The Journal of Computing Sciences in Colleges

Papers of the 33rd Annual CCSC Southeastern Conference

October 25th-26th, 2019
Auburn University
Auburn, AL

Baochuan Lu, Editor
Southwest Baptist University

John Hunt, Regional Editor
Covenant College

Volume 35, Number 4

October 2019

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges. Printed in the USA. POSTMASTER: Send address changes to Susan Dean, CCSC Membership Secretary, 89 Stockton Ave, Walton, NY 13856.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

Jeff Lehman, President (2020), (260)359-4209, jlehman@huntington.edu, Mathematics and Computer Science Department, Huntington University, 2303 College Avenue, Huntington, IN 46750.

Karina Assiter, Vice President (2020), (802)387-7112, karinaassiter@landmark.edu.

Baochuan Lu, Publications Chair (2021), (417)328-1676, blu@sbuniv.edu, Southwest Baptist University - Department of Computer and Information Sciences, 1600 University Ave., Bolivar, MO 65613.

Brian Hare, Treasurer (2020), (816)235-2362, hareb@umkc.edu, University of Missouri-Kansas City, School of Computing & Engineering, 450E Flarsheim Hall, 5110 Rockhill Rd., Kansas City MO 64110.

Judy Mullins, Central Plains Representative (2020), Associate Treasurer, (816)390-4386, mullinsj@umkc.edu, School of Computing and Engineering, 5110 Rockhill Road, 546 Flarsheim Hall, University of Missouri - Kansas City, Kansas City, MO 64110.

John Wright, Eastern Representative (2020), (814)641-3592, wrightj@juniata.edu, Juniata College, 1700 Moore Street, Brumbaugh Academic Center, Huntingdon, PA 16652.

David R. Naugler, Midsouth Representative(2022), (317) 456-2125, dnaugler@semo.edu, 5293 Green Hills Drive, Brownsburg IN 46112.

Lawrence D'Antonio, Northeastern Representative (2022), (201)684-7714, ldant@ramapo.edu, Computer Science Department, Ramapo College of New Jersey, Mahwah, NJ 07430.

Cathy Bareiss, Midwest Representative (2020), cbareiss@olivet.edu, Olivet Nazarene University, Bourbonnais, IL 60914.

Brent Wilson, Northwestern Representative (2021), (503)554-2722, bwilson@georgefox.edu, George Fox University, 414 N. Meridian St, Newberg, OR 97132.

Mohamed Lotfy, Rocky Mountain Representative (2022), Information Technology Department, College of Computer & Information Sciences, Regis University, Denver, CO 80221.

Tina Johnson, South Central Representative (2021), (940)397-6201, tina.johnson@mwsu.edu, Dept. of Computer Science, Midwestern State University, 3410 Taft Boulevard, Wichita Falls, TX 76308-2099.

Kevin Treu, Southeastern Representative (2021), (864)294-3220, kevin.treu@furman.edu, Furman University, Dept of Computer Science, Greenville, SC 29613.
Bryan Dixon, Southwestern Representative (2020), (530)898-4864, bcdixon@csuchico.edu, Computer Science Department, California State University, Chico, Chico, CA 95929-0410.

Serving the CCSC: These members are serving in positions as indicated:
Brian Snider, Membership Secretary, (503)554-2778, bsnider@georgefox.edu, George Fox University, 414 N. Meridian St, Newberg, OR 97132.
Will Mitchell, Associate Treasurer, (317)392-3038, willmitchell@acm.org, 1455 S. Greenview Ct, Shelbyville, IN 46176-9248.
John Meinke, Associate Editor, meinkej@acm.org, UMUC Europe Ret, German Post: Werderstr 8, D-68723 Oftersheim, Germany, ph 011-49-6202-5777916.
Shereen Khoja, Comptroller, (503)352-2008, shereen@pacificu.edu, MSC 2615, Pacific University, Forest Grove, OR 97116.
Elizabeth Adams, National Partners Chair, adamses@jmu.edu, James Madison University, 11520 Lockhart Place, Silver Spring, MD 20902.
Megan Thomas, Membership System Administrator, (209)667-3584, mthomas@cs.csustan.edu, Dept. of Computer Science, CSU Stanislaus, One University Circle, Turlock, CA 95382.
Deborah Hwang, Webmaster, (812)488-2193, hwang@evansville.edu, Electrical Engr. & Computer Science, University of Evansville, 1800 Lincoln Ave., Evansville, IN 47722.

CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

Platinum Partner

Turingscraft
Google for Education
GitHub
NSF – National Science Foundation

Silver Partners

zyBooks

Bronze Partners

National Center for Women and Information Technology
Teradata
Mercury Learning and Information
Mercy College

Foreword

The following five CCSC conferences will take place this fall.

Midwestern Conference	October 4-5, 2019
	Benedictine University in Lisle, IL
Northwestern Conference	October 4–5, 2019
	Pacific University, Forest Grove, OR
Rocky Mountain Conference	October 11-12, 2019
	University of Sioux Falls in Sioux Falls, SD
Eastern Conference	October 25-26, 2019
	Robert Morris University in Moon Township, PA
Southeastern Conference	October 25-26, 2019
	Auburn University in Auburn, AL

The papers and talks cover a wide variety of topics that are current, exciting, and relevant to us as computer science educators. We publish papers and abstracts from the conferences in our JCSC journal. You will get the links to the digital journals in your CCSC membership email. You can also find the journal issues in the ACM digital library and in print on Amazon.

Since this spring we have switched to Latex for final manuscript submission. The transition has been smooth. Authors and regional editors have worked hard to adapt to the change, which made my life a lot easier.

The CCSC board of directors have decided to deposit DOIs for all peer-reviewed papers we publish. With the DOIs others will be able to cite your work in the most accurate and reliable way.

Baochuan Lu
Southwest Baptist University
CCSC Publications Chair

Welcome to the 2019 CCSC Southeastern Conference

Welcome to the 33rd Southeastern Regional Conference of the Consortium for Computing Sciences in Colleges. The CCSC:SE Regional Board welcomes you to Auburn, AL, the home of Auburn University. The conference is designed to promote a productive exchange of information among college personnel concerned with computer science education in the academic environment. It is intended for faculty as well as administrators of academic computing facilities, and it is also intended to be welcoming to student participants in a variety of special activities. We hope that you will find something to challenge and engage you at the conference!

The conference program is highlighted with a variety of sessions, such as engaging guest speakers, workshops, panels, student posters, faculty posters, a nifty assignment session and several sessions for high quality refereed papers. We received 25 papers this year of which 15 were accepted to be presented at the conference and included in the proceedings – an acceptance rate of 60%.

Two exciting activities are designed specifically for students – a research contest and an undergraduate programming competition, with prizes for the top finishers in each.

We especially would like to thank the faculty, staff, and students of Auburn University for their help in organizing this conference. Many thanks also to the CCSC Board, the CCSC:SE Regional Board, and to a wonderful Conference Committee, led by Conference Chair Dr. Richard Chapman. Thank you all so much for your time and energy.

We also need to send our deepest appreciation to our partners, sponsors, and vendors. Please take the time to go up to them and thank them for their contributions and support for computing sciences education – CCSC:SE National Partners: Turing's Craft, Google for Education, GitHub, National Science Foundation, Codio, zyBooks, National Center for Women and Information Technology, Teradata University Network, Mercury Learning and Information, Mercy College. Sponsoring Organizations: CCSC, ACM-SIGCSE, Upsilon Pi Epsilon.

We could not have done this without many excellent submissions from authors, many insightful comments from reviewers, and the support from our editors Baochuan Lu and Susan Dean. Thanks to all of you for helping to create such a great program.

We hope you enjoy the conference and your visit to Auburn University.

Kevin Treu, CCSC:SE Regional Board Chair, Furman University
John Hunt, Program Chair, Covenant College

2019 CCSC Southeastern Conference Steering Committee

Kevin Treu, Conference Chair	Furman University
Richard Chapman, Site chair	Auburn University
John Hunt, Program Chair	Covenant College
Jean French, Local Registrar	Coastal Carolina University
Chris Healy, Student Research Contest Director	Furman University
Nadeem Hamid, Nifty Assignments Co-Chair	Berry College
Steven Benzel, Nifty Assignments Co-Chair	Georgia Highlands College
Andy Digh, Programming Competion Co-Director	Mercer University
Chris Healy, Programming Competion Co-Director	Furman University

Regional Board — 2019 CCSC Southeastern Region

Kevin Treu, Board Chair	Furman University
Kevin Treu, Board Representative	Furman University
Jean French, Registrar	Coastal Carolina University
John Hunt, Treasuer	Covenant College
Laurie Patterson, Secretary	University of North Carolina Wilmington
Susan Dean, publicity chair	retired
John Hunt, Regional Editor	Covenant College

Reviewers — 2019 CCSC Southeastern Conference

Ali, Farha	Lander University, Greenwood, SC
Allen, Robert	Mercer University, Macon, GA
Alvin, Chris	Furman University, Greenville , SC
Angel, N. Faye	Ferrum College, Ferrum, VA
Besmer, Andrew	Winthrop University, Rock Hill, SC
Bogert, Kenneth	University of North Carolina Asheville, Asheville, NC
Bonyadi, Cyrus	University of Maryland, Baltimore College, Longwood, FL
Bowe, Lonnie	Concord University, Princeton, WV
Carl, Stephen	Sewanee: The University of the South, Chattanooga, TN
Dannelly, Stephen	Winthrop University, Rock Hill, SC
Dekhane, Sonal	Georgia Gwinnett College, Lawrenceville, GA
Digh, Andy	Mercer University, Macon, GA
Drawert, Brian	University of North Carolina Asheville, Asheville, NC
Dumas, Joe	University of Tennessee at Chattanooga, Chattanooga, TN
Elliott, Robert A.	Southern University at New Orleans, New Orleans, LA
Garrido, Jose	Kennesaw State University, Marietta, GA
Gaspar, Alessio	University of South Florida, Lakeland, FL
Glass, Michael	Valparaiso University, Valparaiso, IN
Goddard, Wayne	Clemson University, Clemson, SC
Heinz, Adrian	Georgia Gwinnett College, Lawrenceville, GA
Holliday, Mark	Western Carolina University, Cullowhee,, NC
Hong, Gongbing	Georgia College and State University, Milledgeville, GA
Hutchings, Dugald	Elon University, Elon, NC
Lartigue, Jonathan	Collins Aerospace, Cedar Rapids, IA
Lee, Gilliean	Lander University, Greenwood, SC
Lee, Ingyu	Troy University, Troy, AL
Lewis, Adam	Athens State University, Athens, AL
Li, Rao	University of South Carolina Aiken, Aiken, SC
Lindoo, Ed	Nova Southeastern University, Fort Lauderdale-Davie, FL
Liu, Yi	Georgia College State Universi, Milledgeville, GA
Lutz, Robert	Georgia Gwinnett College. Lawrenceville, GA
Lux, Thomas	Roanoke College, Ashland, VA
McGuire, Timothy	Texas A&M University, College Station, TX
Murray, Meg	Kennesaw State University, Kennesaw, GA
Patterson, Brian	Oglethorpe University, Atlanta, GA
Pittarese, Tony	East Tennessee State University, Johnson City, TN
Plank, James	University of Tennessee, Knoxville, TN
Pounds, Andrew	Mercer University, Macon, GA
Spurlock, Scott	Elon University, Elon, NC
Walker, Aaron	University of North Georgia, GA

Introduction to Jetstream: A Research and Education Cloud[*]

Conference Tutorial

Sanjana Sudarshan and Jeremy Fischer
Research Technologies
Indiana University
Bloomington, IN 47401
{ssudarsh, jeremy}@iu.edu

1 Introduction

Jetstream is the first production cloud funded by the National Science Foundation (NSF) for conducting general-purpose science and engineering research as well as an easy-to-use platform for education activities. Unlike many high-performance computing systems, Jetstream uses the interactive Atmosphere graphical user interface developed as part of the iPlant (now CyVerse) project and focuses on interactive use on uni-processors or multiprocessors. This interface provides for a lower barrier of entry for use by educators, students, practicing scientists, and engineers. A key part of Jetstream's mission is to extend the reach of the NSF's eXtreme Digital (XD) program to a community of users who have not previously utilized NSF XD program resources, including those communities and institutions that traditionally lack significant cyberinfrastructure resources. One manner in which Jetstream eases this access is via virtual desktops facilitating use in education and research at small colleges and universities, including Historically Black Colleges and Universities (HB-CUs), Minority Serving Institutions (MSIs), Tribal colleges, and higher education institutions in states designated by the NSF as eligible for funding via the Established Program to Stimulate Competitive Research (EPSCoR).

While cloud resources won't replace traditional HPC environments for large research projects, there are many smaller research and education projects that would benefit from the highly customizable, highly configurable, programmable cyberinfrastructure afforded by cloud computing environments such as Jetstream. Jetstream is a Infrastructure-as-a-Service platform comprised of two geographically isolated clusters, each supporting hundreds of virtual machines and data volumes. The two cloud systems are integrated via a user-friendly web application that provides a user interface for common cloud computing operations, authentication to XSEDE via Globus, and an expressive set of web service APIs.

Jetstream enables on-demand access to interactive, user-configurable computing and analysis capability. It also seeks to democratize access to cloud capabilities and promote shareable, reproducible research. This event will describe Jetstream in greater detail, as well as how its unique combination of hardware, software, and user engagement support the "long tail of science." This tutorial will describe Jetstream in greater detail, as well as how its unique combination of hardware, software, and user engagement support the "long tail of science." Attendees will get a greater understanding of how Jetstream may enhance their education or research efforts via a hands-on approach to using Jetstream via the Atmosphere interface.

2 Tutorial Description

This tutorial requires two to three hours.

- Prerequisites: Basic Linux command line knowledge a plus (but not required)
- Required: Laptop, modern web browser (Chrome, Firefox, Safari)
- Targeting: Educators, Researchers, Campus Champions/ACI-Ref Facilitators, Campus research computing support staff

This tutorial will first give an overview of Jetstream and various aspects of the system. Then we will take attendees through the basics of using Jetstream via the Atmosphere web interface. This will include a guided walk-through of the interface itself, the features provided, the image catalog, launching and using virtual machines on Jetstream, using volume-based storage, and best practices.

We are targeting users of every experience level. Atmosphere is well-suited to both HPC novices and advanced users. This tutorial is generally aimed at those unfamiliar with cloud computing and generally doing computation on laptops or departmental server resources. While we will not cover advanced topics in this particular tutorial, we will touch on the available advanced capabilities during the initial overview.

3 Tutorial Program

This is a sample tutorial program. Time required for this tutorial is approximately 3 hours.

- What is Jetstream?
- Q & A and what brief hands-on overview
- Getting started with Jetstream, including VM launching
- Break
- Accessing your VM, creating and using volumes
- Customizing and saving images, DOIs
- Cleaning up
- Final Q & A

Using Eclipse and IntelliJ with Dynamic Viewers for Program Understanding and Debugging in Java[*]

Conference Tutorial

James H. Cross II and T. Dean Hendrix
Computer Science and Software Engineering
Auburn University
Auburn, AL 36849
{crossjh,hendrtd}@auburn.edu

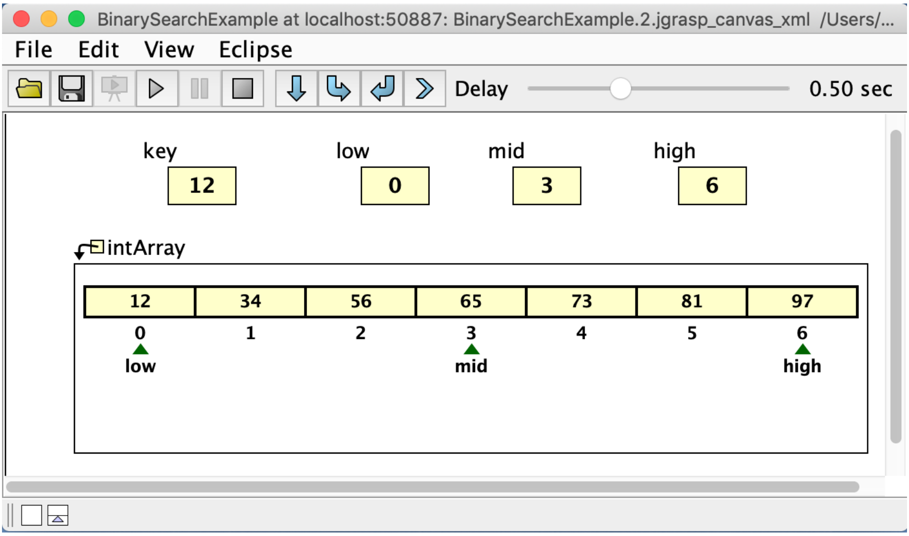
New jGRASP plugins for Eclipse and IntelliJ bring the jGRASP viewers and viewer canvas to the Eclipse and IntelliJ Java debuggers. The plugins provide automatic generation of visualizations that directly support the teaching of major concepts, including classes, interfaces, objects, inheritance, polymorphism, composition, and data structures. The integrated visualizations are intended to overcome the mismatch between what we want to teach and what most IDEs provide in the way of support for learning. This tutorial will focus on the canvas of dynamic viewers which allows students and instructors to create "custom" program visualizations by dragging viewers for any primitive or object onto the canvas and then saving it. Participants are encouraged to bring their own computers with programs from their courses. jGRASP and the plugins are freely available.

All educators who teach Java will benefit from this tutorial. However, it will be especially suitable for instructors who teach Java-based programming, data structures, or algorithms courses. The overall objective of the tutorial is to introduce faculty to the advanced pedagogical features provided by the viewers and canvas for teaching and learning Java. The participants will be guided through numerous scenarios to see how creating visualizations of their programs and making them available to students can make learning to program a more enjoyable experience. In addition to finding the visualizations useful for understanding example programs, students can easily create visualizations of their own programs which will be especially useful while debugging.

[*] Copyright is held by the author/owner.

Since the canvas can be populated with any primitives or objects created by their programs, including traditional data structures (e.g., stacks, queues, lists, and binary trees), the visualizations created by faculty and students are only limited by their creativity. As they "play" or step through their programs in debug mode, all viewers on the canvas are updated dynamically to provide the opportunity for a much clearer understanding of the program.

Consider the following examples which contain multiple viewers on each canvas. Figure 1 shows a canvas in Eclipse for a simple binary search program, which includes five viewers: key, low, mid, high, and intArray. These were created by simply dragging the variables from the debug window or details pane in Eclipse onto the canvas. For the array viewer on intArray, the user has added the variables for the indices which will move along the array as their values change.


BinarySearchExample at localhost:50887: BinarySearchExample.2.jgrasp_canvas_xml /Users/...
File
Edit
View
Eclipse
intArray

Delay 0.50 sec

key 12

low 0

mid 3

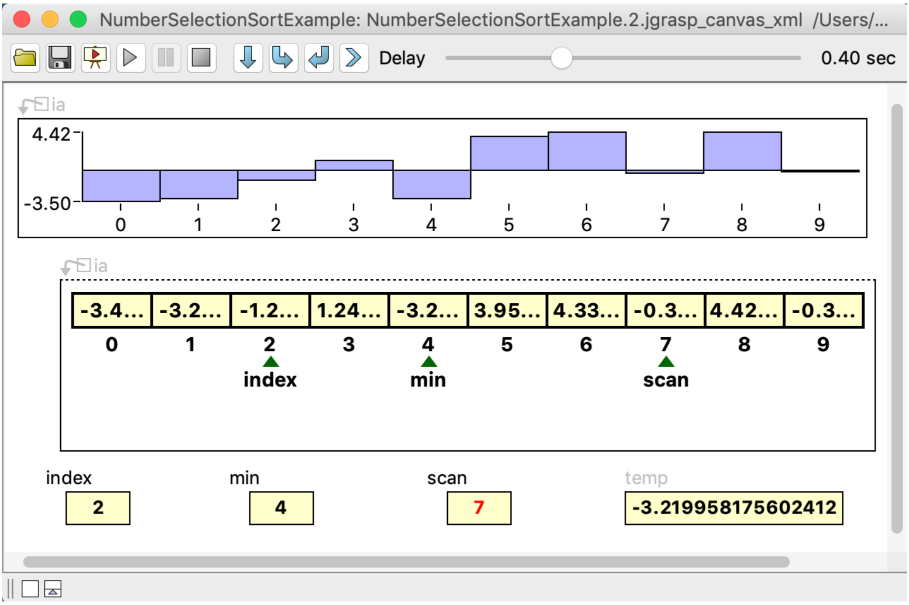
high 6

12	34	56	65	73	81	97
0	1	2	3	4	5	6
low			mid			high

Figure 1: Canvas in Eclipse for a simple binary search program.

Figure 2 shows a canvas in IntelliJ for an implementation of selection sort, which includes six viewers: two on the array ia (one as a bar graph and the other as typical "textbook" presentation), index, min, scan, and temp. The bar graph viewer and the presentation array viewer update automatically as the program runs in the debugger. The bar graph makes it easy to see which value will be the next min for a given iteration through the array.

In each of these examples, the user can simply click the play button on the canvas to auto step through the program, which brings the canvas to life with



NumberSelectionSortExample: NumberSelectionSortExample.2.jgrasp_canvas_xml /Users/...

Delay 0.40 sec

-3.4...	-3.2...	-1.2...	1.24...	-3.2...	3.95...	4.33...	-0.3...	4.42...	-0.3...
0	1	2	3	4	5	6	7	8	9
		index		min			scan		

index 2

min 4

scan 7

temp -3.219958175602412

Figure 2: Canvas in IntelliJ for an implementation of selection sort.

an animated visualization of the program. Since the canvas can be saved, instructors can provide program visualizations with their examples for students, or students can create visualizations to help them understand their own programs and even submit the visualizations as part of their assignments. The canvas of dynamic viewers makes creating visualizations for explaining your own programs quick and easy, and it makes debugging programs almost fun.

The jGRASP IDE and the plugins for Eclipse and IntellliJ are freely available at the jGRASP web site (`https://www.jgrasp.org`). jGRASP and the plugins each include a complete set of examples, including the two in Figures 1 and 2 above.

Building and Expanding a Successful Undergraduate Research Program*

Panel Discussion

Sarah Heckman[1], *Brandon Fain*[2], *Manuel Pérez-Quiñones*[3]
[1]*NC State University*
sarah_heckman@csc.ncsu.edu
[2]*Duke University*
btfain@cs.duke.edu
[3]*University of North Carolina at Charlotte*
perez.quinones@uncc.edu

Undergraduate research is an important means of engaging computer science students outside of the classroom in substantive and original inquiry into the discipline, and to prepare them for independent work in industry or graduate school. We discuss the approaches and challenges of starting, managing, and expanding undergraduate research programs in computer science departments. The presentation should be of interest to faculty developing an undergraduate research program in their department.

During the panel, we will discuss *program contexts* and how that informs decisions about what type of undergraduate research program that may be created and the support structures available for undergraduate students. *Program structure* informs how students connect with faculty, the scope of an undergraduate research project, and what students receive for their work. Additional considerations on *recruitment and admission* into undergraduate research programs should be considered by departments as they think about how to support and grow a program; students may not know undergraduate research is an option. Many students may not know that undergraduate research is an option. Once students are part of a program, *expectations* for success and completion are critical to ensure a good experience. Students may be expected to write a proposal about their work before the project starts, present their work at a poster sessions locally or at the state (e.g., the State of North Carolina Undergraduate Research and Creativity Symposium), national (e.g., National Conference on Undergraduate Research), and international levels, supporting retention in computing [3]. Finally, there are *extensive resources* for supporting undergraduate research. For example, Affinity Research Groups [1] provide a model for creating research teams.

