_row = previous\_row + (0.1)

*# Substitution or matching:*

current\_row[1:] = np.minimum(current\_row[1:],np.add(previous\_row[:-1], (target != s) \* (3)))

*# Deletion*

current\_row[1:] = np.minimum(

current\_row[1:],

current\_row[0:-1] + (1))

previous\_row = current\_row

**return** previous\_row[-1]

levenshtein = np.vectorize(levenshtein)

**def** minimumChapter(xs):

*'''*

*Function used to find the chapter with the least amount of difference*

*between the input and a (sub)chpter's text and name given an index where*

*the smallest index is the (sub)chapter with the least difference*

*'''*

smallest = float("inf")

smallestChapter = None

**for** (value, chapter) **in** xs:

**if** value < smallest:

smallest = value

smallestChapter = chapter

**return** smallestChapter

numberOfTimesRan = 0

filtered\_words\_global = []

**def** bestChapter(input\_string, chapters):

*'''*

*Function used to return the chapter or subchapter withe the smallest*

*difference between the input. It compares all of the chapters and*

*subchapters to the input*

*'''*

**def** countWordsInSubChapter(subchapter):

*'''*

*Function that changes a subchapters text into a lost of individual*

*words, and then removest irrelevant words, stop words, from this list.*

*It then creates a negative accumulator where one is subtracted from it*

*if the word matches the input. The sum of levenshtein distance of a word*

*divided by every word is then added to this accumulator.*

*'''*

**global** numberOfTimesRan, filtered\_words\_global

numberOfTimesRan += 1

accum = 0

text = subchapter.text

words = text.split()

filtered\_words = [w.lower() **for** w **in** words **if** **not** w.lower() **in** stopWords **and** **not** isInt(w) **and** w.isalpha() **and** len(w) > 2]

accum -= len([w **for** w **in** filtered\_words **if** w == input\_string])

filtered\_words\_global += filtered\_words

**for** word **in** filtered\_words:

comp = levenshteinDistance(input\_string,word)

accum += float(comp) / len(words)

**return** accum

countWordsInSubChapter = memoize(countWordsInSubChapter)

**def** chapterPlusIndex(chapter):

*'''*

*Function used to return the (sub)chapter with the name and text closest to the*

*input by comparing their levenshtein distances with a created index which is more*

*accurate than the other chapters or SubChapters*

*'''*

index = levenshteinDistance(input\_string, chapter.name)

**if** type(chapter) **is** SubChapter:

index += countWordsInSubChapter(chapter)

**elif** type(chapter) **is** Chapter:

accum = 0

**for** subchapter **in** chapter.subchapters:

accum += countWordsInSubChapter(subchapter)

accum = accum/len(chapter.subchapters)

index += accum

**return** (float(index), chapter)

xs = map(chapterPlusIndex, chapters)

**return** minimumChapter(xs)

**def** memoize(f):

*'''*

*Function used to make searching the database quicker in the "bestChapter" function by repeating*

*a code that has been ran before in the same way so it takes less time to run it again for another*

*chapter*

*'''*

memory = dict()

**def** memoizedFunction(x):

**if** x **in** memory:

**return** memory[x]

**else**:

memory[x] = f(x)

**return** memory[x]

**return** memoizedFunction

**def** getChapter(input\_string,book):

*'''*

*Function used to find the chapter or subchapter that is closet to the input by comparing the input*

*to every chapter and SubChapter with the "bestChapter" function*

*'''*

**return** bestChapter(input\_string, list(chain(\*[c.subchapters **for** c **in** book.chapters])) + book.chapters)

**def** levenshteinDistance(str1,str2):

*'''*

*A function which takes two string and converts them into unicode, and then compares them with the "levenshtein"*

*function after making them lowercase and removing periods from the strings*

*'''*

str1, str2 = unicode(str1), unicode(str2)

**return** levenshtein(str1.lower().rstrip('.'),str2.lower().rstrip('.'))

**def** isInt(string):

*'''*

*A function used to test if a string contains numbers and returns a list of those strings.*

*This is used to remove numbers from the comparison of the input to the text in SubChapters*