Sarah Heckman is an Associate Teaching Professor and Director of Undergraduate Programs for the Department of Computer Science at NC State University. She oversees the CSC Honors Program which requires an undergraduate research component.

Brandon Fain is an Assistant Research Professor at Duke University. He built an undergraduate summer research program at Duke piloted during 2019 based on a collaboration with similar undergraduate summer programs in data science and software engineering at Duke University.

Manuel Pérez-Quiñones is a Professor at University of North Carolina – Charlotte. In the late 90s, Dr. Pérez-Quiñones was director of the Industrial Affiliates Program[1] at the University of Puerto Rico Mayaguez. The IAP program [2] just celebrated 30 years. In 2002, together with Dr. Scott McCrickard, they started the Virginia Tech Undergraduate Research in Computer Science program[2]. This year's poster session was the 18th iteration of the program. From 2006 until 2010, Dr. Pérez-Quiñones was co-chair of the CREU program[3] as part of the CRA-W/CDC Broadening Participation in Computing Alliance.

References

[1] Ann Gates, Steve Roach, Elsa Villa, Kerrie Kephart, Connie Della-Piana, and Gabriel Della-Piana. *The Affinity Research Group Model: Creating And Maintaining Effective Research Teams.* IEEE Computer Society Press, 2008.

[2] M. Velez-Reyes, M. Perez-Quinones, and J. Cruz-Rivera. The industrial affiliates program at the university of puerto rico - mayaguez. In *Proceedings Of the 1999 Frontiers In Education Conference*, FIE 1999, pages 13C5/13–13C5/18. IEEE, 1999.

[3] Heather M. Wright and N. Burçin Tamer. Can sending first and second year computing students to technical conferences help retention? In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 56–62, New York, NY, USA, 2019. ACM.

[1] https://ece.uprm.edu/iap/

[2] https://www.vturcs.cs.vt.edu/

[3] https://cra.org/cra-w/creu/

A Comparison of Two Popular Machine Learning Frameworks[*]

Chance Simmons and Mark A. Holliday
Department of Mathematics and Computer Science
Western Carolina University
Cullowhee, NC 28723
ltcas97@gmail.com holliday@wcu.edu

Abstract

Using artificial neural networks is an important approach for drawing inferences and making predictions when analyzing large and complex data sets. TensorFlow and PyTorch are two widely-used machine learning frameworks that support artificial neural network models. We evaluated the relative effectiveness of these two frameworks to model a binary classification problem. The binary classification was done using sentiment analysis on a publicly-available data set of product reviews. We first implemented the same model in the same testing environment to see if we were able to achieve similar accuracy with both frameworks. We then compared the training time, memory usage, and ease of use of the two frameworks.

1 Introduction

Artificial neural networks (ANNs) [4] have been demonstrated to be effective for many cases of supervised learning [6], but programming an ANN manually can be a challenging task. Frameworks such as TensorFlow and PyTorch have been created to simplify the creation and use of ANNs.

One of the major uses of artificial neural networks is natural language processing[5] one aspect of which is sentiment analysis. To compare the two machine learning frameworks, the first step was to develop, train, and evaluate the same neural network model in both frameworks. In theory we should be able to obtain the same accuracy in both frameworks. since the same underlying model was being implemented. The second step was to compare the model implementations in the two frameworks based on execution time, memory usage, and ease of development.

2 Data Set

The data set that was chosen to test the frameworks was a publicly-available set of Amazon reviews for video games[1]. The ratings that the individual gave were also included in the data set. Due to the nuances and bias involved in what each individual feels a certain rating should be, the data set was then broken down into only positive and negative reviews. The positive reviews consisted of the reviews with ratings of 4 or 5, whereas the negative reviews consisted of the reviews with ratings 1, 2, or 3. By having only two categories, the problem then becomes a sentiment analysis problem that uses binary classification.

Neural networks use mathematical calculations, so the textual reviews needed to be converted into numerical information. In this case, the text was analyzed to find the most common 10,000 words. Each occurrence of each word in the textual reviews was then replaced by the numerical index of that word in the common word list if that word occurred in the common word list. Any word that were not common enough to be found in the common word list was given the value of 0. Below is an example sentence from the Amazon reviews and the corresponding tokenization.

['Dirt 3 on DVDi collect racing games so had to add this to my collectionSon wated one also']

[98 19 908 496 34 30 80 3 408 11 3 39 31 0 0 0 0 0 0 0 . . . (0x230)]

Each sentence is reduced to a total size of 250 indices. This number was chosen by taking the average length of all the sentences in the data set. Any sentence over 250 words used just the first 250 words found in the vocabulary. If a sentence is shorter than 250 words, then the rest of list is padded with 0's. In the above example, there were only 13 words that were found to be in the vocabulary. This meant that 13 words in the original sentence were converted to their numerical representation and the rest of the list was filled with 0's.

This sentence highlights some of the issues that were found with the data set. Some of the user's reviews included grammatical errors. These errors made it so that those words were not common enough to be included in the final sentence, removing some of the important information. In this case, words like 'DVD', 'collection', 'Son' and 'wanted' are left out from the tokenized sentence because of errors present in the review.

3 Model

In a recurrent neural network (RNN) [3] the output of the RNN cell is fed back into the recurrent network cell as input, allowing for sequences of information to be learned. Since the words that occur before a certain word in a sentence add importance to the current word being analyzed, RNNs are often used in natural language processing. So we used a RNN instead of a simple feed-forward fully-connected neural network.

3.1 Input Layer

The input of the model consist of 32 nodes that are a part of the embedding layer. The embedding layer takes the list of 200 numbers representing the review sentence, and changes them into vector representations that are stored in a list of size 32. The main benefit of using the embedding layer is to cut back on the size of the input list that is being passed into the neural network. Another benefit of embedding layers is that they offer another layer of training. As the inputs are passed in, the embedding will begin to learn the words that are similar in meaning and group them together so they are given similar numbers in the resulting vector list.

3.2 Hidden Layer

There are also 32 nodes in the hidden layer of the tested model. These nodes represent Long Short Term Memory (LSTM) cells. LSTM cells use a memory cell that can maintain its states over time in combination with gates that regulate the information that is going into and out of the cell [3]. These cells make up the recurrent part of the network. The benefit of using LSTM cells over normal RNN cells is that more information of previous sentence structure and words is kept for a longer period of time.

3.3 Output Layer

The output layer is made up of one node. There is only one node in the output because the problem that is being solved is a binary classification problem. The activation function on this node is the sigmoid activation function. The sigmoid activation function will convert the number being passed into the output into a value between 0 and 1. This value is then rounded up/down to get a overall value of 0 or 1. The final output value is compared to the optimal value in order to determine the accuracy of the neural network model.

4 Training

We used Google Colaboratory as the testing environment since supports both frameworks and Python as the language. To maximize the performance of both implementations, we enabled use of the Graphics Processing Units (GPUs).

We used the Adam optimizer in both implementations. The Adam optimizer is a basic optimizer that uses gradient descent and a momentum factor to perform back propagation. Back propagation is the process of adjusting the weights of the links between the nodes in order for the network to become more accurate when similar input is passed into it. The momentum factor is used to change the links weights at a higher rate whenever the same links are being changed constantly.

Over-fitting is a serious issue when training a neural network. Over-fitting occurs whenever the network's training becomes so specific to the training data that its predictions for other data become less accurate. After extensive testing we were able to obtain the best accuracy on new data for both frameworks by using 20 epochs, a dropout of fifty percent, a learning rate of 0.01, a batch size of 1000, and a hidden layer size of 32 nodes. Dropout means that in the training of the neural network during each epoch a random and usually different 50 percent of the nodes in a layer would not be considered.

5 Results

5.1 Accuracy

The TensorFlow Accuracy graph (Figure 1) and the PyTorch Accuracy graph (Figure 2) indicate how close the accuracies of the two frameworks are. The training accuracy in both models are constantly increasing; this is due to the fact that the models are starting to memorize the information that they are being trained on. The validation accuracy indicates how well the model is actually learning through the training process. In both cases, the validation accuracy of the models in both frameworks averaged about 78% after 20 epochs. Clearly both frameworks were able to implement the neural network accurately and are capable of producing the same results given the same model and data set to train on.

5.2 Training Time and Memory Usage

The TensorFlow Training Time graph (Figure 1) and the PyTorch Training Time graph (Figure 2) indicate that the training time for TensorFlow is substantially higher (average of 11.1954 seconds while PyTorch's average was

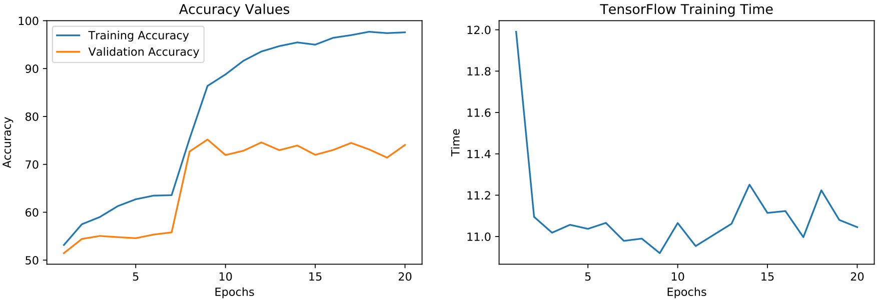


Figure 1: TensorFlow Accuracy and Training Time

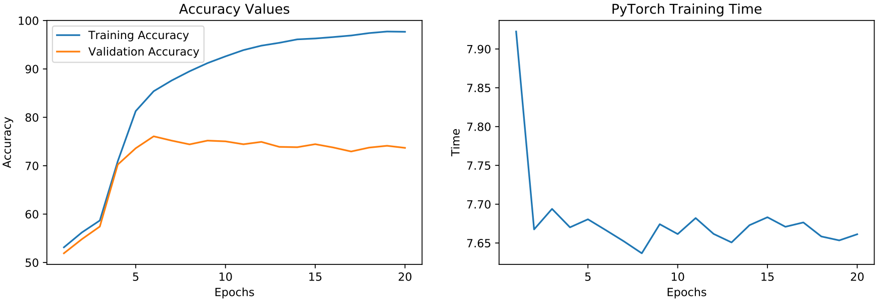


Figure 2: PyTorch Accuracy and Training Time

7.6798 seconds). The durations of the model training times can vary substantially from day to day on Google Colaboratory. However, the relative durations between TensorFlow and PyTorch remain consistent.

TensorFlow had a lower memory usage during training (1.7 GB of RAM while PyTorch's memory usage was 3.5 GB); both had little variance in memory usage during training. Both had higher memory usage (4.8 GB for TensorFlow and 5 GB for PyTorch) during the initial loading of the data.

5.3 Ease of Use

PyTorch's more object-oriented style made implementing the model less time-consuming and the specification of data handling more straightforward. TensorFlow, on the other hand, had a slightly steeper learning curve due to the low level implementations of the neural network structure. The TensorFlow low level approach allows for a more customized approach to forming the neural network which allows implementing more specialized features. The very high level Keras library runs on top of TensorFlow. So as a teaching tool, the very high level Keras library[2] can be used to teach basic concepts, and then Tensorflow can be used to further the understanding of the concepts by having to lay out more of the structure.

6 Conclusions

TensorFlow and PyTorch showed equal accuracy in our experiments. TensorFlow's training time was substantially higher, but its memory usage was lower. PyTorch allows quicker prototyping than TensorFlow, but TensorFlow may be a better option if custom features are needed in the neural network. Our model implementations and data set are available at https://github.com/Ltcas/NLPFrameworkComparison.

Comparing PyTorch to the recently released TensorFlow 2.0 as well as to using the Keras library is possible future work.

References

[1] Amazon reviews data sets. `https://snap.stanford.edu/data/web-Amazon.html`. Accessed: 2018-12-14.

[2] Francois Chollet. *Deep Learning with Python.* Manning, 2017.

[3] Aurelien Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, Sebastopol, CA, 2017.

[4] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[5] Delip Rao and Brian McMahah. *Natural Language Processing with PyTorch.* O'Reilly Media, Sebastopol, CA, 2019.

[6] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach Third Edition.* Pearson Education, 2010.

Alexa Skill Voice Interface for the Moodle Learning Management System[*]

Michelle Melton and James Fenwick Jr.
Department of Computer Science
Appalachian State University
Boone, NC 28608
{meltonml,fenwickjb}@appstate.edu

Abstract

Most educational and training organizations today use some type of learning management system (LMS) to make course material available online to participants. An LMS can be used for face-to-face, fully online, or hybrid courses incorporating versions of both. Learning management system users want easy and fast access to learning materials. LMS access is typically provided through an online interface or a mobile application, both of which require the use of touch and sight on a computer or device. With the rapid growth of technology advancements and user knowledge, LMS users will expect faster and more convenient access.

The last decade has brought considerable progress in voice technology. Significant improvement in the accuracy of speech to text translation has made the use of voice-enabled devices more common. Since both technology and usage are continuing to grow, voice interfaces will become even more important for modern applications.

Two of the top three LMS frameworks on the market today have voice interfaces. Both Blackboard Learn and Canvas by Instructure have Amazon Alexa skill integrations that provide basic course information such as announcements, assignments, and grades. Presently, there is no distributed voice integration for Moodle, the second-ranked LMS provider.

This paper details the development of a voice user interface for a learning management system: specifically, an Amazon Alexa skill for Moodle. The research thoroughly outlines the process of developing an Alexa skill for Moodle, including:

- user-centered interface design;
- developing effective prototypes for early feedback on the design;
- usage of the Alexa Skills Kit for the front-end development of the skill;
- implementing the Moodle API for the development of the back-end web service for the skill; and
- planning and conducting effective usability testing sessions and evaluating results.

An Alexa skill integration with Moodle will allow users to more quickly and conveniently access information from the LMS. Immediate benefits of the project include providing site announcements to all users, course announcements to students and teachers, and overall course grades and upcoming due dates to students. In the future, the application may be expanded to implement instructor capabilities like getting a list of assignments that need grading and the ability to create voice activities for students. Future development may also include providing additional course content for students, such as attendance, missing assignments, and instructor contact information.

1 Introduction

Today, most educational and training organizations make at least some of their course content available on a learning management system (LMS). Whether the courses are online, face-to-face, or hybrid, an LMS makes assignments, grades, and other course material available online.

Students rank easy access and fast access to learning materials as second and third in importance for an LMS [3]. Most learning management systems attempt to meet these needs with an online interface as well as some type of mobile access. With both innovations in technology and user savviness growing rapidly, LMS users will want even faster and more convenient access to course material than the online and mobile interfaces can provide.

Historically, the biggest challenge for voice interfaces (spoken interactions with a computer) was the accurate translation of speech to text [5]. Modern voice technology has improved significantly in the past decade; the speech recognition error rate is now only about 8% [9]. With such a dramatic improvement in the usability of voice-enabled devices, they are becoming more commonplace. In fact, 20% of Google searches are now performed by voice [8].

Many popular applications have already started integrating voice interfaces, including some LMS frameworks. In 2017, Blackboard Learn and Canvas by Instructure, two of the top three learning management systems, implemented Amazon Alexa skills that provide standard course content like announcements, assignments, and grades. There is currently no distributed voice integration for Moodle, the second-ranked LMS provider.

This paper describes the development of an Amazon Alexa skill that enhances the speed and convenience of accessing information in the Moodle LMS. Current features include providing all users access to public site announcements and enabling student access to course announcements, grades, and upcoming due dates. Future development may expand functionality to include instructor actions, such as accessing assignments that need grading and possibly even creating voice interactive assignments, as well as expanding the content available to students.

2 Alexa Skill

Similar to Google Home and Apple's Siri voice assistant, Alexa is Amazon's cloud-based voice service available on Alexa devices. Alexa skills are apps that enable voice-activated capabilities for connected smart devices and online services. Users interact with Alexa by saying a wake word to wake the device and then speaking an invocation phrase that consists of an utterance and the invocation name of the skill. For example, "Alexa, ask Daily Horoscopes for the horoscope for Gemini" consists of the "Alexa," wake word, the "Daily Horoscopes" invocation name, and "the horoscope for Gemini" utterance.

3 Voice Interface Design

General interface design principles can and should be applied to creating voice applications, but a few characteristics of voice user interfaces (VUIs) require special consideration in their design. Auditory interactions differ from visual ones in that they present information one word at a time, the information is constantly changing, and there is no permanent record of what was said [5]. These unique characteristics can place cognitive demands on users by requiring them to use short-term memory and to move at a predetermined pace [4]. It is important to take these cognitive issues into account during the design of the voice interface.

Due to the differences between visual and voice interfaces, standard prototyping for user feedback early in the design process has to be modified for voice interactions. The interaction layer (the dialog and responses of the system) and the presentation layer (the voice, word choice, and speaking rate of the system) are more connected in a voice application, so both should be included in prototypes. Prompts should be fully scripted for the interaction layer, so the user's ability to complete a task is not impaired. The production voice should be used because pitch and pace (the personality of the system) can affect a user's evaluation of the interface [5].

The design process ensured careful consideration of the purpose and capabilities of the skill, what users would say when interacting with the skill, and planning for how Alexa would respond to build a voice interface that provides value and is easy to use.

The process began by identifying user stories for the skill. To determine the capabilities users would find most beneficial, reports from Google Analytics for AsULearn (the Appalachian State University instance of Moodle) were examined to verify the most viewed pages. This data helped inform the decisions of the initial intents for the skill: GetSiteAnnouncementsIntent, GetCourseAnnouncementsIntent, GetDueDatesIntent, and GetGradesIntent.

With the user story intents established, the way users will speak their intentions needed to be considered, which involved outlining the utterances for each intent. To ensure that the invocation phrases considered actually match the words students might use, students completed a basic survey about their preferences for the phrasing of the application name, courses, announcements, grades, and due dates. These results helped guide the design of the Alexa skill in terms of the invocation name, the way courses were spoken to the user and to the skill, and the implementation of additional utterances for each intent.

The last step in the design process was planning how Alexa would respond to user requests. Formatting the responses so they sound natural took priority over using proper grammar to make sure Alexa sounds like a person when a user is interacting with the skill [1]. Responses that need an answer from the user were designed to end with a prompting question to serve as a cue for the user to begin speaking. Multiple variations of responses were designed for each intent, and acknowledgments such as "thanks," "okay," and "great" were planned for inclusion to make the interaction more conversational [1].

Another element of the design focused on adding a layer of access protection to the skill. To address privacy concerns, the design incorporated the ability to set an optional PIN during account linking that can be used to verify the user before personal information is returned.

4 Alexa Skill Architecture

4.1 Front-end

The front-end of the voice user interface for a custom Alexa skill is created in the Alexa Skills Kit (ASK) developer console. Building the interaction model involves configuring the invocation name, intents, sample utterances, and slot types, which define information that can vary within an utterance and are used to facilitate dialog with the user.

The skill invocation name was set to "as you learn" since this is how users speak the branded name of the Appalachian State University Moodle site.

To enable the primary capabilities of the skill, four ASK intents were created: GetSiteAnnouncementsIntent, GetCourseAnnouncementsIntent, GetGradesIntent, and GetDueDatesIntent. Between 50 and 250 utterances for each intent were added to the interaction model, as Amazon recommends at least 30 utterances per intent to enhance skill performance [1]. Several Alexa built-in intents were also implemented to provide for the processing of standard commands, such as handling the typical ways users end a skill session as well as ask for help.

The Dialog interface in the ASK enables dialog between a user and Alexa. A Dialog directive returned with a skill response lets Alexa know that a user response is needed to complete the processing of a request. Responses are then stored in slots in subsequent requests to Alexa. A custom COURSE slot type was created and populated so users can say the name of a course for which they would like to hear announcements. To handle PIN responses from the user, the AMAZON.FOUR_DIGIT_NUMBER slot type was implemented. This slot type provides built-in recognition of the variety of ways four-digit numbers are spoken, such as “nineteen twenty-one” or “one nine two one”, and sends the digits to the web service for processing [2].

To establish the connection between the Alexa skill front-end and the web service back-end that receives and processes the skill requests, the address of the Moodle web service was input as the endpoint. Account linking was enabled to use OAuth 2.0 implicit grant authorization, and the address of the custom login for the Alexa skill for Moodle plugin was set as the authorization URI.

4.2 Back-end

4.2.1 Web Service Plugin

The custom Alexa Skill plugin for Moodle was developed and coded to serve as the back-end web service endpoint for the skill. Moodle already provides a web service API enabling third-party customization. However, several deviations from the standard API were necessary to adhere to the ASK requirements. The Moodle core web service that custom plugins extend only allows the passing of arguments via URL query strings. In order to receive the JSON documents sent by Alexa, a third-party plugin providing the REST protocol with JSON payload support [7] was forked and customized to meet the requirements. Moodle’s web service API requires that the parameters for the web service be pre-defined in the plugin, which would involve declaring all the JSON request properties in the plugin code. This specification posed a problem because the Alexa Skills Kit states that new properties may be added to the request and response formats, and web service endpoints must not break when receiving requests with additional properties [2]. In order to meet this specification, the RESTALEXA plugin was designed to send the JSON request to the Alexa plugin as a text string.

4.2.2 Skill Linking to Moodle Account

To enable account linking, the Alexa Skills Kit requires that the web service login accept a username, password, state, client ID, response type, and redirect URI. The web service needs to generate and return a token for the specified user, along with the state from the request, to Alexa at the provided redirect URI [2]. The Moodle core token request is similar to the core web service request in that arguments are passed via URL query strings. It also only provides the token in the response. This response structure was not sufficient to meet the ASK requirements, so a custom login and account linking process was created.

A PIN verification option was implemented for users who want an added layer of security for accessing personal information in Moodle from Alexa. The security PIN is useful for Alexa devices in shared living spaces like student apartments. After users login to Moodle via their specified authentication method, they are able to create an optional 4-digit PIN that is stored in Moodle as user data. If the web service receives a request from a user with a linked account and a PIN set, Alexa will prompt for PIN verification before providing user-specific information.

4.2.3 Web Service Processing of Requests

When the web service receives an Alexa skill request, it parses the JSON and calls an internal function for the request type specified. When the web service receives a LaunchRequest, sent when a user opens the skill, it sends a response that includes a welcome message and available options, ending with a prompt for the user's choice. If the Moodle account is linked, the response will be personalized with the user's first name.

For the GetSiteAnnouncementsIntent, the web service will respond with the site announcements from the front page. The number of site announcements retrieved is determined by the front page settings, limited to five for usability.

For the GetCourseAnnouncementsIntent, the web service performs account linking and PIN verification, and the list of courses for which the user has enrollments is retrieved. If there are no courses or if a single course with no announcements is found, these respective messages are returned. If there are announcements for a single course, they are provided. The number of announcements retrieved is determined by the course settings, again limited to five. If more than one course is found for a user, the web service responds with the list of course names and a prompt for the user to select a course. The user's course name response is parsed from the COURSE slot value in the request from Alexa and checked against the list of course enrollments for the user. If a match is found, the announcements for that course are returned.

The GetGradesIntent performs account linking and PIN verification, and a response is returned with the overall course grades for each of the student's courses.

The web service also performs account linking and PIN verification for the GetDueDatesIntent and the course enrollments and group memberships for the user are determined and events retrieved. The number of events returned in the response is determined by the site setting, limited to five for usability. The site setting for number of days in the future to look ahead is also used in the evaluation of returned events.

Responses are randomly chosen from several variations so the user experience is more personal and conversational. Responses also include a reprompt, which Alexa speaks if no response is heard from the user within 8 seconds, or if the response is not understood.

4.2.4 Moodle Plugin Installation

Documentation and installation instructions were created for the web service plugin. A JSON file of the interaction model was also included with the plugin code for quickly building the base skill in the Alexa developer console with the JSON Editor import feature. There are several GUI settings that are automatically configured for the Moodle site administrator on installation, and the plugin includes several configurable settings to facilitate installation and use on any instance of Moodle.

5 Results

The overall objective was to build a voice user interface that enhances the speed and convenience of accessing information in a learning management system. This goal was achieved by implementing an Amazon Alexa skill for the Moodle LMS that provides voice access to site announcements, course announcements, grades, and due dates.

Upon development of the four primary intents for the Alexa skill, usability testing was performed to evaluate the voice application. Students from a variety of different colleges, grade levels, and familiarity with Amazon Alexa skills and AsULearn were recruited to participate.

After using the skill, participants were asked to complete an online survey to rate their experience. The feedback survey was designed and built based on the SUISQ-MR [6]. The four usability factors were distributed across the survey as user goal orientation (questions 1-2), customer service behavior (questions 3-4),

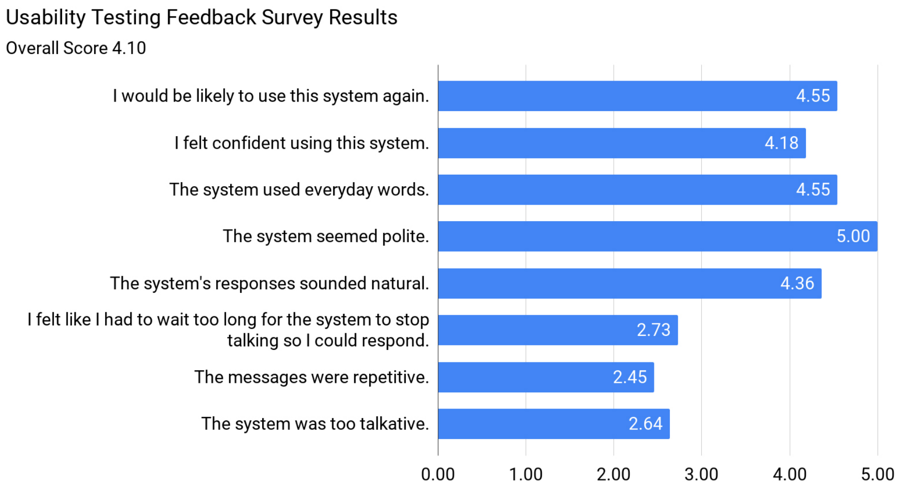


Figure 1: Usability testing survey results.

speech characteristics (question 5), and verbosity (questions 6-8). A 5-point Likert scale was used, with 1 being "Strongly disagree" and 5 being "Strongly agree." Figure 1 shows the results of the survey completed by participants after using the skill.

A follow-up interview was also conducted to get additional feedback. The interview consisted of several open-ended questions to allow participants to discuss their opinion of the skill in greater detail. Participants were asked what they found easy about using the skill, what they found difficult about using the skill, if they encountered anything unexpected during the use of the skill, and if there were other features or capabilities they would find useful to have in the skill. They were also asked about the PIN section of the account linking process; specifically, if it was obvious that it was optional, as well as its purpose.

Interviews revealed that most participants realized the optionality of the PIN after they had already created it during the account linking process, and they assumed its purpose was for an additional layer of access protection. However, they expressed that additional clarity on the account linking form would be helpful. All users reacted positively to the PIN feature.

Comments regarding what was easy as well as difficult about using the skill tended to vary based on the user's familiarity with Alexa. Users who were more familiar with Alexa communicated that the skill was very similar to and even easier to use than other Alexa skills. Those with less Alexa experience discussed difficulty figuring out what they needed to say to use the skill; however, they also indicated that any difficulty with the utterances would be easily overcome with a little practice using the system.

Many of the suggestions for additional features or capabilities for the skill were ideas discovered in previous research, such as the student survey conducted to aid in the design of the skill. Most of these features are already planned for future development work.

All users expressed surprise and delight that Alexa knew and used their name in the response to the LaunchRequest. Personalization of the skill interaction appears to be appreciated and highly valuable for the usability of the interface.

At the end of each testing session, all participants expressed their enjoyment in using the skill and the capabilities it provided, as well as their hope that it will be implemented on AsULearn in production.

6 Conclusion and Future Work

The next step in the project will be to release the skill in production and let a broader audience of students interact with it. The increased usage will enable further usability research, as well as more comprehensive analytics from the Alexa developer console. The additional data will ensure that any proposed changes align with the needs of most of the skill's user base.

The plugin code for the Alexa skill is available on GitHub at `https://goo.gl/jCJGLG`, and the plugin code for the RESTALEXA protocol plugin is available at `https://goo.gl/eMdmBT`.

With the implementation of a few final modifications and enhancements based on usability test feedback, the skill will be submitted to the Appalachian State University Center for Academic Excellence Learning Technology Services team for review before submitting for certification and launch in the Alexa Skills Store.

The initial development of a voice application for accessing information in the Moodle learning management system was the core of this research; however, there are additional, further reaching implications to investigate in the future. Future work may include adding instructor-specific tasks such as the ability to hear a list of assignments that need grading, as well as the ability to create voice activities in Moodle. Allowing students to complete quizzes verbally is a feature that would offer added value for instructors and students alike. Expanding the existing intents would also improve the current capabilities of the skill.

In addition to expanding the functionality of the skill, research on the usability impact would be interesting to explore. The spoken/auditory access to Moodle may enhance the accessibility of the application for users with disabilities. Providing students access to their current performance may also have a positive impact on student success. Research to find out if the increased access to academic status and learning materials afforded by the voice interface positively affects overall student success is another area of interest. With the development of the initial application and usability testing complete, these extended areas of development and research can be explored in the future.

References

[1] Amazon Alexa. *Alexa Voice Design Guide.* World Wide Web electronic publication, https://developer.amazon.com/designing-for-voice.

[2] Amazon Alexa. *Understand Custom Skills.* World Wide Web electronic publication, https://developer.amazon.com/docs/custom-skills/understanding-custom-skills.html.

[3] Lee Yen Chaw and Chun Meng Tang. The voice of the students: Needs and expectations from learning management systems. In *Proceedings.* European Conference on Games Based Learning, 2017.

[4] Michael H. Cohen, Jennifer Balogh, and James P. Giangola. *Voice User Interface Design.* Addison-Wesley, 2004.

[5] Susan L. Hura. Usability testing of spoken conversational systems. *Journal of Usability Studies 12: 155 - 163*, August 2017. http://www.uxpajournal.org/usability-spoken-systems.

[6] James R. Lewis. Standardized questionnaires for voice interaction design. *The Journal of the Association for Voice Interaction Design 1: 1 - 16*, April 2016.

[7] Moodle. *REST protocol (with JSON/XML payload support)*, February 2016. World Wide Web electronic publication, https://moodle.org/plugins/webservice_restjson.

[8] Cathy Pearl. *Designing Voice User Interfaces: Principles of Conversational Experiences.* O'Reilly Media, 2016.

[9] John Rome. Alexa goes to college: Asu's innovative use of voice technology. In *Annual Conference*, 2017.

Auto-Checking Digital Logic Design Labs Through Physical Computing[*]

Gongbing Hong, Gita Phelps, Yi Liu, Kenneth Trussell
Information Systems and Computer Science
Georgia College and State University
Milledgeville, GA 31061
{gongbing.hong,gita.phelps,yi.liu,kenneth.trussell}@gcsu.edu

Abstract

In this paper we introduce a simple and inexpensive solution that auto-checks digital logic design (DLD) labs using Raspberry Pi – a small single board computer with physical computing capability. Given the large number of test cases associated with any typical DLD lab, this work has the benefit of dramatically cutting the amount of manual labor required of an instructor to check DLD lab work. When used by students for self-check, it helps improve learning outcome and experience by providing quick feedback to students.

1 Introduction

This work is motivated by the enormous amount of manual work an instructor has to perform in grading digital logic design (DLD) labs for students. DLD labs often have a significant number of inputs and outputs. The number of test cases grows *exponentially* with the number of digital inputs. For example, for a simple 4-bit adder, the number of inputs is 8 for two 4-bit operands and the number of outputs is 5 for a 4-bit sum and a 1-bit carry-out. The total number of input combinations is thus $2^8 = 256$. For each input combination, one has to check all 5 outputs against the expected result. So the total number of checks is $256 \times 5 = 1280$. Given the size of any typical class, that is undoubtedly labor intensive if the check is done manually. So in reality, instructors often "cut corners" in various ways to reduce the amount of work which in the end can sacrifice the quality of teaching. One solution to this problem is thus auto-checking / grading.

While auto-grading assignments in the teaching of programming related courses is a well established practice, to the best of our knowledge, there has been few simple and inexpensive solutions for the auto-checking / grading of DLD labs. This is likely due to the relative "messiness" in dealing with physical world objects by software tools. But thanks to the recently cheap and commercially available small single-board computers such as Raspberry Pi [5] that readily support physical computing, this issue can now be easily addressed at a very low cost.

In this paper, we attempt to fill a void by presenting a simple and inexpensive solution that extends auto-grading to the field of DLD labs. The remainder of the paper is organized as follows. In Section 2, we review some background information and related work. In Section 3, we introduce our methodology. After that, we present our solution with some discussion in Section 4. In Section 5, we conclude the paper with some future work.

2 Background and Related Work

2.1 Autograding in CS education

Auto-grading involves utilizing automated software tools called *autograders* to check and grade student work automatically. It has been successfully used in checking / grading programming assignments. When a student submits a program to an autograder, the autograder automatically picks a test case, supplies the input from the test case, and runs the program. When the program terminates, its output is automatically compared against the expected result for correctness. The autograder then iteratively tries the next test case until all test cases are exhausted.

Auto-grading is beneficial to students by providing instant feedback about their submissions, which can be used to help correct any mistakes in a timely manner. At a minimum, students will be able to learn almost immediately whether or not their solutions are acceptable. If a submission turns out to be incorrect, a student can try again depending on the settings. Autograders are often set to allow multiple submissions before the deadline of an assignment. This is definitely something extremely useful but hard to do for a human grader. As a result, auto-checking helps enhance the student learning experience – more student work, more effective teaching, and better results. Due to its efficiency, autograding is particularly essential for massive open online courses (MOOCs). It has been reported that autograding helps improve the completion rates in the offerings of MOOCs [6].

There are a variety of autograders available today, both open source and commerical products such as Autolab [9], Submitty [4], and CodeLab [2]. These tools, however, are for the auto-grading of software code only. Our proposed system, on the other hand, will be an auto-checker / grader for hardware-oriented digital logic design labs.

2.2 Current practices in teaching digital logic design

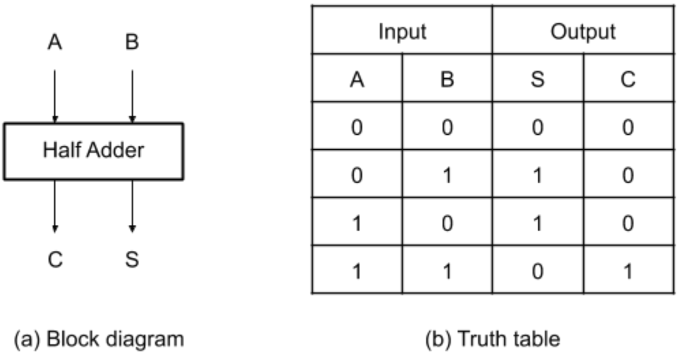
When it comes to teaching DLD, projects may be either simulated or hand-built with tangible IC chips on actual breadboards or both. Commercial software products are available for DLD simulations but are expensive. At our institution, we prefer to use a freely available, light-weight DLD simulation tool called Logisim [1]. Students are instructed to create their design using Logisim and simulate it to eliminate any design issues before they actually implement their design on a breadboard.

While simulation may be considered adequate to some, we have found benefit in having students build and wire the circuits by hand on actual breadboards thereby linking the practical and the theoretical. Tangible learning engages students and the haptic experience concretizes the concepts discussed in class. Constructing circuits can be frustrating because errors can arise from different sources making it difficult to locate and correct hardware bugs. Research has been done in this area to help students by visualizing the states of circuits. Toastboard [3] and CurrentViz [10] are two examples of educational tools used with designing on actual breadboards. Toastboard provides measurement and visualization of voltage and CurrentViz provides measurement and visualization of current on a breadboard. They both rely on custom built breadboards not yet available for widespread use. These tools are more complex than our auto-checker/grader. The DLD auto-checker we propose can be easily reproduced by others using only a Raspberry Pi.

3 Methodology

In this section, we demonstrate how to check the correctness of a simple DLD lab using Raspberry Pi. This example lab asks students to design and implement a half adder that adds two single binary digits A and B to produce two outputs S (sum) and C (carry). Its functionality is given by the block diagram and the truth table in Figure 1.

To auto-check the correctness of the lab work, the binary inputs must be supplied and the outputs must be read and checked against the expected outputs given in the truth table. This can be done through GPIO signal pins of a Raspberry Pi processor. The inputs and the outputs of the half adder can be wired to any four chosen GPIO signal pins. Newer Pi models provide a



Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) Block diagram

(b) Truth table

Figure 1: Half Adder Lab.

40-pin GPIO header with a layout as shown in Figure 2. For example, in the figure, GPIO signal pin 22, which we will simply call GPIO pin 22, is found at physical pin location #15 on the 40-pin GPIO header. As shown, we will use the GPIO pins 27 and 22 for inputs A and B. We will use the GPIO pins 23 and 24 for outputs S and C.

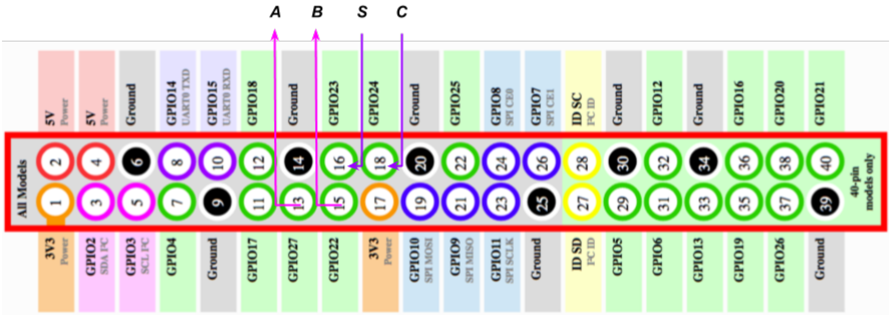


Figure 2: Raspberry Pi GPIO Pin header.

Some of the pins on the GPIO header are labeled 5V/3.3V and Ground. For most digital circuit labs that do not require much electrical power, these pins can be a huge convenience to the user – they can be used to directly power the circuits without issue.

The I/O pins on the GPIO header can be programmed to be either input or output pins. A pin programmed as an output can be programmatically driven to either high (1) or low (0). So an output GPIO pin can be used as a binary input to a digital circuit. A pin programmed as an input can be wired to an output of a digital circuit and read to get the value of the digital output. Such reading can then be used to check against the expected result of a circuit output for correctness. For the pin allocation shown in Figure 2, GPIO pins 27 and 22 should be programmed as output pins to provide digital inputs A and B. GPIO pins 23 and 24 should be programmed as input pins to check digital outputs S and C.

A script is then written to enable the auto-checking of the lab. We can write such script using a variety of scripting/programming languages such as Python and C/C++. Various GPIO driver libraries are available. GPIO Zero [8], a Python package, is one of the easiest. Using this package, each GPIO signal pin can be abstracted into a Python object:

```
# Inputs to the circuit (outputs from Pi)
gateInA = DigitalOutputDevice(27)    # use GPIO pin 27 for A
gateInB = DigitalOutputDevice(22)    # use GPIO pin 22 for B

# Output from the circuit (inputs from Pi)
gateOutC = DigitalInputDevice(24)    # use GPIO pin 24 for C
gateOutS = DigitalInputDevice(23)    # use GPIO pin 23 for S
```

With these Python objects, the following code snippet tests the input combination ($A = 1$ and $B = 0$) and checks the actual output against its expected output ($C = 0$ and $S = 1$):

```
# Set digital input (Pi output)
gateInA.on()    # A = 1
gateInB.off()   # B = 0

# Check digital output (Pi input)
if gateOutS.isActive and gateOutC.value == 0:
    print(`Pass')
else:
    print(`Fail')
```

The complete Python script for auto-checking the half adder DLD lab based on this methodology is available here[1].

4 Solution and Discussion

In the previous section we demonstrated how to use Raspberry Pi to auto-check a DLD lab in an *ad-hoc* fashion. Based on that approach without additional work, one would have to write a new script for each new lab. That is clearly not ideal. Below we consider the problem of writing a generic script that can be used for *any* DLD lab.

We will first present a generic script that works for *any* combinational circuit DLD lab. Due to the complexity of sequential circuits, a generic script

[1] https://drive.google.com/open?id=1gDH_CZjsIClylR-JeSceihJOh6IZ2nNB

is not available at this time. Instead we will choose an example sequential circuit DLD lab and provide an *ad-hoc* solution to it to illustrate some of the characteristics specific to sequential circuits.

4.1 A generic script for any combinational circuit lab

The key to a generic script is to separate the functionality of a specific DLD circuit from the logic of the script. Our solution is that for every DLD circuit lab, the user of the generic script will provide a definition file that describes the functionality of the circuit. The generic script will first parse the definition file and then automatically generate / drive the checks.

Fortunately the functionality of a combinational circuit is quite easy to describe with either *min-term* or *max-term* expressions. For example, for the half adder example above, the outputs S and C can be described in min-terms as below:

$$S = \bar{A}B + A\bar{B} = m1 + m2 = \sum m\{1, 2\}$$

$$C = AB = m3 = \sum m\{3\}$$

The min-term expression of a logic function is quite straightforward and easy to understand. For example, the above min-term expression for S states that, for S to have an output of 1, the input combinations AB will have to be either 01 or 10 in binary (that is 1 or 2 in decimal). The definition file for the half adder lab is given below:

```
# Input signals simply listed and separated by spaces
A B

# Multiple output functions listed on separate lines in min-terms
S = 1 2
C = 3
```

The following example definition file specifies the syntax rules on writing a definition file:

```
# Comment lines start with `#'
# Blank lines will be skipped
# ALL tokens must be separated by spaces for correct parsing!!!

# Input signals are simply listed and separated by spaces
A B C D

# Multiple output functions are listed on separate lines
# A function definition starts with an output signal name
# followed by an "=" sign, then followed by min-terms.
# Optionally min-terms can be followed by "\", then
# "don't care" terms.
F = 2 3 5 10 \ 7 8
G = 1 3 8 11 12
```

Our generic auto-grading script for any combinational circuit lab can be found here[2]. Roughly it does the following:

1. Parse the input definition file line by line (skip comment / blank lines)
    (a) Parse the first line for input signals into a list of inputs
    (b) Parse the rest of the lines each as an output function into a list of outputs
2. Generate and drive the checks
    (a) Initialize a GPIO pin list for all GPIO pins available on a Raspberry Pi
    (b) Check if there are enough GPIO pins for the inputs and the outputs
    (c) For each input or output signal, allocate a GPIO pin from the GPIO pin list
    (d) Print out instructions for the user to wire their circuit to Raspberry Pi
    (e) For each binary combination of the inputs:
        i. Drive the inputs to the circuit
        ii. For each logic function:
            A. Read the actual output and compare it with the expected
            B. Print the test result (PASS or FAIL)

Digital circuit labs may have more signal lines than the number of GPIO pins available on a Raspberry Pi. Solutions are available to expand the number of GPIO pins using off-the-shelf port expander IC chips such as MCP23017 and MCP23S17 [7].

4.2 An ad-hoc solution for an example sequential circuit lab

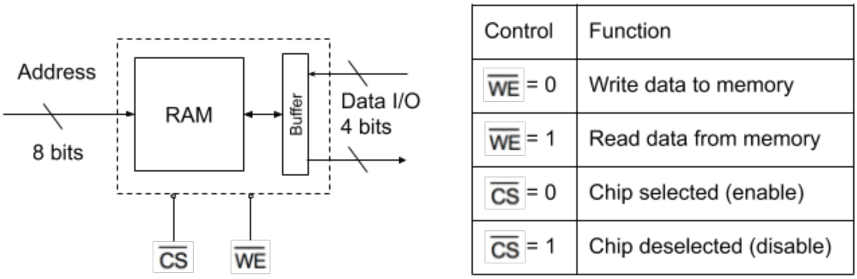
Unlike combinational circuits, the functionality of sequential circuits is not that easy to describe depending on the types of the circuits. Some types of sequential circuits can be relatively easy but others will likely be difficult. For this reason, we have not attempted a generic script for any sequential circuit labs at this time.

In the following, however, to illustrate some of the characteristics specific to sequential circuits, we demonstrate the use of an *ad hoc* script that can auto-check a sequential circuit lab we gave to our students. The lab in question is a RAM lab, in which students are asked to build a circuit for a random access memory system as illustrated in Figure 3 using an Intel 2114 static RAM chip.

The operation of any sequential circuit such as the one for this RAM lab requires the control signals to be given in proper order. For example, the address must be provided before the chip is selected. Certain signals must be maintained for a certain period of time to ensure proper operation. Together these constraints require the steps to be properly sequenced and delays to be inserted at critical junctions in the script.

A code snippet that tests writing/reading to/from the RAM is given below:

[2] https://drive.google.com/open?id=1HZ0FEM45M0k03MpQ2hUgkyUrqnQiEH4u



Control	Function
$\overline{WE}$ = 0	Write data to memory
$\overline{WE}$ = 1	Read data from memory
$\overline{CS}$ = 0	Chip selected (enable)
$\overline{CS}$ = 1	Chip deselected (disable)

(a) Block diagram

(b) Function table

Figure 3: RAM Lab.

```
# Write value `val' to address `addr'
setPins(addr, addrIn)   # set address pins with `addr'
time.sleep(0.001)       # address setup time
setPins(val, dil)       # set data input lines with `val'
weIn.off()              # write mode enabled
csIn.off()              # chip selected
time.sleep(0.001)       # hold sufficient time for write
csIn.on()               # chip deselected (after data written)
weIn.on()
time.sleep(0.001)       # input data hold time

# Read from address `addr' (data goes into `dat')
setPins(addr, addrIn)   # set address pins with `addr'
time.sleep(0.001)       # address setup time
weIn.on()               # read mode enabled
csIn.off()              # chip selected
time.sleep(0.001)       # hold sufficient time for read
dat = readPins(dol)     # read data from the data output lines
csIn.on()               # chip deselected (after data read)
time.sleep(0.001)       # output data off delay
```

In the above, we have inserted more delays than is required. Actual required delays can be shorter depending on the types of chips used. The complete script is given here[3].

4.3 Discussion

We have successfully deployed this solution in two of our hardware-oriented courses at our institution. For each DLD lab, we asked students to do a self-check with a Raspberry Pi. Student feedback has been all positive as it is easy to do. Students are reportedly more engaged. For any failed check, students

[3] https://drive.google.com/open?id=1C4Mt7QNvEt7uL73MQU5rxSkhShYiMlz5

were motivated to address their design and / or implementation issues. Failed checks were often due to wrong wiring and misreading of the datasheets – that is the whole point of the automated checks. We have not seen any students abandoning their labs in the last two years. Before this solution was deployed, we had no idea how many students failed or partially failed a lab without telling us.

As currently implemented, this automated check is black box in nature – no details of the circuits are required for the script to run. Debugging help is not supported but users can use the results of the auto-check as their guide to troubleshoot issues. It is debatable as to whether it helps to pinpoint exact errors in their circuits with specialized tools such as Toastboard [3] when it is the time for the students to develop their own debugging skills. In addition, to be able to pinpoint the exact errors in a circuit, systems such as Toastboard have to be provided with the exact schematics of the circuit along with the information on where each component is wired on the breadboard. This can be quite complicated and hard to use.