*'''*

string = [x **for** x **in** string **if** x **in** '0123456789']

**return** len(string) > 0

*####### Search Data End (Functions)*

*####### Search Data Begin*

*# These stop words are words that are not related to data seach, but used in spoken and written English*

stopwords = """

a about above after again agains all am an and any are aren't as at be because been before being below between both but by can't cannot

could couldn't did didn't do does doesn't doing don't down during each few for from further had hadn't has hasn't have haven't having he

he'd he'll he's her here here's hers herself him himself his how how's i i'd i'll i'm i've if in into is isn't it it's its itself let's me

more most mustn't my myself no nor not of off on once only or other ought our ours ourselves out over own same shan't she she'd she'll

she's should shouldn't so some such than that that's the their theirs them themselves then there there's these they they'd they'll they're

they've this those through to too under until up very was wasn't we we'd we'll we're we've were weren't what what's when when's where

where's which while who who's whom why why's with won't would wouldn't you you'd you'll you're you've your yours yourself yourselves first

second twice third fourth - + one two three four five six seven eight nine ten john many times much will can miles years cent men hours

exercise man must

"""

stopWords = stopwords.split()

**def** search(input\_string,book):

*'''*

*Main function which uses the functions above to compare an input to the*

*book's chapters and subchapter, and returns the chapter with the closest*

*related information*

*'''*

filtered\_words\_global = []

chapter = getChapter(input\_string,book)

**return** displayPages(chapter.pages)

**def** displayPages(pages):

*'''*

*Function used to return the chapter's pages file names which are*

*used to find the file locations of the book's pictures*

*'''*

pages = sorted(list(pages))

**return** [os.path.join('Math6-**%03d**.png'%(x)) **for** x **in** pages]

*####### Search Data End*

*'''*

*Statement used by server to run the search using an input sent*

*from the webpage*

*'''*

**if** \_\_name\_\_ == '\_\_main\_\_':

createNewMathOut()

book = readBook()

search(u'',book)

**import** **tornado.ioloop**

**import** **tornado.web**

**from** **mimetypes** **import** MimeTypes

**import** **urllib**

**import** **tornado.escape**

**import** **data**

**import** **json**

**import** **os.path**

**class** **MainHandler**(tornado.web.RequestHandler):

**def** get(self):

f = open("../HTML/index.html")

s = f.read()

self.write(s)

**class** **HTMLFileHandler**(tornado.web.RequestHandler):

**def** get(self,fileName):

url = urllib.pathname2url(fileName)

t = MimeTypes().guess\_type(url)

**print** t

self.set\_header("Content-Type", '' + t[0] + '; charset="utf-8"')

f = open("../HTML/" + fileName)

s = f.read()

self.write(s)

**class** **MathFiles**(tornado.web.RequestHandler):

**def** get(self,fileName):

url = urllib.pathname2url(fileName)

t = MimeTypes().guess\_type(url)

**print** t

self.set\_header("Content-Type", '' + t[0] + '; charset="utf-8"')

f = open("../IPython/tranMath6/" + fileName)

s = f.read()

self.write(s)

**class** **QueryHandler**(tornado.web.RequestHandler):

**def** get(self, query):

**print** "searching"

data.createNewMathOut()

book = data.readBook()

results = data.search(query,book)

**print** "here"

self.write(json.dumps(results))

application = tornado.web.Application([

(r"/", MainHandler),

(r"/(index\.css|index\.js|Digital%20Aristotle\.png|DA\.png|doge\.png|favicon.ico)", HTMLFileHandler),

(r"/(Math6-[0-9]+\.png)", MathFiles),

(r"/search/(.\*)", QueryHandler)

])

**if** \_\_name\_\_ == "\_\_main\_\_":

application.listen(8888)

tornado.ioloop.IOLoop.instance().start()