5 Conclusions and Future Work

In this paper we presented a solution that can auto-check/grade digital logic design labs similar to autograders for programming assignments. Our solution to auto-check a DLD lab is very simple to use. By taking advantage of the $35 Raspberry Pi with physical computing capability, the solution is also an inexpensive one.

This solution has been successfully used at our institution. It has helped instructors to greatly reduce the amount of manual work needed in the grading of DLD labs. Students enjoy the solution because it is capable of providing quick feedback to them about their lab work. Such feedback encourages them to keep trying. The solution keeps students engaged at a higher level.

Moving forward a more generic solution for sequential circuit labs is an apparent direction worth pursuing. A simple outright solution may not be immediately available but generic scripts for certain subtypes of sequential circuits are very likely.

References

[1] Carl Burch. Logisim. `http://www.cburch.com/logisim/index.html`, Accessed May 2, 2019.

[2] Turing's Craft. CodeLab: A powerful tool for programming instruction. `https://www.turingscraft.com`, Accessed May 2, 2019.

[3] Daniel Drew, Julie L Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Björn Hartmann. The Toastboard: Ubiquitous instrumentation and automated checking of breadboarded circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 677–686. ACM, 2016.

[4] Rensselaer Center for Open Source. Submitty. `https://submitty.org`, Accessed May 2, 2019.

[5] Raspberry Pi Foundation. Raspberry Pi – Teach, Learn, and Make with Raspberry Pi. `http://www.raspberrypi.org`, Accessed May 2, 2019.

[6] Katy Jordan. Massive open online course completion rates revisited: Assessment, length and attrition. *The International Review of Research in Open and Distributed Learning*, 16(3), 2015.

[7] Derek Molloy. *Exploring Raspberry Pi.* Wiley Online Library, 2016.

[8] Ben Nuttall. GPIO Zero: A friendly Python API for physical computing. `https://www.raspberrypi.org/blog/gpio-zero-a-friendly-python-api-for-physical-computing/`, Accessed May 2, 2019.

[9] Carnegie Mellon University. Autolab Project. `http://www.autolabproject.com/`, Accessed May 2, 2019.

[10] Te-Yen Wu, Hao-Ping Shen, Yu-Chian Wu, Yu-An Chen, Pin-Sung Ku, Ming-Wei Hsu, Jun-You Liu, Yu-Chih Lin, and Mike Y Chen. Currentviz: Sensing and visualizing electric current flows of breadboarded circuits. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 343–349. ACM, 2017.

Similarity Matching in News Articles[*]

Nathaniel Ballard and Deepti Joshi
Department of Cyber and Computer Sciences
The Citadel
Charleston, SC 29409
{nballard,djoshi}@citadel.edu

Abstract

With the need for intelligence growing every day, big data analytics has become the forefront of intelligence work. Having the ability to access news archives and other news article databases have given a data analyst a plethora of information to analyze, but which article is the same event, is there any redundant data within these collection sets of data? This paper expands on these questions. By implementing cosine similarity and vector space modeling, we can start to get a better understanding of the dataset. The results suggest that an article can be related to another article by topic or word semantics allowing now the creation of a web of articles that are similar or in some cases even the same.

1 Introduction

With most modern newspapers being available online, and now containing easily accessible archives, access to big data has become more available to the public. However, this abundance of news articles leads to the question – is there some way to automate the discovery of article relation without manually reading every single article? As of 2017, there were 30,948,149 total estimated circulations of U.S. weekday newspapers [6]. It is safe to say that many if not all of these articles contain different writing styles. So even though two articles may be about the same event or topic, different writing styles can make them significantly distinct from each other. This makes the problem even more significant, where comparing large amounts of data with different writing styles can be daunting, but not impossible. There already exists natural language processing (NLP) algorithms, latent semantic analysis (LSI) [3,1], and topic detection algorithms [4,5] that can analyze plain-text documents for semantic structure [2]. However, these existing algorithms do not account for different writing styles and semantics. In this paper, we outline one possible solution that solves the problems stated above in a clear and concise manner.

Currently, our work centers around the use of the algorithm developed by Radim Řehůřek named Gensim [2]. By compiling Gensim's analysis for plain text documents, we have created a structure that can analyze newspaper articles' pairwise similarities based on different parameters such as the topic of the article, the primary actions in the article, the actors involved, or even a full redundancy of the article.

The paper is outlined as follows: Section 2 discusses the background of Gensim, why it was chosen, its features and how it will be implemented. Section 3 presents the methodology of our ongoing work – the ideas behind the structure implemented and how it is used in real time. Section 4 shows our preliminary results. We also analyze and expand the results to discuss the importance of this research. Finally, Section 5 presents the conclusion and future work.

2 Background

Gensim (gensim = "generate similarly") is a topic detection modeling for humans algorithm developed by Radim Řehůřek for his Ph.D. thesis. This project uses Gensim's algorithms to develop a structure to be used to tackle our big data set and produce effective results. Specifically, the solution uses two sets of algorithms in Gensim's "analyze plain-text documents for semantic structure" [2] approach, namely, cosine similarity and vector space modeling. Vector space modeling is known as the representation of documents as vectors in common vector space [3].

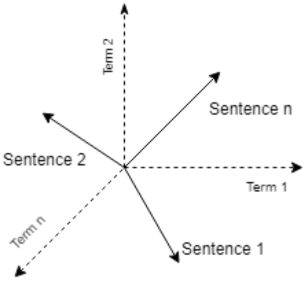


Figure 1: Vector Space Model as a graph [4].

Looking at Figure 1, sentence 1, sentence 2, and sentence n can be treated as documents on a vector space. Then when the documents are placed in the vector space model, Gensim uses cosine similarities to find the distance between each of the articles.

$$\vec{a} * \vec{b} = ||\vec{a}||||\vec{b}||cos\theta \text{ , where } cos\theta = \frac{\vec{a}*\vec{b}}{||\vec{a}||||\vec{b}||}$$

The cosine similarity equation ($\vec{a} * \vec{b}$) shown above takes two vectors that create a triangle; then the method finds the angle between the documents which is the result of the similarities test, by looking at the angle of the vectors instead of the magnitude [4]. By using the combination of vector space modeling and cosine similarities, we create a space that allows the algorithms to calculate similarities between the text of two articles, which is the entire idea of this research problem.

Figure 2 shows how different vector locations can affect the similarity score. The first graph contains an acute angle meaning that the theta of the cosine similarity is smaller or nearing zero giving a similarity value of close to 100% or 1. The middle graph contains an orthogonal angle meaning close to 90 degrees. In this case, because the vectors are orthogonal, the similarity score is 0%. In the last graph, the angle of the two vectors is approaching 180 degrees meaning that the "documents" are opposites of each other, giving a score of -100% or -1. With the triangles and angles laid out, we can state that our scale for the document similarity is [-1 to 1] – going from least similar (nothing in common) to most similar (almost duplicates).

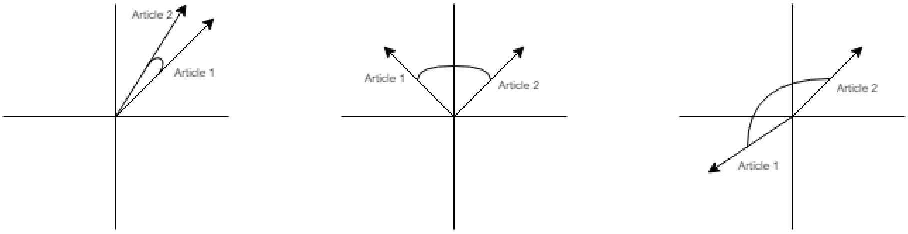
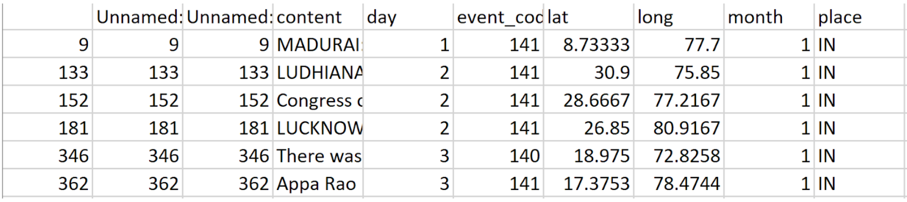


Figure 2: Different vector modeling spaces using cosine similarity [4].

The objective of using aforementioned algorithms in our work is to apply them to a much larger dataset, and in the intelligence field. This paper will go into greater detail about what was adjusted and developed on the algorithms in the methodology section of this paper.

3 Methodology

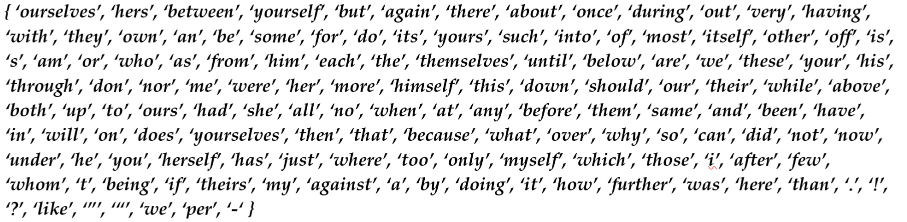
The objective of this project was to analyze unrest articles by finding pairwise similarities to discover redundancies and episodes of unrest. We started with a CSV file (Comma-Separated Values; see Figure 3) of pre-tagged articles about unrest. We are classifying unrest as anything that follows the following definition. “A state of dissatisfaction, disturbance, and agitation in a group of people, typically involving public demonstrations or disorder.“ The goal of this work is to find redundancies in the dataset of unrest articles, where the redundancies can be of two kinds – the same article is added to the database twice, or the same unrest event is reported from different sources. Additionally, we also want to find articles reporting on related unrest events.



	Unnamed:	Unnamed:	content	day	event_coc	lat	long	month	place
9	9	9	MADURAI	1	141	8.73333	77.7	1	IN
133	133	133	LUDHIANA	2	141	30.9	75.85	1	IN
152	152	152	Congress c	2	141	28.6667	77.2167	1	IN
181	181	181	LUCKNOW	2	141	26.85	80.9167	1	IN
346	346	346	There was	3	140	18.975	72.8258	1	IN
362	362	362	Appa Rao	3	141	17.3753	78.4744	1	IN

Figure 3: CSV file layout used for the analysis.

Our initial dataset contains 502 articles from the pre-tagged CSV file. When comparing each article to itself and others, we get (502x502) 252,004 similarity queues. However, to get to this point, we had to significantly change the Gensim stop-word list, which is a list of words that need to be pulled out of the vector space because they are redundant or might skew the data. The stop-word list used is shown in Figure 4 below.



{ ‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’, ‘about’, ‘once’, ‘during’, ‘out’, ‘very’, ‘having’, ‘with’, ‘they’, ‘own’, ‘an’, ‘be’, ‘some’, ‘for’, ‘do’, ‘its’, ‘yours’, ‘such’, ‘into’, ‘of’, ‘most’, ‘itself’, ‘other’, ‘off’, ‘is’, ‘s’, ‘am’, ‘or’, ‘who’, ‘as’, ‘from’, ‘him’, ‘each’, ‘the’, ‘themselves’, ‘until’, ‘below’, ‘are’, ‘we’, ‘these’, ‘your’, ‘his’, ‘through’, ‘don’, ‘nor’, ‘me’, ‘were’, ‘her’, ‘more’, ‘himself’, ‘this’, ‘down’, ‘should’, ‘our’, ‘their’, ‘while’, ‘above’, ‘both’, ‘up’, ‘to’, ‘ours’, ‘had’, ‘she’, ‘all’, ‘no’, ‘when’, ‘at’, ‘any’, ‘before’, ‘them’, ‘same’, ‘and’, ‘been’, ‘have’, ‘in’, ‘will’, ‘on’, ‘does’, ‘yourselves’, ‘then’, ‘that’, ‘because’, ‘what’, ‘over’, ‘why’, ‘so’, ‘can’, ‘did’, ‘not’, ‘now’, ‘under’, ‘he’, ‘you’, ‘herself’, ‘has’, ‘just’, ‘where’, ‘too’, ‘only’, ‘myself’, ‘which’, ‘those’, ‘i’, ‘after’, ‘few’, ‘whom’, ‘t’, ‘being’, ‘if’, ‘theirs’, ‘my’, ‘against’, ‘a’, ‘by’, ‘doing’, ‘it’, ‘how’, ‘further’, ‘was’, ‘here’, ‘than’, ‘.’, ‘!’, ‘?’, ‘like’, ‘”’, ‘“’, ‘we’, ‘per’, ‘-‘ }

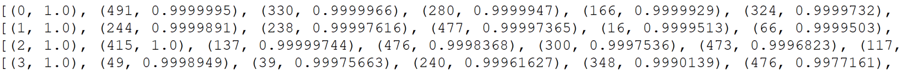
Figure 4: Stop-word list for words to ignore

The reason we chose this extensive list of stop-words, is that we wanted to pull out any words from the articles that might have skewed our data away from the significant vocabularies of unrest. We compiled a stop-word list of common words, then built onto it. The list was made up of words that were scraped from our sample article data set that were non-significant and common English stop-words. With the inclusion of our words from our research and common English stop words we eventually creating a more accurate list that led to more accurate results as shown in Figure 4. Although this is an extensive list every writing style is different, so words are still being added as new articles are being added to the dataset. The ultimate goal is to make the most comprehensive and practical stop word list for English news article analysis.

Another feature that we edited in the code is the frequency counter to count the number of times a particular word shows up in a selected article. We set a bound on the frequency count to ‘2’, to make sure that any words that were not significant and not being picked up in the stop-word list did not skew our final results. Once the stop-word list has completed removing all of the unnecessary words Gensim places the remaining words into a corpus or a dictionary using Latent Semantic Indexing (LSI) [1] that can be translated into a vector space. After the translation to a vector space we started comparing each article to itself and all other articles using Genism’s similarity algorithm [2] mentioned in the background section.

4 Results

The results were delivered in a 502x502 matrix of ordered tuples from most like the originating article to the least like the originating article. The data is displayed as (article number, similarity value). See Figure 5 for an example.



```
[(0, 1.0), (491, 0.9999995), (330, 0.9999966), (280, 0.9999947), (166, 0.9999929), (324, 0.9999732),
[(1, 1.0), (244, 0.9999891), (238, 0.99997616), (477, 0.99997365), (16, 0.9999513), (66, 0.9999503),
[(2, 1.0), (415, 1.0), (137, 0.99999744), (476, 0.9998368), (300, 0.9997536), (473, 0.9996823), (117,
[(3, 1.0), (49, 0.9998949), (39, 0.99975663), (240, 0.99961627), (348, 0.9990139), (476, 0.9977161),
```

Figure 5: Example of the output where each line is a new article

In our dataset the results for Article 0, 307, and 95 are: (0,1.0), (307,0.95), (95,-0.99) where each of the results are comparing to article 0. *Article “0” states: “While many All India Anna Dravida Munnetra Kazhagam supporters conducted celebrations on the occasion of VK Sasikala assuming charge as the general . . . ”* , and *Article “307” states: “The sudden ban on sale of crackers in NCR has caught shopkeepers by surprise. . . Both wholesalers and retailers are now scared after the ban. . . ”*

We can see that these two articles are not talking about the same event, but with a score of 0.95 between the two articles we know that they are talking about the same issue, and here we can see that the issue is “protest” and “surprise.” The first article is talking about an election and the protest and surprise against a political leader, and the second is talking about a firecracker ban and the surprise and protest against the ban. We can see that these are not the same event, but after doing the similarity analysis, we can see that they are related in terms of semantics and being about protests.

On the other hand, *Article “95” states: “ABOUT 60 members of a caste-based outfit manhandled filmmaker Sanjay Leela Bhansali, damaged his crews* *equipment ... For all the latest Entertainment News, download Indian Express App."* Here, we can see that it is discussing mishandled equipment and damage. This article is not about protest or surprise, so that is why it received a similarity score of -0.99.

5 Conclusion and Future Work

The work presented in this paper is our initial work of finding similarities within news articles related to unrest with the ultimate goal of finding episodes of unrest – that is, related unrest events. We have created a stable structure that produces accurate results with the use of Genism and providing our stop-word list and customizing other parameters to work with our dataset. Our results proved that the similarities found thus far are accurate and reliable.

Moving forward we will further modify the stop-word list with every new article because each writer's style is different, and there creating a comprehensive English news article analysis stop word list. We will also take into account the time and space for each article, to say that two articles are related. We want to say that an event in an article is related to another event not only semantically but also in its location and in time.

References

[1] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. Journal of the American Society for Informational Science 41, 6 (1990), 391–407.

[2] Radim Řehůřek. 2019. Gensim Topic Modelling for Humans. (April 2019). Retrieved May 6, 2019 from https://radimrehurek.com/gensim/

[3] Cambridge University Press. 2009. The vector space model for scoring. (April 2009). Retrieved May 6, 2019 from https://nlp.stanford.edu/IR-book/html/htmledition/the-vector-space-model-for-scoring-1.html

[4] Christian S. Perone. 2013. Machine Learning :: Cosine Similarity for Vector Space Models (Part III). (December 2013). Retrieved May 6, 2019 from http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

[5] James Allan. 2002. Topic Detection and Tracking: Event-Based Information Organization. Kluwer Academic Publishers, Norwell, MA, USA.

[6] PEW Research Center. 2018. Newspapers Fact Sheet. (June 2018). Retrieved May 6, 2019 from https://www.journalism.org/fact-sheet/newspapers/

Categorizing User Stories in the Software Engineering Classroom[*]

Brian T. Bennett and Tristan Onek
Department of Computing
East Tennessee State University
Johnson City, TN 37614
{bennetbt,onektr}@etsu.edu

Abstract

User story documentation is a significant aspect of Agile development in which developers document possible actions that users may take within a software system. It is essential to educate students in a software engineering curriculum on how to create this documentation so they can be competent developers in their future careers. This study uses the INVEST user story rating system to assess user stories that students wrote for a term project in a two-part software engineering course series. We demonstrate potential issues that students experience in user story creation based on the INVEST analysis and propose potential solutions to this problem.

1 Introduction

The use of agile development methodologies has increased in the software engineering industry since the signing of the Agile Manifesto [2] in 2001. This manifesto grew from the application of lean production principles to software engineering [4], and recognized that software development life cycles should focus on customer interaction and user needs rather than copious amounts of documentation. Therefore, the documentation focus of agile methods is writing down system features in terms of user interactions. Many modern software systems, such as web and mobile applications, involve heavy user interaction with the system, making documentation of such interactions imperative for completing the project successfully.

The most common form of agile requirements documentation is the *user story*. User stories provide a lightweight method for documenting a user's interaction with a piece of software. The most common form of a user story is "As a <user role>, I want to <perform a task>, so I can <accomplish a goal>." These short statements describe in a few words what a user would like to do within the system and why they would like to do it, but the story format is not intended to describe how a developer should implement the feature. User stories accomplish two things: (1) they allow customers to focus on each action within the system, and (2) they provide developers with small, manageable chunks of work and with implementation freedom.

Teaching students the proper format for user stories and the motivation behind this format is vital for creating well-prepared employees. In a software engineering curriculum, students should learn about and experience agile software development through participation in a group project using agile concepts to create a modern software system effectively. Participation in an agile project requires students to create and maintain a backlog of user stories as an integral part of their education, addressing the Agile Manifesto's [2] concerns for customer and user interaction. However, many students are unable to write 'good' user stories when they must first use the concept. This study aims to categorize the problems found in student-produced user stories. We analyze user stories produced by software engineering students during group project assignments and categorize them using the INVEST method developed by Bill Wake [5]. Locating the students' weaknesses in user story design based upon the INVEST principles can lead to appropriate revisions to the curriculum to prevent students from causing documentation debt in future class assignments and upon entering the industry. Based on the analysis and judgments made, we conclude how to better educate students about user story development and use, while keeping the limitations of the study in mind.

2 Motivation

Despite how useful user stories are for documentation, researchers show that limitations exist when using them. Issues with poorly designed user stories can lead to issues with code based on those user stories. Mendes et al. [3] identify common issues with user stories and solutions to those issues, while Wake [5] gives a precise method for analyzing user stories that has become a standard for determining story quality.

Mendes et al. [3] describe documentation debt as the impact of missing, inadequate, or incomplete documents in a software engineering project. Mendes et al.'s [3] goal is to analyze documentation debt instances in a software engineering case study through agile requirements. Artifacts like user stories constitute part of a project's agile requirements [3]. These user stories cannot be effective if certain causes lead to the aforementioned issues of incompleteness and insufficiency. Mendes et al. [3] define several common causes leading to these issues, including a lack of information, requirement volatility, and lack of non-functional requirements (NFRs). For example, consider the following user story.

> "*Get the back end set up.*"

The story is not user-centered and attempts to describe an NFR. It is also non-specific because it lacks information that is pertinent to the feature. It has no meaning to anyone other than the project team who created it; even the creators of this story could forget the meaning after time passes.

Students who create user stories should consider them carefully. One method of performing careful consideration is through the INVEST mnemonic created by Bill Wake [5] to assist with user story development and evaluation. Wake [5] observed that good user stories should be *independent* (I) from other stories to allow for easier scheduling. In addition, good stories should be *negotiable* (N) to give flexibility in implementation. Furthermore, good stories are *valuable* (V) to the customer and will show a return on the investment of development effort. User stories should also be *estimable* (E) to allow for easier scheduling and prioritization. Another quality of good user stories is that they are *small* (S) in scope, effort, and description. Finally, Wake [5] notes that a good story should be *testable* (T) so developers can effectively write test cases for it. For novice software engineers, especially students, producing user stories that meet these principles can be difficult.

The above studies provide considerations regarding the problem presented in this research–improving the quality of student user stories. Good user story design should ultimately lead to a minimization of documentation debt to prevent other development issues from arising. If students understand how to write user stories that accurately reflect actions users will take, and can also write stories to adhere to standards such as INVEST, they will be more prepared to work effectively in their future careers.

3 Approach

This research uses the INVEST mnemonic [5] to assess students' user stories. The INVEST system's factors are significant because they consider the different ways that user stories may add business value to a project. In the software engineering curriculum, students learn about the importance of adding business value throughout the development life-cycle. This system can be applied to student user stories to determine if they understand how to write stories that will result in increased business value. Students from several sections of two courses were assessed–Software Engineering 1 and Software Engineering 2–from Spring 2018 to Spring 2019. In both courses, students were divided into groups each semester to create a software system for a client and were required to write user stories for their projects. Students used the tracking software Jira [1] throughout each semester to track user stories, and this allowed for ease of data collection.

In total, seventeen projects were selected for analysis based on the availability of user stories in these projects. Because of the availability of user stories, only seven projects were used from Software Engineering 1, while ten were from Software Engineering 2. Ten user stories were chosen randomly from each project, resulting in 170 user stories. Next, both authors assessed each user story based on the INVEST criteria, quantifying each user story's INVEST values using a binary system in which '0' does not satisfy the category and '1' satisfies the category. The authors' scores were then averaged to determine final ratings. The sums of the averaged values represent the user story's overall strength from 0 to 6, where 0 represents the worst score, and 6 represents the best score. This assessment allows computation of how many of the overall user stories are considered 'good' according to INVEST, and the number that adheres to each INVEST principle. These data points provide insight into student performance and provide direction for updated instruction with user stories in the software engineering classroom.

4 Results

Results are based on the 170 user stories analyzed using the INVEST method–70 stories from Software Engineering 1 and 100 stories from Software Engineering 2. Figure 1 shows a histogram of INVEST scores. Only five user stories received a perfect score (2.9%). The majority of user stories–101 (59.4%)–fell in the 4-5 range, with 5 being the most frequent value (32.35%). User stories with INVEST scores in the range 0-3 accounted for 37.64% of the stories analyzed, with 22 (12.9%) stories having an INVEST score of 0.

Figure 2 shows the average overall scores for each INVEST characteristic (where 170 is the maximum) and the percentages of each. The highest scoring characteristic is *value*, with an average score of 122.5. Results indicate that approximately 72.1% of the user stories analyzed were estimated to be valuable to the customer. The next-highest characteristic is *testability* with an average score of 114 of 170 (67.1%). These numbers indicate that two out of every three user stories contained enough information to write test cases for verifica-

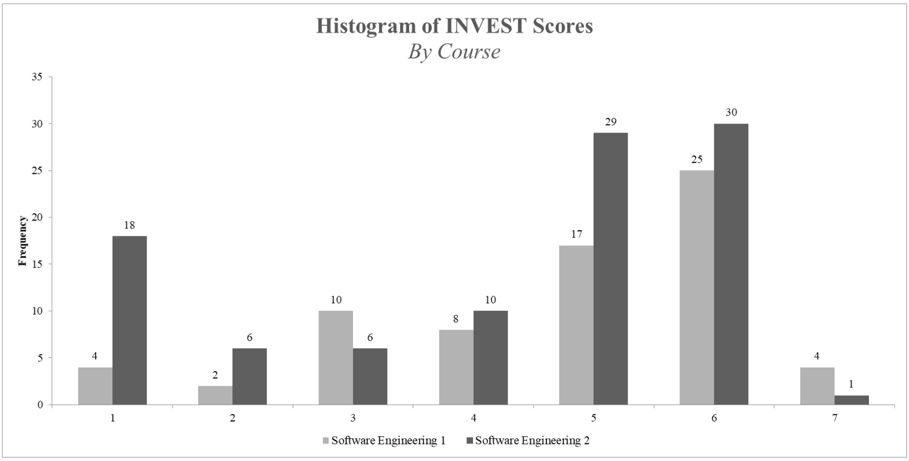


Figure 1: Histogram of INVEST Scores in the analyzed data

tion. Similarly, 112.5 user stories were *negotiable* (66.2%), containing pertinent details without attempting to enforce a specific design. A total of 97 stories (57.1%) were considered *small* enough for agile development purposes. A total of 93.5 (55.0%) user stories were considered *estimable*, providing enough information to estimate the time required for the feature. The lowest scoring

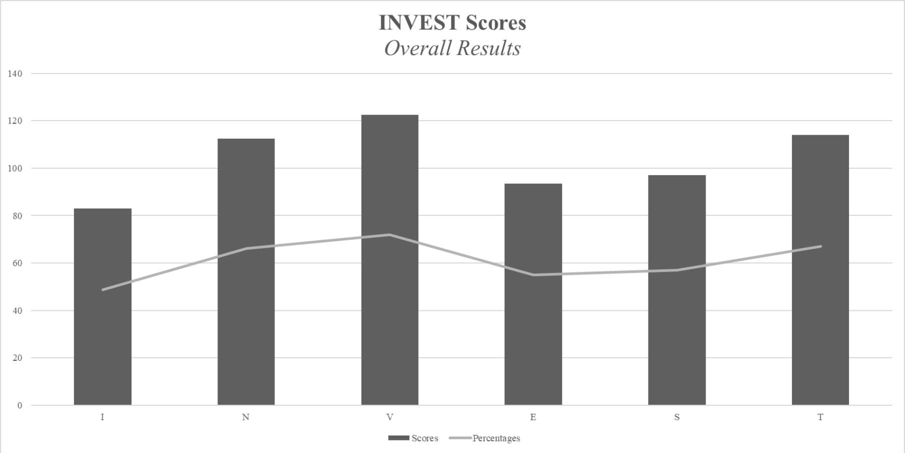


Figure 2: Overall INVEST Scores and Percentages

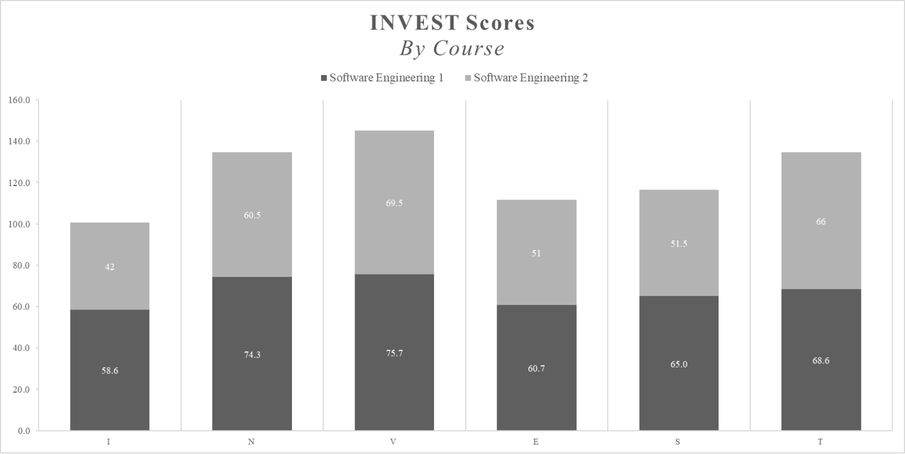


Figure 3: Normalized INVEST Scores of user stories created in Software Engineering 1 and Software Engineering 2

characteristic is *independence*, where only 83 (48.8%) of the user stories could be developed without dependency with others.

Figure 3 shows normalized details by course. This figure shows percentages for both Software Engineering 1 and Software Engineering 2 in each category. Because the two Software Engineering courses are intended to be taken as a sequence, one would hypothesize Software Engineering 2 students would show improvements over those in Software Engineering 1. However, results show this is not the case. In each INVEST category, Software Engineering 1 students outperform Software Engineering 2 students. The largest difference is in the *independent* category, where 58.6% of Software Engineering 1 user stories are considered independent but only 42% of Software Engineering 2 stories are considered independent, a difference of 16.6%. Others with fairly large gaps include the *negotiable* (a gap of 13.8%) and *small* (a gap of 13.5%) characteristics. The smallest gap (2.9%) is in the *testability* characteristic.

5 Discussion and Future Work

The assessments on student user stories indicate that students are more capable of fulfilling some INVEST characteristics than others. This leads to the conclusion that students must learn not only how to write a user story but also what components should be present and why those components are necessary. Students demonstrate a stronger capacity to make user stories testable and valuable, but these alone do not make a user story complete under the INVEST criteria. Independence, estimation, and breaking stories into smaller features should be enforced when teaching user story development. Because scores dropped in the second course of the sequence, INVEST principles must continue to be reinforced when requiring students to write user stories throughout the course sequence.

To reinforce INVEST principles, which also support proper development practices, students must receive consistent reinforcement on how to create good user stories. Assessing student documentation abilities should be not done once but multiple times through both courses. Thoroughly reviewing students' user stories at each stage of the group project and placing more focus on the quality of this documentation can help students develop better habits with user story creation and can also serve as a way to demonstrate how important user stories are. This focus on reviewing documentation ideally leads to less documentation debt both in the group projects assigned and in future industry jobs.

Some limitations exist within this study that could affect the stated conclusions. First, this study is dependent on the INVEST system rather than other rating systems that may have a more comprehensive picture of user story quality. Evaluating other review systems may prevent this conclusion from being entirely dependent on one system. Second, the study required the manual rating of each user story, which is subjective. Although having two people scoring each story attempted to mitigate this limitation, having more people scoring stories would be ideal. Third, the study involved a small sample size of only 170 user stories. Randomly choosing 10 stories from 17 projects could affect results by missing problems that exist but were not selected for analysis. However, the data set will continue to expand in future semesters.

This study provides a brief analysis of common issues with user stories in software engineering education with a proposed solution to mitigate some of these issues. The conclusions and reviewed literature in this study may serve as a basis for future experiments that assess software engineering students across different performance metrics. Future analysis may take the form of the case study that Mendes et al. [3] performed, except that students would be analyzed instead of professional software engineers. This would provide similar insights that Mendes et al. [3] collected, but would be through an academic rather than an industry context. By focusing on the academic context and understanding its significance, however, students can receive appropriate training on essential matters such as documentation before transitioning into their career paths.

References

[1] Jira. https://www.atlassian.com/software/jira.

[2] Manifesto for agile software development. https://agilemanifesto.org/.

[3] Thiago Souto Mendes, Mário André De F. Farias, Manoel Mendonça, Henrique Frota Soares, Marcos Kalinowski, and Rodrigo Oliveira Spínola. Impacts of agile requirements documentation debt on software projects. *Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC 16*, 2016.

[4] I. Nonaka and H. Takeuchi. The new new product development game. *Harvard Business Review*, 64(1), 1986.

[5] Bill Wake. Invest in good stories, and smart tasks, 2013. https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/.

Rethinking the Role of Simulation in Computer Networks Education[*]

Qian Liu
Mathematics and Computer Science Department
Rhode Island College
Providence, RI 02908
qliu@ric.edu

Abstract

Fundamentals in Computer Networks are essential to one's deep understanding of network internals. In general, simulators are used in introductory networking courses to illustrate abstract concepts and to help students observe network behavior without requiring dedicated hardware. However, due to their limitations, they don't provide ways for students to investigate some essential topics. In this paper, we introduce several activities for students to practice, investigate, and learn those underlying essentials in detail. Our activities create a personalized learning environment in which students can learn things at their own pace and explore topics based on their interests. Our study indicates that students have achieved a better understanding of network internals and have gained practical skills after these activities.

1 Introduction

Simulation is typically used in Computer Networks courses to illustrate network fundamentals and to emulate various scenarios without requiring dedicated hardware. In general, a typical undergraduate networking course focuses on the topics listed in Table 1 although there are often differences in the teaching order or covered depth. Various simulations are introduced and used in different ways to illustrate those topics. Some are using simulation applets [9, 7] to illustrate fundamentals or algorithms in animations so that abstract concepts could be explained visually. Some are using network simulators such as GNS3 [2] or Packet Tracer [1] to provide students with subnet management practice without the need for hardware devices. However, one big concern is that simulators usually illustrate network fundamentals in an ideal scenario. In actual environment, network events are not well-organized and do not occur in the same order as in simulations. Therefore, student's understanding of those fundamentals may stay at the theoretical level.

Table 1: Topics offered in a typical introductory Computer Networks Course

Internet Stack	Details
Layer 5	Application protocols and Socket Programming
Layer 4	TCP error control, flow control, congestion control, TCP/UDP headers and their usage in data transfer
Layer 3 & Layer 2	IP fragmentation, subnet management, routing and switching, Ethernet, ARP, etc

Another concern is that students usually have no control on protocols and mechanisms because simulators only visualize their behaviors to users, and that conceals too much of lower layer details and their relations to upper layers [6]. For instance, OPNET [5] allows students to configure protocol parameters to compare performance in various scenarios, but it doesn't provide ways for students to investigate protocol details such as how error control mechanism handles various error cases to ensure reliability. In order to examine how protocols are working internally, students need to delve into simulator internals. For example, in order to investigate how error-control works internally in open-source simulator OMNeT++ [4] or NS-3 [3], students need to introduce customized models into these simulators, design experiments with specific traffic patterns, and collect specific network statistics. These require students to learn and deal with many simulator-private structures which would create a steep learning curve that goes beyond course requirements.

2 Design and Deployment

We have introduced several learning activities to help students investigate protocol details in our introductory networking course. Those activities are not a replacement for any existing simulators, instead, they are practical complements to them. In our activities, students are instructed to write their own “simulators” to delve into topics listed in Table 1 that are not fully covered or cannot be examined by existing simulators. Students are working in groups, using *socket* library to build simulators. Unlike existing learning models [11, 12, 10] that arrange students to work in specific framework, students in our activities are not limited in any context, and they can introduce customized models or extensions to explore topics based on their interests and learn things at their own pace.

2.1 Activity 1: Error Control Model

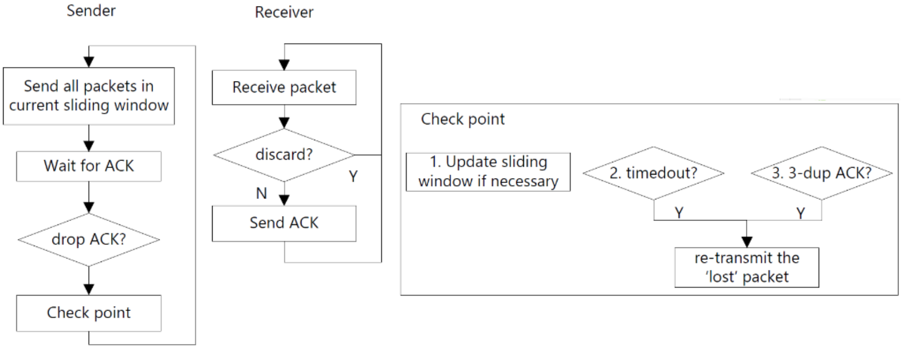


Figure 1: Basic Algorithms in Activity 1

In this activity, students are using socket model to simulate techniques in error-control mechanism: acknowledgement and retransmission, which not only are important to understand transport layer, but also have significant effect on the design of other networks, for instance, InfiniBand [8] uses similar techniques to ensure reliability in its transport layer. Students are instructed to build their simulators on top of TCP. A random packet discarder is introduced to discard packets (simulate packet loss), depending on a configurable “loss rate” parameter. That is, after receiving a packet, receiver runs the discarder to decide whether the current packet should be discarded “manually” as if it was not received previously. Figure 1 lists the basic algorithms used in this simulator. The reasons we use TCP and a random packet discarder, instead of implementing those features on UDP, are:

- students have control on the loss rate and can simply adjust it to compare different scenarios.
- It can create a personalized learning environment in which students could learn essential error-control techniques at their own pace. For instance,

when students start with the basic concepts, it is not necessary to introduce the discarder so that they could get familiar with basic TCP workflow in ideal situation; then, the discarder could be introduced to receiver only so that students could focus on how sender detects packet loss and deals with retransmission; afterwards, the discarder could be introduced to sender to simulate lost ACK scenarios, and in this case, more events should be considered.

How TCP ensures reliability is important for students to understand the internal principles of data transfer, but related techniques are usually discussed theoretically in classrooms with diagram illustrations or animations. Although existing simulators allow users to introduce packet loss rate, they don't provide ways for students to investigate those techniques in detail. In this activity, students build their own simulators to deal with packet loss detection, timed out, and retransmissions. That would help students obtain a deep understanding of those error-control techniques, and of how changes in attributes, such as sliding window size, loss rate and retransmission timer, impact network throughput. In addition, this activity allows students to introduce error models step by step to investigate error-control in various scenarios so that students can learn things at their own pace, thus creating a personalized learning environment.

2.2 Activity 2: TCP State Transition

The model in activity 1 could be reused in this activity to simulate and keep track of TCP state transition. Students are generally not aware of how TCP states are transited because it is usually discussed theoretically without any experiments, and existing simulators don't visualize its procedure nor provide interfaces for students to explore it. A good understanding of TCP states would prepare students for advanced topics because similar transition technique is used in RDMA QP (Queue Pair) transition [8]. This activity provides students with hands-on practice to fully understand how TCP state changes.

In this activity, students could introduce TCP states into the model in activity 1, handle incoming packets by analyzing their data payload (packet format shown in Figure 2 and discussed shortly), and change current TCP state if necessary. The random packet discarder could be disabled in this activity so that students can focus on the transition. Packets sent between two hosts (client and server) should at least convey the following simulated information: packet sequence number, ACK number, and flags bits (FIN, SYN, etc).

Simulation begins when one side, client, calls *socket connect* method regularly to connect to another side, server. Then, both sides maintain local states starting from "simulated" CLOSED state with the simulated sequence number field (Figure 2) set to 0. The complete transition procedure should be simu-

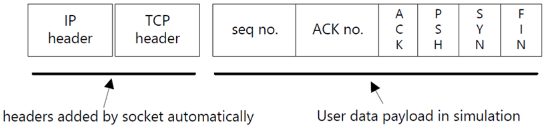

IP header
TCP header
seq no.
ACK no.
ACK
PSH
SYN
FIN
headers added by socket automatically
User data payload in simulation

Figure 2: Packet Format in our Activities

lated, that is, from one side sends a SYN packet (a packet with the SYN bit set) to initiate a (simulated) connection request, to one side sends a FIN packet to terminate the "simulated" connection. Both sides should move local states appropriately in response to different flag bits. The sequence number and ACK number fields must also be considered, for instance, if one sends a packet with SYN and ACK bits set (connection reply) and receives a packet with ACK bit set only, then the sequence number in the ACK packet must match the ACK number in the connection reply packet sent previously, otherwise, the TCP state cannot transit to the "simulated" ESTABLISHED state.

2.3 Activity 3: Segmentation and Fragmentation

In this activity, students implement the basic arithmetic operations of TCP segmentation and IP fragmentation in their simulations with several input parameters: a (pseudo) destination IPv4 address, a port number, message size, and a sequence of MTUs in which each value represents the MTU supported by a (pseudo) router, therefore, the list of MTUs simulates the path of routers between the local host and destination. The MSS value is set to the max MTU value of the list minus 40 (TCP and IP headers). Students should divide the (pseudo) message into segments and encapsulate them with appropriate TCP and IP headers in their simulations, that means, students should build a packet structure in their programs to simulate TCP header and IPv4 header exactly, and fill out necessary fields in these headers, especially the sequence number and port number in TCP header, and the length, DF, MF, fragment offset fields in IP header. When a packet traverses a "pseudo" router with MTU less than the packet length, the packet should be broken it into multiple fragments based on fragmentation policy. When a packet leaves the last "router", it should be buffered, if necessary, and re-constructed into a complete TCP segment.

This simulation provides students with hands-on practice in protocol headers, segmentation, and fragmentation; generally, the latter two are discussed without programming exercises in a typical networking course. Running *ping* command on a physical host would allow one to observe the fragmented packets but not the internal principles. This model helps students comprehend when and how external networks handle segmentation and fragmentation.

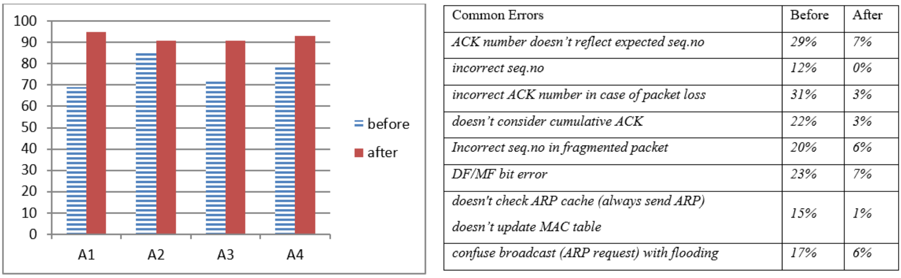
2.4 Activity 4: MAC Table and ARP

Students are instructed to implement switch self-learning capability and ARP procedure in a simple star topology consisting of a switch and eight host objects. Initially, each host object has a (pre-assigned) IPv4 address and an empty ARP cache table, and the switch object has an empty MAC table. In order to clearly reflect the link layer internals, we introduce a mechanism called *softMAC* that automatically converts an IPv4 address to a MAC address. The *softMAC* conversion is simple: it attaches 16 bits of 0s at the front of a given IP address to generate a MAC address.That means, a host with IP *a.b.c.d* has MAC address *0-0-a-b-c-d*. The simulation begins when a host sends a data packet to a random selected host. The sending host should follow the ARP look up procedure and send ARP request if necessary. The switch then checks its MAC table to decide where to forward the packet and whether it has learnt the MAC address. When a host receives a packet, it first determines the type of the packet (ARP or data packet) and then sends a new data packet to a new random selected host to continue the simulation if necessary. All hosts perform random communication until the switch records all MACs in its table, then the simulation stops.

Students could introduce more extensions in this model to gain better comprehension of link layer mechanisms. For instance, if MAC table aging time is introduced, what would happen if the switch object receives a packet but doesn't know where to forward it (difference between broadcasting and flooding). In addition, they would learn how to simulate sending/receiving by using certain events instead of generating actual traffic, for instance, they could register an event with pre-defined structure into a list in which each event will be executed in the future based on its timer. This is the general way modern simulators use, and we hope this activity would help students conduct advanced research in the future.

3 Results

We use these activities in our introductory networking course at undergraduate level with the objective of engaging students in effective learning and helping them comprehend core techniques in transport layer, network layer, and link layer. For each activity, we give two tests in a stepwise learning process. We first take a test after lecture and practice in network simulators, depending on topics we discussed, then, students work in our activities, and after that, we give another test on the same topics but with more advanced questions. Finally, we grade these tests and review student progress. Figure 3 (aggregated) shows grades comparison and the percentage of students making common errors before and after taking those activities. They indicate that students have shown remarkable progress, especially in error-control activity (A1), and have developed well-organized, structured knowledge after these learning activities.



Common Errors	Before	After
ACK number doesn't reflect expected seq.no	*29%*	*7%*
incorrect seq.no	*12%*	*0%*
incorrect ACK number in case of packet loss	*31%*	*3%*
doesn't consider cumulative ACK	*22%*	*3%*
Incorrect seq.no in fragmented packet	*20%*	*6%*
DF/MF bit error	*23%*	*7%*
doesn't check ARP cache (always send ARP) *doesn't update MAC table*	*15%*	*1%*
confuse broadcast (ARP request) with flooding	*17%*	*6%*

Figure 3: Student Assessment before and after our activities

4 Summary

This paper introduces several activities to help student practice, investigate, and learn networking essentials such as error-control techniques, state transition, segmentation and fragmentation, ARP and switch working mechanism. A good understanding of those not only helps students comprehend the architecture and principles of network communication, but also prepares them for advanced skills. These activities are not a replacement for existing simulators, instead, they are practical complements to introductory networking courses. More importantly, these activities allow students to introduce customized models to explore topics and learn things at their own pace, thus creating a personalized learning environment. According to our observation, these activities can be used not only in classrooms to demonstrate topics in various scenarios, but also in labs or after class to provide projects and hands-on practice to enhance effective learning.

References

[1] Cisco Packet Tracer.
`https://www.netacad.com/courses/packet-tracer`.

[2] GNS3 Simulator. `https://www.gns3.com/`.

[3] NS-3. `https://www.nsnam.org/`.

[4] OMNeT++ Discrete Event Simulator. `https://omnetpp.org/`.

[5] OPNET Simulator.
`https://www.riverbed.com/products/steelcentral/opnet.html`.

[6] D. Feinberg. Teaching Simplified Network Protocols. In *Proceedings of the 41st ACM technical symposium on Computer science education*, Mar. 2010.

[7] M. Holiday. Animation of computer networking concepts. In *Journal on Educational Resources in Computing*, 3(2), Jun. 2003.

[8] InfiniBand Trade Association. Infiniband Architecture Specification Volume 1, Release 1.3, March 2015.

[9] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach (7th edition).* Pearson, 2016.

[10] K. Lee, J. Kim, and S. Moon. An educational networking framework for full layer implementation and testing. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, Mar. 2015.

[11] J. M. Pullen. Teaching network protocol concepts in an open-source simulation environment. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, Jul. 2018.

[12] W. Zhu. Hands-on network programming projects in the cloud. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, Mar. 2015.