<!DOCTYPE html>

**<html** lang="en-US"**>**

**<head>**

**<title>**Digital Aristotle**</title>**

**<link** rel="icon" type="image/png" href="DA.png"**>**

**<script** src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"**></script>**

**<script** type="application/javascript" src="index.js"**></script>**

**<link** rel="stylesheet" type="text/css" href="index.css"**>**

**</head>**

**<body>**

**<img** id="displayImg" src="Digital Aristotle.png"**>**

**<br><br>**

**<input** type="text" placeholder="Search" autocomplete="off" id="search"**>**

**<div** id="output"**>**

**</div>**

**</body>**

**</html>**

$(document).ready(**function**() {

$("#displayImg").hide().show(1000);

$("#search").delay(1000).hide().slideDown(1000);

$("#search").change(**function**() {

**var** search=$("#search").val();

search = $.trim(search);

**if**(search == "doge") {

$("#output").prepend('<img src="doge.png" />').hide(100).show(1000);

}

**else** **if**(search == "back in the day" || search == "lel" || search == "LOL") {

**var** random = Math.floor(Math.random()\*10);

**var** twitter = [

"http://jordanemedlock.com"

];

window.open([twitter[random]]);

}

**else** {

$.getJSON("/search/" + encodeURIComponent(search), **function**(data) {

console.log(data);

**for** (**var** i = data.length - 1; i >= 0; i--) {

console.log(data[i]);

$("#output").prepend('<img src="' + data[i] + '"/>');

}

})

}

})

})

**body** {

**text-align**: **center**;

**background-color**: #DDD

}

#search {

**height**: 10%;

**font-size**: 40pt;

**border**-radius: 10px;

**outline**-**width**: 0;

**width**: 70%;

}

#displayImg {

**width**: 50%;

}

#output {

**display**: **block**;

**margin-left**: 0 **auto**;

**margin-right**: 0 **auto**;

**text-align**: **center**;

**overflow-x**:**scroll**;

**position**: **relative**;

**width**: 100%;

**height**: 100%;

**font-size**: 20pt;

**display**: **inline**-**block**;

}

**Abstract**

Traditional education is generalized and inefficient in society and can be improved on. Traditional tutoring while better than traditional education, still contains significant flaws and can be perfected through the use of computing. More individual and accessible means of tutoring can be created to teach both students who are struggling and excelling the information best suited for them. This will be achieved by the use of searching capabilities and adaptive code to create a more individual and online tutor. Also lectures will be added to allow a more dynamic and less linear learning experience. For the first year, Digital Aristotle contains a database of open source knowledge and returns pertinent information based on the student’s input.

**Background**

Throughout history, humans have searched for better ways to gain knowledge and share it with others. This desire to learn created the education system seeking to share knowledge with everyone. However, the education system has many flaws, most prominently in the lack of individualized attention given to students that are either struggling or excelling in the subject matter. This method used in schools is inefficient and needs a more dynamic and individual approach to ensure better understanding of the subject material. To fix this tutoring was created to fill the need of individualized attention to better develop a student’s understanding of the subject.

Although tutoring is a step above normal education, there are still significant problems and room for improvement. These problems stem from the need of a human to identify the needs of the student and decide the best way for the student to learn. This process is limited in the lack of tutors to fill the need of the students, the expense of the tutor to tutor the student. These problems make traditional tutoring inefficient compared to other means of tutoring. Other tutoring programs have been created to more efficiently teach students.

Other online websites have been created to solve these problems with traditional tutoring, one being Kahn academy. Kahn academy takes a different approach to giving students the assistance necessary to understand the subject. They use tutoring videos and online assignments to help students understand curriculum to solve the problem of expense and amount of tutors. This method however good, still has flaws that have to be addressed. For one the individualization of tutoring is not as prevalent as the free aspect of Kahn academy. This is a problem because the way Kahn Academy teaches is not effective to all students. This lack of individual attention is still covered by traditional tutoring which allows for improvement on both.

Another prominent tutoring system is Aleks. Aleks takes a different approach than Kahn Academy by focusing on the individual aspects of tutoring rather than the human aspects. Aleks uses adaptive questioning to determine what the student is most ready to learn. This process is flawed due to the lack of human interaction which is needed to round out the learning experience. This lack of interaction is also fulfilled by the still imperfect system of tutoring and can be expounded on with the use of all elements of tutoring. All tutoring systems seem to have a lack of well-rounded teaching ability which is prevalent in traditional tutoring which has room for improvement.

**Conclusion**

Education is a key part of society that impacts every citizen looking to excel in life. Currently the approach to teaching students is flawed due to the generalized teaching in classrooms rather than individual attention. Tutoring tries to fill this lack of individualized attention by having a teacher teach the material in what they believe is the most effective to their students understanding. Tutoring still has flaws that can be improved on such as the amount of tutors compared to the amount of students. Other flaws include the expense of hiring a tutor and the time it takes to tutor. The use of adaptive code in Digital Aristotle creates an individualized attention for the students by understanding their strengths and weaknesses. Digital Aristotle impacts the foundation of traditional education by being an easy source to gain a deeper understanding of the knowledge sought. As the first year closes on this project, Digital Aristotle is a base foundation of what is going to be accomplished. The search engine allows for students to search for specific topics within the database and allows them to find relevant information to assist them. The full project solves the problem of students needing individual assistance by using a adaptive algorithm to adapt to the students strengths and weaknesses and to have a human element as well. This approach is unique to Digital Aristotle through the use of multiple tutoring methods used in cooperation.

**Discussion**

The Digital Aristotle tutoring system, much like its human namesake, seeks to transform selective areas of knowledge and effectively communicate this knowledge to students. Although the scope for development of Digital Aristotle in the first year of this multiyear project was limited by design, significant progress in planning continuous improvements in Digital Aristotle. The Digital Aristotle tutoring system consists of a database of knowledge commonly referred to as the knowledge-base, an overarching knowledge retrieval subsystem, and a human-computer interface enabling input queries and output visualization of retrieved information. The first year effort created a baseline system comprised of a knowledge-base and search engine that retrieves information from an algebra textbook based on user input queries. Completion and use of the baseline Digital Aristotle system revealed many insufficiencies. These insufficiencies guiding our planning for future improvements and upgrades to Digital Aristotle. Most notably we plan on increasing the content and diversity of the knowledge-base. In addition, the lack of selectivity in retrieved information, and lack of specificity pertaining to the user input query will need to be fixed as well. This project will be improved on throughout the next 4 years as it is an ambitious project. This organizational system will better segment the goals of Digital Aristotle.

* First Year:
  + In the first year, we would like to create a search engine to answer basic questions that students have about mathematics by displaying specific information about the inquired topic. This process will use the knowledge-base to depict the information searched for. The knowledge base will contain many open source textbooks that have a plethora of different information.
* Second Year:
  + In the second year, we would like to start creating the dynamic elements of Digital Aristotle and better develop the student’s ability with tests and quizzes. These elements will create a foundation for the deeper individualization later on in the project. The assignments will provide a base curriculum for the student and allow for practice and mastery of the topic.
* Third Year:
  + In the third year, we would seek to expand the individualization of the assignments by using adaptive questioning and code to give students more help on subjects they do not understand and are struggling with. The code will interpret the assignment scores and provide more assistance with the material if needed.
* Fourth Year:
  + In the final year, we would seek to give the human element to Digital Aristotle to round out the learning process. This includes lectures, forums, and other human interactive devices that allow for the student to gain assistance from another peer or a teacher of the subject. This is needed because human tutoring is still invaluable due to the more complex understanding of two people.

**Bibliography**

Kahn, Salman. "Khan Academy." *Khan Academy*. N.p., n.d. Web. 29 Mar. 2015.

NYU, and UCI. "ALEKS -- Assessment and Learning, K-12, Higher Education, Automated Tutor, Math." *ALEKS -- Assessment and Learning, K-12, Higher Education, Automated Tutor, Math*. N.p., n.d. Web. 29 Mar. 2015.

Clarke, Wallace. *A First Book in Algebra*. N.p.: n.p., n.d. *Project Gutenberg*. 27 Aug. 2004. Web. 24 Mar. 2015.

**Acknowledgments**

We would like to thank Jordan E. Medlock for his invaluable knowledge in programming and his assistance throughout this entire project. We would also like to thank our parents for constant support and help editing this paper. Lastly we would like to thank CGP Grey for piquing our interest in the topic of individualized education.