Detecting Areas of Social Unrest Through Natural Language Processing on Social Media[*]

Timothy Clark and Deepti Joshi
Department of Cyber and Computer Sciences
The Citadel
Charleston, SC 29409
{tclark6,djoshi}@citadel.edu

Abstract

With the growing use of internet and social media as a source for news, information is becoming faster and easier to access than ever before. The rise of internet and social media has also brought a voice to a much broader demographic. With this, each user has the ability to take the role of an active reporter, creating a massive amount of data on ongoing events. The goal of this research is to collect and review this data, from Twitter in particular, to detect, analyze, and display events of Social Unrest in India, Pakistan, and Bangladesh.

1 Introduction

From humble beginnings as amateur community platforms, social media has rapidly blossomed into a complex web of global information and online interaction [4]. Due to this, many have turned to social media as an outlet to voice their views and opinions on the communities, cities, and countries in which they live. Not only does this create a broader spectrum of view points about various events, but the response time in relation to those events far exceeds that of conventional news. As pointed out by *TwitterStand: News in Tweets*, there have been several events that show mass amounts of tweets pertaining to the subject hours before the first news source reports it[3]. The goal of this research is to use data collected from social media outlets, Twitter in particular,in an effort to detect, analyze, and display events of Social Unrest in India, Pakistan, and Bangladesh.

In order to do so, we have collected large amounts of geo-coded tweets from our region of interest (ROI), namely, India, Pakistan and Bangladesh. For more details, see section 3.1. While data is being collected, we have attempted to extract relevant data using several classification methods (see section 3.2). Next, in order to run spatial queries, we have reverse geo-coded the geospatial coordinates for the tweets (see section 3.3). Finally, we have started to conduct natural language processing on each tweet to determine the who, what, when, where, and why (5Ws) of each event (see section 3.4). Section 4 discusses our results and the use of the graphical database Neo4j. Finally, Section 5 presents the conclusion and future work.

2 Background

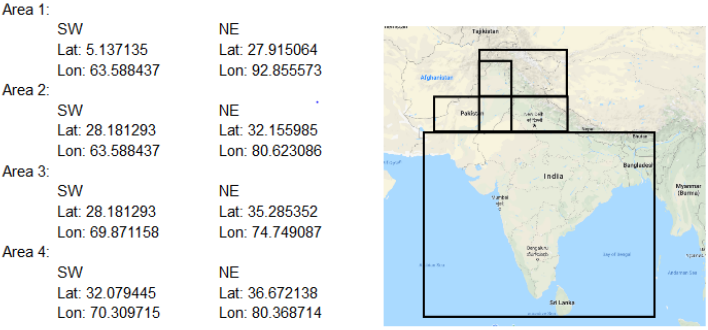
In recent years, twitter has become a major outlet for real-time news. Twitters restriction to the use of 280 characters works as a double edged sword. While confining users from creating a well formed idea or argument, it allows users to post short blurbs that serve as bulletin for a much larger idea [3]. Tying these blurbs in with images and links to further information allows tweets to serve as headlines to pressing issues and events. In previous works, like TwitterStand, applications have been created in order to sift through the noisy medium to extract underlying trending topics[3].

Events of social unrest can be defined as a demonstration or action from an individual or group against a larger group, organization, or government. While most events initially intend to serve as a demonstration to the public or government, in many occasions they often escalate into general chaos, resulting in violent forms of crime and social disorder [2].

3 Methodology

3.1 Collection

Our first step in this research was the collection of Twitter data from the ROI. To collect tweets and all of the metadata associated, we used two methods, both of which were built upon the Twitter API for developers. The first method used an open source Java software Tweets2SQL (https://github.com/jgontrum/Tweets2SQL). This software allows data from Twitter to be scraped in real-time and automatically imported into a MySQL Database. This set of tweets came with latitude and longitude for each tweet. However, for spatial queries to work, we needed to reverse geo-code each tweet (details in Section 3.3). In order to limit the amount of reverse geo-coding and to import new data into Neo4j - a graphical database that allows better visualization of dynamic relationships between the various data nodes, we began to collect Tweets using code created with the Tweepy python package (http://docs.tweepy.org/en/v3.5.0/). To limit and maximize the tweet collection from our ROI, we first mapped out several overlapping bounding boxes over the areas in question as seen in Figure 1. From there, using the coordinates of the SW and NE corner of each respective bounding box, these can be added into the streaming service's filter configuration.



	SW	NE
Area 1:		
	Lat: 5.137135	Lat: 27.915064
	Lon: 63.588437	Lon: 92.855573
Area 2:		
	Lat: 28.181293	Lat: 32.155985
	Lon: 63.588437	Lon: 80.623086
Area 3:		
	Lat: 28.181293	Lat: 35.285352
	Lon: 69.871158	Lon: 74.749087
Area 4:		
	Lat: 32.079445	Lat: 36.672138
	Lon: 70.309715	Lon: 80.368714

Figure 1: Bounding box for tweet scraping.

3.2 Classification

Once a large set of tweets had been compiled, identifying tweets related to unrest was the next task. The first approach implemented for this task was to look for presence of social unrest vocabulary key words within the tweet. This approach however returned futile results when applied to look for a tweet with any single term. Searching for multiple instances of our vocabulary, on the other hand, returned a very limited subset of our data due to the brevity of tweets. Thus, to identify tweets related to some form of social unrest, we applied supervised machine learning to classify tweets as related or unrelated to unrest. By hand selecting a subset of tweets and categorizing them as being within the context of social unrest, we can train a neural network to accurately classify tweets as social unrest. Facebook's open source software *FastText* has served as a wonderful framework to build our neural network. Using a training set of 146 hand selected tweets, *FastText's* neural net creates a list of unique words and weights them based on their appearance in tweets considered relevant or irrelevant [1]. From there, each new tweet is given a score based on the tweets text compared to that of the weighted list.

3.3 Reverse Geo-coding

While tweets collected recently through Tweepy include place names associated with the coordinates, our older data contains gaps in data that is key to viewing geographic connections. To solve this, reverse geo-coding allows us to link a physical address to each tweet. Using The Nominatim python package (https://geopy.readthedocs.io/en/stable/), we are able to input the latitude and longitude from each tweet node, and return a formatted address. Listing 1 provides an example of the output produced, providing several degrees of preciseness. These varying degrees will be used in displaying geographic connections and serve as points to query in our graphical database.

Listing 1: Reverse geo-coding output example

```
{'neighbourhood': 'Islampura', 'suburb': 'Cantonment',
'city': 'Sargodha', 'county': 'Sargodha␣District',
'state': 'Punjab', 'postcode': '40100',
'country': 'Pakistan', 'country_code': 'pk'}
```

3.4 Natural Language Processing

Once our data has been scrapped, classified, and formatted, we are able to start analysis on the tweets themselves. This is done using popular natural language processing (NLP) techniques of named entity recognition (NER) and parts of speech (POS) tagging. In order to help identify the 5W of analysis of tweets (**Who**, **What**, **When**, **Where**, and **Why** of the twitter post), the tweet must be stripped down to its key parts. Named entity recognition helps us with this by identifying organizations, people, and places referenced in the tweet. Parts of speech can also help us further extract context by finding verbs and other nouns in each tweet. With these we can start to identify each of the 5Ws for each tweet. These key items can then be attached to the corresponding tweet node to help identify similarities between tweets and where certain subjects or organizations are becoming topics of discussion.

4 Results

From our research, we now have a strong data set that can be used to identify key events of social unrest in our ROI. With our classification techniques we are able to identify tweets with accuracy of 77% based on a testing our neural net with a training set of 110 and and a testing set of 36. From our reverse geo-coding, we can now filter events based on key geographical points, as well as map coordinate points to view concentrations of social unrest tweets (see Figure 2). Lastly, our NLP strategies will allow us to set a foundation for topic detection by identifying the 5Ws for each tweet. Associating each tweet with a Who, What, When, Where, and Why, also provides further points to be used to filter data to analyze.

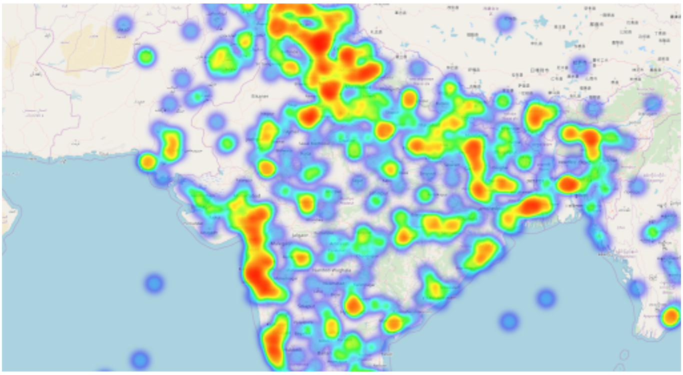


Figure 2: Heatmap of identified social unrest tweets

To compliment our research we have began the development on a web application to serve as an easy way to access, query and display every aspect of our data. Tying in our named entity recognition software, the text visualization tool is able to view text of tweets with each entity annotated and highlighted. Also, as stated before, with the use of the graphical database system Neo4j, we have the ability to create visual representations of each data point and the relations that connect each point. Figure 3 gives an example of how our database displays the geographic connections of tweets. In this example, each country node is linked with several nodes representing the cities residing in that country. Each city node is then connected with all tweets posted from that specific city or region. Note that the example graph is only partially expanded.

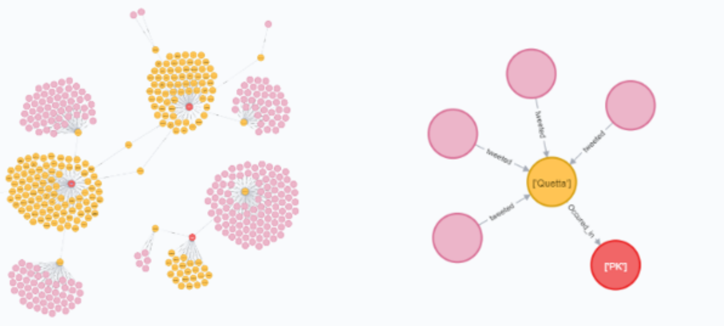


Figure 3: Neo4j graphical representation of Twitter data

5 Conclusion and Future Work

From this research, tweets can now successfully be processed to complete 5W analysis. Moving forward we hope to incorporate more elements of twitter, i.e. the use of hashtags and emojis, as another way to fill in components of the 5W analysis as well as start to analyze sentiment. Other aspects of of future work include refining our training set and classification methods to increase detection accuracy. As we further our research, we will also begin to expand our ROI into more countries. With this the importance of translating tweets while maintaining sentiment value will become pertinent in our research.

References

[1] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[2] Fengcai Qiao, Pei Li, Xin Zhang, Zhaoyun Ding, Jiajun Cheng, and Hui Wang. Predicting social unrest events with hidden markov models using gdelt. *Discrete Dynamics in Nature Society*, pages 1 – 13, 2017.

[3] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: News in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 42–51, New York, NY, USA, 2009. ACM.

[4] J. van Dijck. *The Culture of Connectivity: A Critical History of Social Media*. Oxford Scholarship online. OUP USA, 2013.

Take Note: An Investigation of Technology on the Line Note Taking Process in the Theatre[*]

René Borr and Valerie Summet
Mathematics and Computer Science
Rollins College
Winter Park, FL 32789
{rborr,vsummet}@rollins.edu

Abstract

The very nature of theatre is that every performance is unique, which establishes one of the main challenges when creating technology to be used in the theatrical setting. Often, technology used in the theatre is adapted from another field such as art, music, lighting or construction. This paper discusses the design, creation, and evaluation of a software program to help stage managers take line notes during rehearsal for theatrical productions.

1 Problem and Background

Before a play or musical can be seen by an audience, it must be fully rehearsed and have a series of full runs. A full run of a production occurs during a rehearsal when the show is performed in its entirety without any of the technical elements such as lighting, sound, or costumes. The stage manager is in charge of ensuring that the production as a whole goes smoothly by working alongside the actors, designers, and technicians. During a full run, the stage manager also takes line notes. Line notes serve as feedback to the actors and indicate what they said on stage versus what they should have said according to the script. Unfortunately, all current methods – including handwritten notes on a photocopy of the script or digital notes on a digital copy of the script – used are not fast or efficient enough to make the process of taking line notes simple for the stage manager.

2 Literature Review

There is little computing technology developed for theatre in comparison to other fields. However, there are current commercial software products available for theatre management. We begin by examining these areas.

One of the most challenging aspects of a theatre production is the collaboration between the different artists involved, such as costume designers, set designers, directors and actors. Theatre design software is developed to foster collaboration among the different parties. For example, A virtual reality system was developed to help theatrical designers collaborate over long distances [2]. This system helped designers visualize aspects of a production such as lighting design and abstract set designs by utilizing a tabletop projection system. Another system is a digital script user interface that allows both actors and directors to visualize a script in a more cohesive fashion [5].

There are multiple commercial software systems that assist with organization during a rehearsal process. Virtual Callboard [7], Propared[3] and Stage Write [6] are software systems used by both designers and actors to manage a production including document management or tracking an actor's location and movements during rehearsal. There are also systems such as Cuelist [1] and QLab[4], which were designed to help with the process of creating a prompt book, a book which specifies all cues for all parts of a production.

Products like the ones outlined in this section can prove to be invaluable when organizing large-scale theatrical productions. However, none of these systems assisted stage managers in taking lines notes.

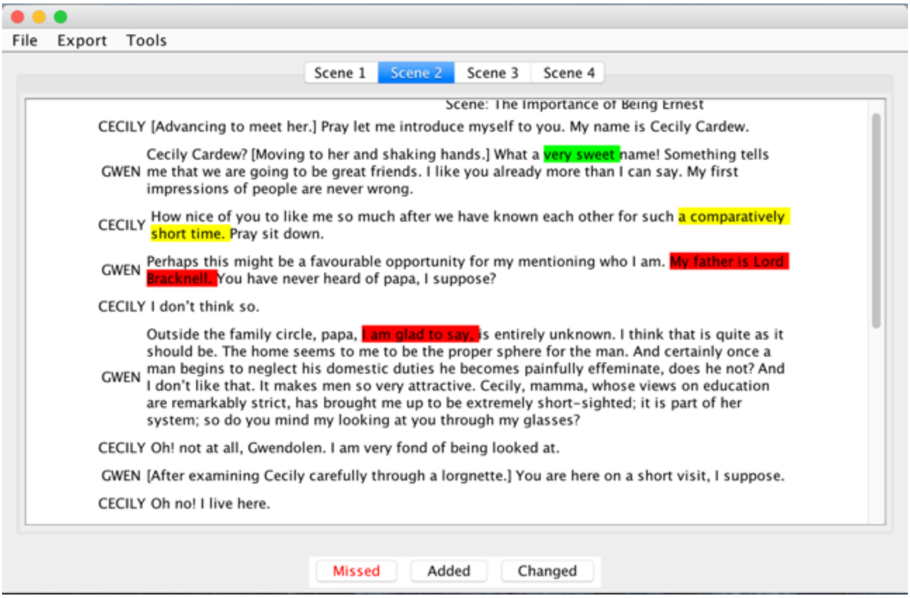
3 System Design

Before implementing the system, a series of preliminary interviews were conducted. A total of six stage managers with experience working on at least one full production were interviewed about their experience taking line notes.

While methods varied slightly, most stage managers highlighted the lines in a digital version of the script that were said incorrectly. The color of the highlight would correspond to the type of error made. Stage managers would re-organize lines said incorrectly and distribute them to actors after rehearsal ended. Based off of these interviews a series of user requirements were compiled:

- parse and format a text-based script for display.
- allow script traversal which facilitates monitoring the play in real time.
- allow the user to quickly mark specific words if a line was said incorrectly, preferably with color coordination.
- export all notes in a readable format which can be understood by an actor performing in the production.
- allow the user to categorize how lines are exported (by character or scene).

Following object-oriented principles and a loose SCRUM methodology, we developed a Java-based software system for taking line notes which met the above user requirements. The system consisted of a Parser which assisted the stage manager in importing a digital script and segmenting it by Acts, Scenes, Lines, Words, and Characters. This "master script" could then be saved and used again to avoid repeating the parsing process. The Notetaker Interface (Figure 1) is the GUI component that allows the user to notate errors said by the actor and includes the ability to navigate between Scenes (top buttons) and change categories of errors with a button click (lower buttons). The stage manager could then export these line notes with color coordination as feedback to the actors.



Scene: The Importance of Being Ernest

CECILY [Advancing to meet her.] Pray let me introduce myself to you. My name is Cecily Cardew.

GWEN Cecily Cardew? [Moving to her and shaking hands.] What a very sweet name! Something tells me that we are going to be great friends. I like you already more than I can say. My first impressions of people are never wrong.

CECILY How nice of you to like me so much after we have known each other for such a comparatively short time. Pray sit down.

GWEN Perhaps this might be a favourable opportunity for my mentioning who I am. My father is Lord Bracknell. You have never heard of papa, I suppose?

CECILY I don't think so.

GWEN Outside the family circle, papa, I am glad to say, is entirely unknown. I think that is quite as it should be. The home seems to me to be the proper sphere for the man. And certainly once a man begins to neglect his domestic duties he becomes painfully effeminate, does he not? And I don't like that. It makes men so very attractive. Cecily, mamma, whose views on education are remarkably strict, has brought me up to be extremely short-sighted; it is part of her system; so do you mind my looking at you through my glasses?

CECILY Oh! not at all, Gwendolen. I am very fond of being looked at.

GWEN [After examining Cecily carefully through a lorgnette.] You are here on a short visit, I suppose.

CECILY Oh no! I live here.

Figure 1: The Notetaker Interface

4 Evaluation

To understand the effectiveness of the software, we designed a study to assess the system's usability in a simulated theatrical environment. We recorded four scenes using different volunteer actors from the Rollins College Theatre Department. Each scene was one to two minutes long and included approximately ten errors of varying types. These error types fell into three different categories that could be easily simulated: missed (Red), added (Yellow), and changed (Green). Each of the four scenes was played in succession, and five different veteran stage managers used our system to notate the errors in comparison to the scripts they were given. After they completed the line notes, we interviewed each participant to obtain qualitative data.

To analyze the data, the line notes from each participant were compared to a correct version and the total number of errors found was calculated. If a participant located an error but marked additional words, or if the error was found but categorized incorrectly, we counted the line note as correct. In a real theatrical setting, the actor would be able to use this feedback to understand that a line was said incorrectly.

While this study format does not allow us to simulate the fatigue experienced by the user during a two hour rehearsal, nor all the possible categories, it does allow us to efficiently test the system for learnability and ease-of-use.

5 Analysis and Discussion

Overall, the results from the study were positive. However, the testing pool was small, and we cannot assume that these findings will generalize without further study.

As a group, the participants were able to locate 70% of the 41 errors in the experiment. There was improvement in performance from the first scene (48% of errors found) to the final (84% of errors found). This increase in correctness can be attributed to two factors: participants becoming more familiar with the system and the speed at which the scenes were performed. The first scene was comedic scene which tends to be performed faster while the second was a dramatic scene which has a slower pace. Table 1 shows breakdown of performance by participant.

5.1 Usability and Strengths

During the interviews, the users identified several strengths of the software system. The users liked the ability to click to navigate from scene to scene instead of scrolling. Participant 2 talked about the ability to change scenes by

Table 1: Errors Found by Each Participant

Participant	Scene 1 10 Errors	Scene 2 10 Errors	Scene 3 10 Errors	Scene 4 11 Errors	Total 41 errors
P1	5	8	8	10	31
P2	4	8	7	10	29
P3	5	7	8	8	28
P4	6	8	7	9	30
P5	4	6	7	9	26
AVERAGE	48%	74%	74%	84%	70%

saying, “I like how you can flip back and forth between the scenes. I think it is a lot easier to be able to see all the scenes instead of having to scroll."

Second, the users liked the ability to use the buttons to change error categories. Due to the fact that the buttons were marked with the category name and color, the system helped reduce the cognitive load. Participant 3 said, “ [Changing the color of text in Word or Adobe] takes longer than this program did because this ... will automatically make it the color you want if you click on the right thing [button].”

Third, stage managers liked the ability to mark individual words with a double click which made it easier to mark small details while keeping up with the actors. Participant 2 remarked, “I think [error marking] is especially easier when it’s just one word. When you can just double click on it, it is a lot easier.”

While not explicitly tested in the study, the participants appreciated the software’s ability to export categorized line notes. This functionality is unique to this software, and it saves the stage manager time post-rehearsal. Participant 3 said “[Compiling notes] is what takes the longest for me, so it’s really helpful that I don’t have to do that manually.”

5.2 Critiques and Future Work

The biggest criticism of the software were some UI details. For example, the placement of a drop down menu which appears when the user highlights words blocks their vision of some of the remaining sentences. Additionally, several participants complained that yellow was hard to see on the button panel. Both of these complaints are easily solved in future versions.

Some participants were concerned that the software does not have a backup if the user’s computer is rendered unusable for a period of time. Several stage managers currently use Google Docs to ensure their notes are accessible at any time. This concern informs future work to include cloud backups.

Additionally, future work includes a study of the system during a full length production. This would provide a stage manager with the time to become an expert with the system and would allow us to study the system in a fast-paced, *in situ* use case.

6 Conclusion

In this paper we have discussed the design, implementation, and study of a software system to aid stage managers in taking line notes during theatrical productions. We have designed a system that not only allows for accurate line note taking, but provides a better user experience. Additionally, our system saves the user time by compiling notes for distribution. After completing a user study, we found that on average, the users were able to find errors with 70% accuracy using our system and all the stage managers who evaluated the software were enthusiastic about trying it during a full length production.

References

[1] Cuelist. *reinventing collaboration for theater and live events*, 2019 (accessed May 2, 2019). `https://www.thecuelist.com`.

[2] Y. Horiuchi, T. Inoue, and K. Okada. Virtual stage linked with a physical miniature stage to support multiple users in planning theatrical productions. In *Proc of the 2012 ACM Intl Conf on Intelligent User Interfaces*, IUI '12, pages 109–118. ACM, 2012.

[3] Propared. *Production Planning Software Revolutionizing How Organizations Manage Show Logistics and Streamline Communications*, 2019 (accessed May 2, 2019). `https://www.propared.com`.

[4] QLab. *QLab*, 2019 (accessed May 2, 2019). `https://figure53.com/qlab/`.

[5] S. Sinclair, S. Ruecker, S. Gabriele, and A. Sapp. Digital scripts on a virtual stage: the design of new online tools for drama students. In *Proc of the 5th IASTED Intl Conf on Web-Based Education*, WBE '06, pages 155–159. ACTA Press, 2012.

[6] Stage Write. *Capture Creativity with Stage Write*, 2019 (accessed May 2, 2019). `https://www.stagewritesoftware.com`.

[7] VirtualCallboard. *Online Stage Management and Production Management*, 2019 (accessed May 2, 2019). `https://www.virtualcallboard.com`.

Exploring Collaborative Talk Among African-American Middle-School Girls in the Context of Game Design for Social Change*

Jakita O. Thomas[1], *Yolanda Rankin*[2], *Quimeka Saunders* [3]
[1]*Auburn University, Auburn, AL 36849*
jnt0020@auburn.edu
[2]*Florida State University, Tallahassee, FL 32306*
yolanda.rankin@cci.fsu.edu
[3]*Spelman College, 350 Spelman Lane*
Atlanta, GA 30314
qsaunder@scmail.spelman.edu

Abstract

Computer Science education research establishes collaboration among students as a key component in learning, particularly its role in pair programming. Furthermore, research shows that girls, an underrepresented population in computing, benefit from collaborative learning environments, contributing to their persistence in CS. However, too few studies examine the role and benefits of collaborative learning, especially collaborative talk, among African-American girls in the context of complex tasks like designing video games for social change. In this exploratory study, we engage 4 dyads of African-American middle school girls in the task of designing a video game for social change, recording the dyads' conversations with their respective partners over an eight-week summer game design experience during the second year of what has now become a six-year study. Qualitative analysis of dyadic collaborative discussion reveals how collaborative talk evolves over time in African-American middle-school girls.

1 Introduction

Collaborative Learning has been presented as one of the many ways of addressing some of the known failures of traditional methods of instruction. Some of these known failures include low rates of retention, failure to transfer learning and inability of learners to apply knowledge flexibly. Collaboration, for the purposes of this research, has been defined as the joint effort of two individuals to complete a given/task or project. The act of collaboration allows groups to reveal conflict, negotiate meanings and uncover common ground that can serve to model the thinking necessary to carry out certain actions/task.

Computer Science (CS) education research espouses collaborative learning as beneficial for students learning how to program whether in an entry level or advanced CS course [10][7]. However, additional studies are needed to better understand how diverse populations of students master collaboration while developing key computational thinking skills. In this paper, we explore collaborative talk between dyads of African-American middle school girls engaged in game design for social change. This research poses the following question: *What conversational patterns do the African-American middle school girls generate as they engage in collaborative talk in the context of designing games for social change?* We address this research question by analyzing recorded video observations of dyads working together during the second year (or Season) of SCAT's two-week intensive summer game design experience.

2 Background

Collaborative learning has been presented as one of the many ways of addressing some of the known failures of learning by traditional methods of instruction [4][1]. Although, collaborative learning has been shown to improve some of the challenges of individual learning, putting two children together for group work will not necessarily ensure that they will profit from the interaction in a learning environment. Children benefit from the interaction to the extent in which they participate [6]. Teasley [8] found, when comparing dyads that talked with and dyads that did not (i.e., no-talk dyads), that dyads that talked generated better hypotheses than no talk dyads. In her study, dyads collaboratively programmed a sprite to move a certain way. She also characterized and categorized the collaborative talk, called utterances, that those dyads engaged in using verbal coding scheme. The coding scheme accounted for utterances, which were defined as individual message units that consisted of single sounds, sentence fragments and interruptions or complete sentences and nonverbal activity such as nods, shrugs, pointing, writing and control of the computer mouse [8]. The coding scheme included the following categories: procedural, command selection, plans, predictions, strategies, describes movement, references program, hypothesis, meta, off-task, questions experimenter, other, checks with partner, and resource management [8]. Procedural, command selection, plans, predictions and strategies together were called program generation utterances, which relate to the dyad creating the program. Describes movement, references program, and hypothesis were evidence evaluation utterances, using the program itself as evidence to evaluate or assess what is happening in the program or to explain the program's output. Meta, off task, questions experimenter, and other were described as general utterances. Checks with partner and resource management were described as dyad utterances because they characterized how utterances that dealt with the way the dyad engaged in the collaborative implementation of the program.

Looking at the individual verbal coding categories [8], procedural utterances concern the basic "how to operation of the task". Command selection names a selected program command. Plans states the intended plans of action (more than the selection of individual commands). Predictions make a specific prediction about an action or outcome. Strategies note that an individual command or sequence of commands affects the interpretability of the program. Describes movement identifies and/or counts sprites' movement as the program was executing. References program refers back to the program that was entered. Hypothesis states a (correct or incorrect) hypothesis about the effect of the command. Meta indicates an assessment of one's own understanding. Off task states information that was unrelated to the specific task, in this case, making a sprite (spaceship) move a certain way. Other indicated utterances that were inaudible, uninterpretable, or did not fit in any other category. Checks with partner addresses questions to a partner that serve to remind or clarify. Finally, resource management manages turn-taking or sharing the computer.

When engaging in collaborative talk, dyads are likely to use informal conversational like talk, which is talk that allows them to work out meanings and clarify, expound on and qualify ideas, known as expressive talk [5]. As dyads converse, the talk they produce becomes more formal, and use of appropriate vocabulary occurs instead of ambiguous terms like "it" or "that" [10]. Game design is inherently collaborative, with game design companies consisting of individuals working together on large teams to design and implement games over a certain number of years. As a result, the collaborative context that game design provides makes it an ideal context for examining collaborative talk, even in younger game designers (i.e., African-American middle school girls). Game design also involves the design, implementation, adaptation and assessment of algorithms, making it an ideal context to study a complex cognitive capability like computational algorithmic thinking (CAT). *CAT* is the ability to design, implement, and assess the design and implementation of algorithms to solve a

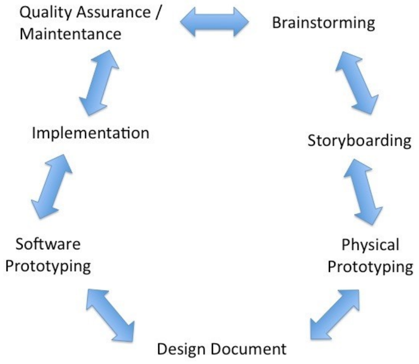


Figure 1: Game Design Cycle

broad range of problems [9].

The game design cycle consists of seven phases, which include brainstorming, storyboarding, physical prototype, design document, software prototype, implementation and quality assurance/maintenance, are iterative themselves (See Figure 1). Between each iterative phase beginning with storyboarding, playtesting occurs. Playtesting involves target players playing the game in different forms and providing feedback that informs the iterative design of the game [3]. During the brainstorming phase, dyads are to come up with as many game ideas as they possibly can and may present these ideas to an audience for feedback. After an idea is generated, dyads are required to explain their idea visually using paper-and-pencil in order to give a thorough explanation of their game. The images or series of "screenshots" are called storyboarding, which simulate the players' movement throughout the entire game while using few words to describe non-visual elements of the game (e.g., sound). During the physical prototyping phase, dyads create a playable prototype using craft materials, and their physical prototypes are playtested. After iteratively working through each prototype, the design document is drafted, which describes every aspect of the game. Next, each dyad creates software prototypes, which model the core gameplay and are playtested to help dyads make remaining design decisions. During the implementation phase, dyads implement their games, which are playtested after each iteration. Lastly, quality assurance is done and consists of making sure that the target audience has access to the software game and there are no lingering issues within the software prototype[3][9].

3 The SCAT Learning Environment

Supporting Computational Algorithmic Thinking (SCAT) is a longitudinal between-groups research project that explores how African-American middle-school girls develop CAT capabilities over time (i.e., three years) in the context of game design for social change. SCAT is also a free enrichment program designed to expose African-American middle-school girls to game design. Originally intended to span three years, but now in it's sixth year, participants develop CAT capabilities as they work in dyads to design more and more complex games that address issues or problems identified by the Scholars themselves. SCAT Scholars began the program the summer prior to their 6th grade year and have continued through their 11th grade year (i.e., June 2013 – present). For the first three years of the program, each year Scholars engaged in 3 types of activities: 1) a two-week intensive game design summer experience; 2) twelve technical workshops where Scholars implemented the games they designed using visual and programming languages in preparation for submission to national game design competitions; and 3) field trips where Scholars learned about applications of CAT in different industries and careers. Scholars also had several scaffolds in the learning environment to support them in the ways cognitive apprenticeship suggests including the facilitator, undergraduate assistants, the Design Notebook, and other Scholars[2][9].

4 Methodology

4.1 Setting and Participants

We have worked with 23 African-American middle school girls over the past six years, beginning the summer prior to their 6th grade year and continuing, now, through their 11th grade year. Note that 23 represents the total number of African-American girls who have participated in the program for the past 6 years but always in dyadic formation. In this study, we focus on the second year (or Season) of SCAT (June 2014), particularly the two-week intensive summer game design experience. Out of the 10 dyads in SCAT that Season, here we examine four target dyads (8 Scholars total): two dyads each consisting of two Scholars who were in the SCAT program for two consecutive years (called returning Scholars) who worked together both of those Seasons; one dyad consisting of two returning Scholars working together for the first time, and one dyad consisting of one returning Scholar and one Scholar who was new to SCAT at that time.

4.2 Data Collection and Analysis

We videotaped each dyad for six hours each day over the course of two weeks (10 weekdays), generating over 600 hours of video data for all 10 dyads. Our four target dyads represent over 240 hours of video data. Transcripts of the target dyads' conversations were generated from the videotaped observations. Each of the transcribed blocks were analyzed by two coders using Teasley's coding scheme [8]. Differences in categorization were settled via discussion. We expected those dyads of returning Scholars who worked together both Seasons would talk more, and that the character of those utterances would largely involve procedural, plans, and checks with partner. We expected the dyad of returning Scholars working together for the first time would talk less than the dyads who had worked together for two Season, and that the character of those utterances would largely involve command selection, meta, checks with partner, and resource management. Finally, we expected the dyad consisting of a returning Scholar and a new Scholar would talk even less than the dyad of returning Scholars working together for the first time because former dyad members would be less familiar with each other. We expected that the character of those utterances would largely involve command selection, meta, and checks with partner.

5 Results

Here, we present and characterize the discussions (or utterances) that our four target dyads produced as they engaged in designing and implementing their games. For those utterance types, we include excerpts of collaborative talk as representative examples of how our target dyads conversed as they moved through the game design cycle.

5.1 Procedural

Our analysis revealed new Scholar/returning Scholar dyads communicated more about tasks and the basic "how-to" operations for each phase of the game design cycle. The returning scholar was able to scaffold the novice scholar through the different phases when needed. For example, in the following excerpt, Team member A (the returning Scholar) and Team member B (the new Scholar) are constructing their physical prototype. Team member B is not clear about physical prototype construction, and Team member A, who has had prior experience constructing a physical prototype, explains:

```
Team member A: "They have to actually play it"
Team member B: "No I mean like on this"
Team member A: "I know"
Team member B: "For real?"
Team member A: "Yeah."
Team member B: "So what are we going to do?
                Like attach the thing to something?"
Team member A: "Yeah were going to attach a string to it."
```

We found that dyads with at least one returning Scholar also communicated procedural tasks when one partner was opposed to the way a task was being completed. For example, in this excerpt, the dyad is implementing their game using SCRATCH. They are trying to use the same screen in two different parts of the game, but only have the Play button show on one of those screens. Team member A (the returning Scholar) and Team member B (the new Scholar) disagree about how to accomplish that:

```
Team member A: "We can use the same play button -
                no not that play button"
Team member B: "No no no no no -- no seriously no"
Team member A: "You don't have to do that -
               stop stop stop stop"
Team member B: "No seriously no -
                its going to make me angry"
Team member A: "No just do that",
Team member A: "I know but it has to be on that page -
                so what you do is just make it hide"
Team member A: "just put that color to show at um -
                backdrop five".
```

5.2 Plans

Analysis reveals that new Scholar/returning Scholar dyads stated intended plans to be completed together, or plans were stated in such a way that one member asked questions to clarify a portion of a task. However, dyads of two returning scholars working together for the first time discussed plans to assure that each partner agreed before beginning to work the task. For example, below Team Member A and Team Member B (returning Scholars working together for the first time) discussed, during implementation, how they will display character descriptions and instructions for playing the game:

```
Team member A: "so that if they choose Cali,
                then how about we have the information
                about her...that screen."
Team member B: "yeah I know."
Team member A: "then like her family and friends and teacher.
                Then if they choose Johnson then they have
                his family, his friends and his teacher and
                we have, like, the principal and the nurse
```

```
                    – people."
Team member B: "and then tell them how to play".
```

Dyads with two returning Scholars did not state or discuss intended plans of action. Instead, using the Design Notebook, they worked simultaneously on different portions of one task, or one partner took the lead with no prior discussion of which partner would take the lead or which part of the task each partner would work on. Instead, those roles were negotiated in a seamless way without dialogue.

5.3 Checks with Partner/Meta

Both new Scholar/returning Scholar dyads as well as dyads with two returning Scholars working for the first time consistently checked with their partner for clarification to assure tasks were being completed accurately and in a way their partner liked. Below, Team member A and Team member B are designing levels of their game, discussing how players will be able to access the bonus level:

```
Team member A: "game...bonus level...games...bonus level...
                but that's only if they pass the game."
Team member B: "yeah, they pass the game then they'll
                get the bonus level...how about if they
                only get a certain amount of points
                they get the bonus level...like if the
                most amount of points you can get to pass
                the level is like 50 the least you can get
                is like 30 to pass the level...but if
                they get they 40 points then they get
                the bonus level but they don't get to
                go to the next round".
```

In dyads with returning Scholars working together for a second Season, partners rarely checked with each other, but focused instead on getting agreement from their partner on the process they were engaged in. For example, while creating their storyboards, Team member A checks in with her partner around the process for creating the storyboards.

```
Team member A: "Like we drew...we drew like the front page,
                but if you click on each one...like where
                would we go to...So are we supposed to like
                draw the game like somebody's playing it?"
Team member B: "mhm"
Team member A: "I know, but I still get confused that was
                like, a while ago".
```

6 Conclusion

We anticipated that dyads with two returning Scholar who had also worked together for two Season would engage in the most discussion, while dyads with one returning Scholar and one new Scholar would engage in the least discussion. The analysis of data of our four target dyads suggests otherwise. Instead, we found the opposite. The new Scholar/returning Scholar dyad engaged in collaborative talk most frequently with the returning Scholar (the more expert team member in terms of designing games for social change) helping scaffold the novice team member throughout the game design cycle. Further, while the returning Scholars working together for the first time, engaged in collaborative talk less than the new Scholar/returning Scholar dyad, they did engage in more collaborative talk than the returning Scholars working together for the second time. Perhaps most surprising was the finding that the dyads containing two returning scholars working together for the second time (or second Season) engaged in the least collaborative talk out of all of the target dyads with sporadic discussions throughout each phase of the game design cycle and with these dyads working mostly independently on each task and checking in with each other after completing a portion of a task. This suggests that the amount or duration of collaborative talk may not be the best indicator of collaboration for every task or process, especially when dyads work together for extended periods of time (in this case, into the second SCAT season, or more than one year). It would appear that these groups had developed practices over the course of the first Season that later supported them in engaging in game design for social change without a lot of collaborative talk during the second Season. Further, our analysis revealed that the utterance categories that showed up most in the data were Procedural, Plans, and Checks With Partner/Meta. Additionally, our analysis revealed that the utterance categories that showed up most in the data were uttered or enacted differently depending upon the type of dyad (i.e., two returning Scholars working together for a second Season, two returning Scholars working together for the first time, or one returning Scholar working with one new Scholar). For example, for dyads with one returning Scholar and one new Scholar, plans utterances focused on stating the intended plans together or asking questions about the plan for clarity. For dyads with two returning Scholars working together for the first time, plans utterances focused on ensuring that both members of the dyad agreed before proceeding with a plan of action. However, for two returning Scholars working together for a second Season, plans utterances were not stated or spoken at all. Instead, these dyads relied almost completely on the Design Notebook, and were able to divide the workload up in such a way that they could work independently, trusting that each partner would execute their plans in the ways they both intended. Future work includes conducting the same examination for the same groups using dyad video observations from the first Season of SCAT during the two-week intensive summer game design experience, where all Scholars participated in the SCAT program for the first time and where all dyads worked together for the first time.

Acknowledgements: We would also like to gratefully acknowledge the support of NSF (DR K -12 #1150098).

References

[1] Margarita Azmitia. Peer interaction and problem solving: When are two heads better than one? *Child Development*, 59(1):87–96, 1988.

[2] Allan Collins, John S. Brown, and Susan E. Newman. *Cognitive apprenticeship*, chapter 14, pages 453–494. Erlbaum, Hillsdale, NJ, 1989.

[3] Tracy Fullerton, Christopher Swain, and Steven Hoffamn. *Game Design Workshop: designing, prototyping and playtesting games.*

[4] T. D. Koschmann. Towards a theory of computer support for collaborative learning. *The Journal of the Learning Sciences*, 3:219–225, 1993.

[5] B. Latour. *Science in action: How to follow scientists and engineers through society.* Harvard University Press, 1987.

[6] Barbara Rogoff. *Apprenticeship in thinking: Cognitive development in social context.* Oxford University Press, 1990.

[7] O. Ruvalcaba, L. Werner, and J J. Denner. Observations of pair programming: Variations in collaboration across demographic groups. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, March 02-05:90 – 95.

[8] S. D. Teasley. The role of talk in children's peer collaborations. *Developmental Psychology*, 31(2):207 – 220.

[9] J. O. Thomas. The computational algorithmic thinking (cat) capability flow: A methodological approach to articulating complex cognitive skills and capabilities over time. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2018.

[10] L. Werner, J. Denner, and S. Campbell. Children programming games: A strategy for measuring computational learning. *ACM Transactions on Computing Education*, 4(4):22.

Assessing Ethics in a Computer Science Curriculum: Instrument Description and Preliminary Results[*]

Kevin R. Sanft
Department of Computer Science
University of North Carolina Asheville
1 University Heights
Asheville, NC 28804
ksanft@unca.edu

Abstract

Ethics and professional conduct are components of many undergraduate computer science curricula. Assessment of students' knowledge and conduct is important for evaluating ethics-related student learning outcomes and teaching effectiveness. We present a survey designed to assess a student learning outcome evaluated on five dimensions related to ethics and professional conduct. A rubric application is provided to categorize the responses into rubric levels. Preliminary assessments from an introductory programming course, a professional development course and a computer science capstone course are presented and compared. Assessing ethics is challenging due to time demands and the variety and nuance that realistic ethical dilemmas entail. Shortcomings such as the possibility of dishonest responses are discussed. Overall, the instrument provides a mechanism requiring minimal time commitment for assessing ethics-related student learning outcomes.

1 Introduction

Ethics and professional values have been a part of many computer science curricula for decades [2, 8]. However, these topics have received renewed attention in recent years due to the increasing role of technology in society and scandals at tech companies. From the use of social media to spread misinformation, poor data privacy policies, security breaches, concerns about automation and artificial intelligence, and many other challenges, ethical issues related to technology continue to make headlines, underscoring the importance of emphasizing these topics in the computer science curriculum.

In this work we will use the term *ethics* broadly to encompass ethics, professional values and professional conduct. Previous work on assessing ethics-related student learning outcomes has often focused on scenario-based approaches (e.g. [7, 11, 5, 9]). These typically involve a description of a challenging hypothetical scenario and the application of a rubric to categorize the response. Other work has emphasized the importance of aligning assessment instruments to the student learning outcomes and the difficulties in assessing hypothetical scenarios [6, 4]. Scenario-based assessments tend to be time-consuming and most approaches are susceptible to bias due to disingenuous responses.

In this work we present an assessment survey to evaluate an ethics-related student learning outcome across five predefined dimensions. The questions feature some answer choices that are designed to detect dishonest responses. Student responses are classified into rubric levels automatically. The assessment results are designed to provide useful metrics in a continuous improvement plan.

2 Student Learning Outcomes

The student learning outcome assessed in this work is based on an outcome in the ABET Criteria for Accrediting Computing Programs, 2018-2019, namely, “Students will make informed judgments in computing practice based on legal and ethical principles" [1]. The outcome is assessed on five dimensions that are adapted from the 2018 ACM Code of Ethics and Professional Conduct [3]: 1) Knowledge of Legal Issues, 2) Plagiarism, 3) Privacy and Confidentiality, 4) Societal Impact, and 5) Quality of Work. These dimensions, along with a matrix of rubric levels are available in Reference [10].

3 Instrument

The survey consists of fifteen multiple choice questions, including scenario-based, factual, and opinion questions. The questions and response options are listed below. The *Likert scale* choices are 1=Strongly Disagree, 2=Disagree, 3=Neutral, 4=Agree, and 5=Strongly Agree. The first question is “check all that apply", the rest are single option response.

1. Legally, which of the following factors are relevant in determining if a student can use a portion of copyrighted text or images in a homework assignment (check all that apply): a) You can never use someone else's copyrighted material; b) If the copyright holder is given attribution; c) That the use is for academic/educational purposes; d) The amount of material copied; e) The effect of the use on the copyright holder's income.

2. Suppose you sign a contract with a company to develop a portion of a large software product. The contract states that you are the author and not the owner of the software you write. This means: a) As the author, I am allowed to use my code only for my own personal future projects.; b) As the author, I am allowed to help other software companies using the same code.; c) Since I'm not the owner, I can't use the code anywhere else.; d) I'm not sure.

3. Copyright law applies to computer programs. (True, False, Don't know)

4. Copyright applies only when the copyright is registered. (True, False, Don't know)

5. Have you plagiarized (code or other writing) on a homework assignment (in any course) at the college level within the past 12 months? (Yes, No)

6. Have you ever plagiarized (code or other writing) on a homework assignment (in any course) at the college level? (Yes, No)

7. If you did not have time to finish a computer program for a homework assignment, would you copy programming code from another student? (Yes, No)

8. Imagine you work for a social media company. When do you think it is acceptable to use or share private and confidential user data (choose the best response): a) When it is convenient for software development purposes.; b) When the financial benefits of using the data are worth more than the expected cost of the risks.; c) When necessary for business purposes and it adheres to the company's privacy policy.; d) When necessary for business purposes but only with explicit user consent.; e) It is never acceptable to use or share confidential data.; f) None of these choices reflect my views.

9. It is important that computer scientists consider societal impacts when implementing systems. (Likert scale)

10. In the future, I will consider social needs and accessibility in design and implementation of computing systems. (Likert scale)

11. It is important that Computer Scientists create awareness and understanding of systems, their limitations, vulnerabilities, and opportunities. (Likert scale)

12. I sacrifice the quality of my work to meet deadlines and/or if other projects have higher priority. (a) Frequently, b) Rarely, c) Never)

13. How important is it for computer scientists to produce quality work (choose the best response): (a) Not particularly important; b) It can be important for the individual's success as an employee and for their company's success; c) In some cases, it can be a matter of life and death)

14. Are you aware that the Association for Computing Machinery has a "Code of Ethics and Professional Conduct"? (Yes, No)

15. It is important for computer scientists to pursue continuing education throughout their careers. (Likert scale)

4 Rubric Application

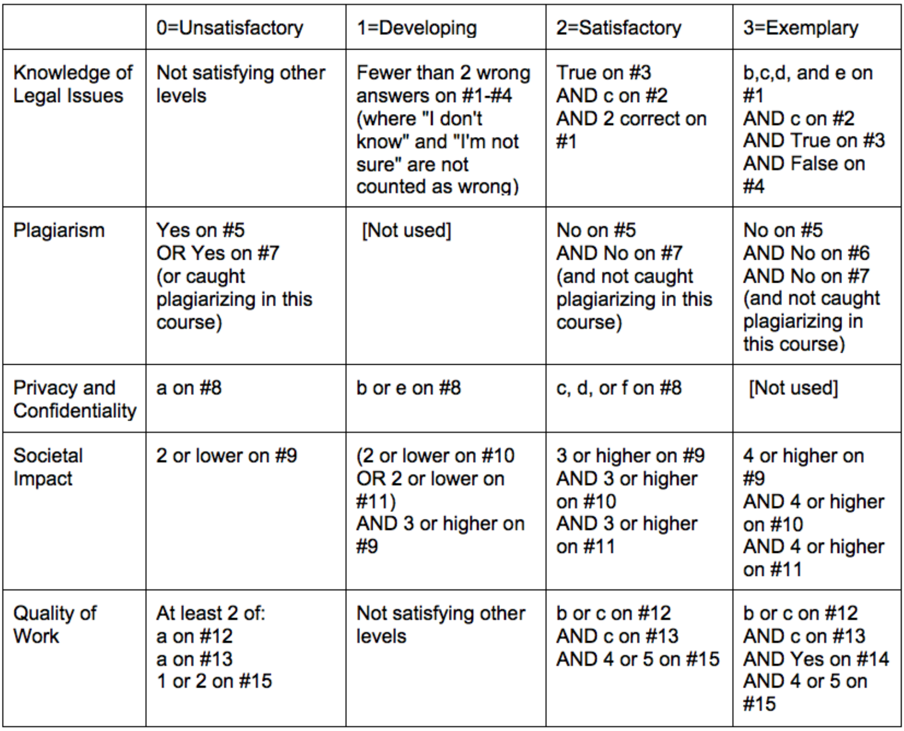
Figure 1 shows how the responses to the survey are mapped to the rubric levels. A simple script aggregates the results.

5 Preliminary Results and Discussion

The survey was administered via Google Forms in a single semester to three computer science courses: an introductory programming course open to majors and non-majors (majors were primarily first-year students, while non-majors were typically further in their program), a mid-level professional development course, and a capstone project course. Rubric application results are shown in Table 1. All students at our institution complete a Humanities program that includes ethics-related topics; we did not consider students' prior exposure to ethics via the Humanities program or other courses in our assessment. The professional development course was created as part of a recent curriculum redesign. The ethics-related student learning outcome was introduced with the new curriculum, therefore, it was not assessed in the previous curriculum.

5.1 Discussion and Conclusion

Several response choices are designed to detect answers that are disingenuous or that fail to capture the nuance of ethical issues. For example, "Never" sacrifice work quality on Q12 or on Q8 saying "It is never acceptable to use or share confidential data", which neglects legitimate business purposes (e.g.



	0=Unsatisfactory	1=Developing	2=Satisfactory	3=Exemplary
Knowledge of Legal Issues	Not satisfying other levels	Fewer than 2 wrong answers on #1-#4 (where "I don't know" and "I'm not sure" are not counted as wrong)	True on #3 AND c on #2 AND 2 correct on #1	b,c,d, and e on #1 AND c on #2 AND True on #3 AND False on #4
Plagiarism	Yes on #5 OR Yes on #7 (or caught plagiarizing in this course)	[Not used]	No on #5 AND No on #7 (and not caught plagiarizing in this course)	No on #5 AND No on #6 AND No on #7 (and not caught plagiarizing in this course)
Privacy and Confidentiality	a on #8	b or e on #8	c, d, or f on #8	[Not used]
Societal Impact	2 or lower on #9	(2 or lower on #10 OR 2 or lower on #11) AND 3 or higher on #9	3 or higher on #9 AND 3 or higher on #10 AND 3 or higher on #11	4 or higher on #9 AND 4 or higher on #10 AND 4 or higher on #11
Quality of Work	At least 2 of: a on #12 a on #13 1 or 2 on #15	Not satisfying other levels	b or c on #12 AND c on #13 AND 4 or 5 on #15	b or c on #12 AND c on #13 AND Yes on #14 AND 4 or 5 on #15

Figure 1: The criteria for mapping responses from the instrument to the rubric levels.

billing). One may be able to use these and perhaps additional questions to create a *reliability score* but this was not explored in this work. On plagiarism, the preliminary results were interesting. In the intro course, four students had been caught plagiarizing earlier in the semester, yet all four of them answered that they had not plagiarized! Conversely, in the professional development seminar, where ethics were discussed, two students admitted to plagiarizing (in other courses). Our university maintains a repository of academic honesty violations which we intend to use to supplement the survey responses.

In evaluating the assessment results, “success" is institution and program dependent and should be based on the curriculum’s learning outcomes. For the baseline results reported here, preliminary conclusions might be that our program should increase emphasis on dimensions 1 (Knowledge of Legal Issues), 3 (Privacy and Confidentiality) and 5 (Quality of Work), which could be integrated into a continuous improvement plan. The instrument described in

Dimension	Rubric Level	Overall (N=54)	Intro (N=30)	Seminar (N=17)	Capstone (N=7)
Knowledge of Legal Issues	0	15%	10%	18%	29%
	1	50%	40%	65%	57%
	2	30%	40%	18%	14%
	3	6%	10%	0%	0%
Plagiarism	0	13%	20%	6%	0%
	2	2%	0%	6%	0%
	3	85%	80%	88%	100%
Privacy and Confidentiality	0	2%	0%	6%	0%
	1	50%	53%	41%	57%
	2	48%	47%	53%	43%
Societal Impact	0	4%	3%	6%	0%
	1	4%	3%	6%	0%
	2	26%	30%	24%	14%
	3	67%	63%	63%	86%
Quality of Work	0	0%	0%	0%	0%
	1	70%	80%	59%	57%
	2	11%	17%	0%	14%
	3	19%	3%	41%	29%

Table 1: Baseline assessment results for three computer science courses: an intro programming course open to majors and non-majors, a professional development course, and a capstone course. Percentages may not add to 100% due to rounding. Dimensions are described in Reference [10]. Rubric levels are 0=Unsatisfactory to 3=Exemplary as applied in Figure 1.

this work provides a convenient mechanism for assessing ethics-related student learning outcomes. Future work will explore longitudinal results and the effects of specific continuous improvement plan actions.

Acknowledgements

The author thanks Charley Sheaffer of the UNC Asheville Department of Computer Science for his contributions to the learning outcome, dimensions and rubrics.

References

[1] ABET. Criteria for accrediting computing programs, 2018-2019. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2018-2019/.

[2] Richard H. Austing, Bruce H. Barnes, Della T. Bonnette, Gerald L. Engel, and Gordon Stokes. Curriculum '78: Recommendations for the undergraduate program in computer science&mdash; a report of the acm curriculum committee on computer science. *Commun. ACM*, 22(3):147–166, March 1979.

[3] Association for Computing Machinery. Acm code of ethics and professional conduct. https://www.acm.org/code-of-ethics.

[4] Ursula Fuller and Bob Keim. Assessing students' practice of professional values. *SIGCSE Bull.*, 40(3):88–92, June 2008.

[5] Mary J. Granger, Elizabeth S. Adams, Christina Björkman, Don Gotterbarn, Diana D. Juettner, C. Dianne Martin, and Frank H. Young. Using information technology to integrate social and ethical issues into the computer science and information systems curriculum: Report of the iticse '97 working group on social and ethical issues in computing curricula. *SIGCUE Outlook*, 25(4):38–47, October 1997. Chairman-Little, Joyce Currie.

[6] Matthew W. Keefer, Sara E. Wilson, Harry Dankowicz, and Michael C. Loui. The importance of formative assessment in science and engineering ethics education: Some evidence and practical advice. *Sci Eng Ethics*, 20:249, 2014.

[7] Keith Miller. Integrating computer ethics into the computer science curriculum. *Computer Science Education*, 1(1):37–52, 1988.

[8] The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society. Computer science curricula 2013. https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf.

[9] D.B. Parker. *Ethical Conflicts in Computer Science and Technology.* AFIPS Press, 1981.

[10] Charley Sheaffer and UNC Asheville Department of Computer Science, 2018-2019. https://drive.google.com/file/d/1wTsOP4M-5ORLUWR9fx1g--6bb_Fvkzw8/view?usp=sharing.

[11] L.J. Staehr and G.J. Byrne. Using the defining issues test for evaluating computer ethics teaching. *IEEE Transactions on Education*, 46(2):226–234, 2003.

Reflective Writing Through Primary Sources[*]

Valerie Summet
Mathematics and Computer Science
Rollins College
Winter Park, FL 34761
vsummet@rollins.edu

Abstract

In this paper, we present a series of reflective writing assignments. In contrast to previous use of reflective writing in computer science, this work aims to provide a discussion of ways of incorporating reflective writing and primary sources into higher level courses and providing a platform for reflection on a large scope of events including the computer science major, the undergraduate learning experience, and plans for both education and careers. We give examples of the primary sources, the overall structure of the assignments, and some student reactions to them.

1 Background

Reflective writing is an important skill in an undergraduate education. Reflective writing encourages critical thinking, assists in developing student inquiry, and may help students understand content material or larger patterns in their learning. In CS, reflective writing has most often been used to reflect on a small, practical activity such as a specific exercise. The purpose of this paper however, is to examine how students reflect on their undergraduate education and, in particular, their major in Computer Science. In this paper, we present a series of writing assignments designed to expose CS student to primary sources, facilitate a close reading of those sources, and encourage students to reflect on the connections between the reading and their experiences as a computer science major at Rollins College.

2 Related Work

In CS, reflective writing has been highly targeted. For example, Stone and Madigan [7] integrated reflective writing into two specific assignments: a teardown of a PC in an architecture class and in case studies during a network security class. They noted, “The early papers were more like research papers where the students repeated the concepts but did little in tying the material to their own experiences. As the term progressed the reflections became less research and fact-oriented and more reflective and personal.” George [3] used reflective journal in a Data Structures and Algorithms course. These journal entries were specifically tied to course content and programming assignments and asked students to reflect on their understanding and learning of those topics. Fekete, et al. [2] used reflective diaries in CS1 to encourage students to develop regular study habits and learning patterns. The diary entries asked concrete questions such as what new things the students had learned this week, what readings they had completed, how they did on last week’s plan, and what their plan for the coming week was. VanDeGrift [9] studied writing in the context of pair programming. Specifically, she designed project reports which were completed after three different pair projects based around reflective writing. “Reflective and personal” writing was an explicit goal of the current project, but the desire was to decouple and expand the reflection away from course-specific programming or content.

Reflective writing has also been used as a feedback mechanism to the course instructor in Computer Science (e.g. [6]) but instructor feedback was not the goal of this project.

3 Assignment Structure and Methodology

For each reflective writing assignment, a reading was chosen which served to highlight a different topic in computer science and require the students to make connections to that theme in their education. On the day an assignment was due, the class also had a discussion about the reading and student responses.

3.1 Structure

Each assignment was scaffolded to take the students through three levels of questions leading to deeper reflection. The assignment began with questions encouraging close reading, for example, asking the students to find quotes that they found funny, insightful, or confusing. Then the students had a series of questions which tied the reading to a theme or topic. These two types of questions provided a low-stakes way of beginning class discussion and served as a starting point for students to share their opinions on value-neutral questions. The assignment then transitioned to the reflective questions.

The reflective questions at the end of the assignment specifically ask the student to reflect on their education thus far and the choices they have made. Students are also asked to think critically about the entirety of their education and how they will actively manage their careers and lives in the future.

3.2 Primary Sources

While there are many possible choices for primary sources, the readings below were chosen to expose the students to a wide spectrum of writing: women authors, historical sources as well as modern writing, and peer-reviewed scholarly articles as well as accessible pieces written for a lay audience.

1. Ada Lovelace's *Note A* [5]. This writing is a historical primary source which few students had interacted with. Most students found it challenging due to the language (typical of the 19th century), its mathematical foundations, and the wildly different vocabulary concerning the "computational" aspect of her writing.
2. Vannevar Bush's article *As We May Think* [1]. Again, this is a historical primary source which has had vast implications in information management and the development of the World Wide Web as we know it today.
3. Nancy Leveson's article *Medical Devices: The Therac-25* [4]. This article provides an in-depth dive into a medical disaster brought about by poor software engineering and software bugs. In spite of its length (35 pages), this is one of the most popular readings and discussion sessions.
4. D.A. Winsor's *Communication Failures Contributing to the Challenger Accident: An Example for Technical Communicators* [10]. This article highlights the importance of communication with both your peers and managers who may not have the same technical background to facilitate a common understanding.
5. Ellen Ullman's book chapter *New, Old, and Middle Age* [8]. In this piece, Ullman reflects on the mental effort required to stay educated in the field of computer science. As she approaches middle-age, she questions if she has the drive to learn the skills younger people graduate with. Unsurprisingly, the students relate to the mental exhaustion of always learning new things. Again, this has proved to be a very popular discussion piece.

3.3 Topics

Each primary source serves as the basis for one assignment on a specific topic. Some sources correspond to the topic better than others, but the desire was for the student to begin to reflect on this topic in relation to the reading.

There are several questions in which the students must tie the topic into the reading. Some of the topics included creativity, lifelong learning, career planning and management, ethics and ethical work, prioritization and perspective, communication, and teamwork.

Some assignments focused on only one topic while others incorporate multiple topics. Many class discussions tied into multiple topics. For example, communication between technical experts and lay people was a theme that was discussed in Bush's, Leveson's, and Winsor's writing.

4 Data, Observations, and Reactions

During the Spring 2018 semester, student writings were collected and analyzed with IRB approval. Specifically, a total of 101 unique writings were collected from 24 students over five assignments given during the Senior Capstone course which students traditionally take in their final semester.

In general, the students liked the readings and class discussions. From anonymous course evaluations, we received the following comments:

- *I also looked forward to the class discussions about the readings. There are not enough discussion based classes in the computer science program.*
- *I really enjoyed having the reading assignments within this class because it helps broaden the work of computer science majors.*
- *...the discussions we had during class were actually interesting.*
- *I really liked the inclusion of a writing component this semester, because I think that is hugely beneficial in the job setting...*
- *Reading about the programming cases made me interested in some of the policy. Not enough to go into law, but definitely interested enough to keep up with regulations.*

The majority of negative comments were focused on the mechanics of the assignments and included things such as grading and due dates.

Moreover, we saw evidence that the students were using the primary source readings and the writing assignments as venues for personal reflections. The following quotes showcase some reflections:

On teamwork: *"I believe a pretty universal aspect of teamwork for computer scientist is documentation of code. API's and libraries of code need documentation so people outside of those who created the documents can understand and implement them. Reflecting on this question made me realize how collaborative the computer science field is. ... One of the most interesting parts about teamwork and CS is how it looks differently than teamwork for other fields."*

On begin prepared for the responsibilities outlined in the ACM Ethics Code: *"I worry about responsibility 2.5, 'Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of* *possible risks.' I don't feel comfortable exhaustively testing my code for all errors. I can detect some errors along the development of an application but exhaustively testing a program from all possible errors worries me."*

On communication: *"I'd love to offer some great examples of times when I was able to communicate some complex Computer Science idea to a lay person, but I find myself struggling to communicate on basic levels sometimes. This is something I am working on."*

On keeping skills up-to-date: *"I have to be honest here. I have been languishing in learning fatigue for a few semesters now. The only thing keeping me going is my friends, my pride, and caffeine."*

On how learning changed over time: *"When I first started at Rollins I wanted to learn every little detail of everything that we touched on in any class. However, I quickly found out that there is too much technology and its being updated too quickly to have any possibility of 'keeping up.' So I tried to focus a lot more on learning skills that I can apply regardless of what language or technology I'm currently using. Things like problem solving techniques, algorithm analysis, or good coding practices will most likely remain useful no matter what new language I use. Then when it comes to learning a new language or technology I focus on learning what I need to accomplish a specific goal."*

On not understanding new things: *"This [reading] made me feel like I was not alone. Often, I feel like I am scrambling to know what is going on around me and want to make others realize I know what is going on."*

These quotes demonstrate the student candor and depth of thought which went into their writings. Students were surprisingly willing to share these thoughts with us through their writings and sometimes shared them during class discussions as well. Moreover, in class the students sometimes shared moments of cohort bonding. For example, Ullman's writings in particular helped students of all genders discuss imposter syndrome and how even the "best" students experienced it to some degree.

5 Future Work and Conclusion

One of the unexplored themes to arise from the analysis of the students' writings has been program deficiencies. We have been able to identify several areas of our curriculum that need enhancement or redesign based on the students' reflections. These type of insights provide an opportunity for instructor and department reflection on missing components and opportunities to restructure certain courses to meet these needs, and further analysis in this area is planned.

These reflective writing assignments have shown promise in encouraging students' critical thinking, inquiry, and self-reflection. They have also provided students with valuable context provided by primary sources and historical documents and allowed the students to reflect on the entirety of their education.

References

[1] Vannevar Bush. As we may think. *Atlantic Monthly*, (176):101–108, 7 1945.

[2] Alan Fekete, Judy Kay, Jeff Kingston, and Kapila Wimalaratne. Supporting reflection in introductory computer science. In *Proc of the 31st SIGCSE Technical Symp on Computer Science Education*, SIGCSE '00, pages 144–148. ACM, 2000.

[3] Susan E. George. Learning and the reflective journal in computer science. In *Proc. of the Twenty-fifth Australasian Conf. on Comp. Sci.*, ACSC '02, pages 77–86. Australian Computer Society, Inc., 2002.

[4] Nancy Leveson. *Safeware: System Safety and Computers*, chapter Appendix A - Medical Devices: The Therac-25 story. Addison-Wesley, 1995.

[5] Ada Lovelace. Sketch of the analytical engine invented by charles babbage: Note A. `http://www.fourmilab.ch/babbage/sketch.html`. Accessed: 2019-04-24.

[6] Jeffrey A. Stone. Using reflective blogs for pedagogical feedback in CS1. In *Proc of the 43rd ACM Technical Symp on Computer Science Education*, SIGCSE '12, pages 259–264. ACM, 2012.

[7] Jeffrey A. Stone and Elinor M. Madigan. Integrating reflective writing in CS/IS. *SIGCSE Bull.*, 39(2):42–45, June 2007.

[8] Ellen Ullman. *Close to the Machine: Technophilia and Its Discontents.* City Lights Publishers, 2001.

[9] Tammy VanDeGrift. Coupling pair programming and writing: Learning about students' perceptions and processes. In *Proc of the 35th SIGCSE Technical Symp on Comp Sci Ed*, SIGCSE '04, pages 2–6. ACM, 2004.

[10] D.A. Winsor. Communication failures contributing to the Challenger accident: an example for technical communicators. *IEEE Transactions on Professional Communication*, 31(3):101–107, 9 1998.

Mapping and Securing User Requirements on an IoT Network[*]

J. Delpizzo, R. Honeycutt, E. Spoehel, S. Banik
Department of Cyber and Computer Sciences
The Citadel, Charleston, SC 29409
{jdelpizz,rhoneycu,espoehel,baniks1}@citadel.edu

Abstract

The number of IoT (Internet of Things) devices connected in the Internet has been increasing rapidly. Each of these devices are manufactured by different vendors and provide multiple options for connections. When these devices are connected with default settings to create a user centric IoT network, it exposes a lot of vulnerabilities. In this research we propose a framework that will create a 3-layered abstraction in the IoT network to identify the user requirements on the IoT devices and explore all possible connections in an IoT network. Our goal is to provide a mapping of the user requirements on the IoT network and ensure that the mapping is secured.

1 Introduction

Internet of Things (IoT) is a set of devices that are connected to the Internet and offer different services to the users. These devices that include smart phone, smart watch, smart switch, smart thermostat, smart refrigerator have become seamlessly integrated into our everyday lives. Network created with these devices have expanded to smart homes, smart cities, medical centers, and corporate offices. It is estimated by 2025, there will be 50 billion connected smart home devices, making up a 6 trillion dollar industry [5]. Fundamentally it will change the landscape of the Internet and increase the surface area for malicious attackers. Vendors have introduced different types of IoT devices that provide different types of connectivity. While the product meets the functionality requirements, it may not always meet security requirements. A unsuspecting user may purchase IoT devices to add convenience to their lives. However, when these devices are connected to the home IoT network, they open major gaps in the existing network that can be exploited to expose private information or create a botnet, like the mirai botnet, to infect other systems.

We propose a framework of three layers in an IoT network. For a set of IoT devices, the physical layer identifies all possible connections among these devices, the requirement layer identifies all the connections required by the user for using these devices, and the logical layer maps the user requirements on the physical network. First the user needs to identify the devices that they would like to have connected in their home IoT Network. We propose an algorithm that will identify all the physical connections between the user selected devices. Next we get the requirement of user for connecting these devices. For example, the user wants to use the smart phone to control the temperature. We propose an algorithm that maps the requirement of the user on the physical network of IoT devices. We identify the physical connections that are used to map the user requirements. Our goal is to design the secured logical layer that will map the user requirements on the physical network and restrict the connections that are not used in the mapping.

The remainder of this paper is organized as follows. In Section 2, we present the recent research in IoT security from the literature. Section 3 describes our System Model. Section 4 outlines the proposed algorithms. In Section 5, we discuss our experimental testbed and results. Section 6 provides ideas for future work and concludes the paper.

2 Literature Review

Lee et. al. [1] has described different types of IoT devices and their networking technologies and outlined different types of attacks on these devices. The authors have proposed some recommendations for securing the smart home that include robust user authentication, device authentication, network monitoring, secure key management, and physical protection. Kolias et. al. in [2] have created IoT security labs that include a light switched system, remote plant watering system, and system to automatically control devices. Security flaws encountered by the authors are insecure web applications, wireless protocol vulnerabilities, architectural vulnerabilities, and limits on resource available on the IoT devices for security. In [3], the authors explains the holistic aspects of smart home security. This requires security in depth, integrated hardware, software, network, and application layers. Some core issues addressed by the authors are on the computational and energy constraints of hardware, access control, routine software updates, protocol diversity, and loopholes in machine learning at the application layer. Jjung et. al. [4] have proposed to divide an IoT network into the Task, Interface, and Interaction layers. Semantic dependency links are generated to create the Interface layer that connects entities across the task layer. Once the interface layer is created, the authors proposed algorithm that searches for semantically matching interfaces.

3 Proposed System Model

Our proposed system model creates a three layered abstraction (Figure 1) in an IoT network. The bottom layer which is the Physical Layer is composed of all the physical connections present in the IoT network for a given set of IoT devices. The top layer that keeps track of all of the connections that the user wants is called the Requirements Layer. The middle layer that is called the Logical Layer is in essence the Requirements layer superimposed onto the Physical Layer.

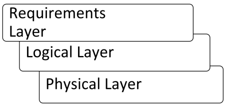


Figure 1: Diagram of Layer Model

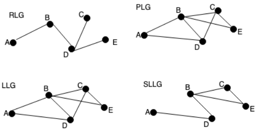


Figure 2: Graph example of Layer Model

Each layer is represented with a graph (Figure 2) where vertices are devices and edges are the connections between those devices. In the Physical Layer Graph (PLG), the graph describes the connectivity between the devices. A variety of common protocols that characterize the edges at this level include Wi-Fi (IEEE 802.11), Bluetooth, Zigbee, and Ethernet. In the Requirement Layer Graph (RLG), we connect two vertices with an edge if that connectivity is required by the user. The Logical Layer Graph (LLG) includes all the vertices from RLG but it includes only the edges that are required to map the RLG into PLG. For any edge that exists in the RLG we find all of the paths from those two vertices in the PLG, those paths are then added to LLG. The point of constructing the LLG is to remove or block unnecessary connections that exists in PLG which might be exploited by attackers. After creating the LLG, we prune the LLG to create a Secured Logical Layer Graph (SLLG). For constructing the SLLG, we analyze the security of each path in the LLG, and keep the path that is secured for each mapping of RLG onto PLG.

4 Proposed Algorithms

In this section we present two algorithms. Algorithm 1 constructs PLG for a given a set of IoT devices. Algorithm 2 maps each edge in RLG into PLG and constructs LLG.

Algorithm 1 Physical Mapping Algorithm

```
Require: : Set of IoT Devices
Ensure: : Database for constructing PLG
procedure MAP(Network)                    ▷ Mapping all connections in a network
    Scan all devices adjacent to host and add all links to a database
    Request all discovered devices to scan
    Request scan data
    Update database for newly discovered devices
    while new devices discovered do
        Request new device scan
        Request scan data from device
        Update Database
    Return Database
```

Algorithm 1 starts with scanning all devices connected to the host machine and records information for connected devices in a database. Then for each device in the database it requests, scans, and records the connection information in the database. The algorithm stops when all the devices in the database are scanned. The approach is very similar to a breath first search.

Algorithm 2 RLG to PLG Mapping Algorithm

```
Require: : RLG, PLG
Ensure: : LLG
create LLG(RLG, PLG)
     for(j = 0 to M)
          for(i = j + 1 to N)
               if RLG[j][i] = 1
                    firstnode.list = j
                    createPaths(j,i,list)
               LLG[j][i] = list
createPaths(j,i,list)
     if PLG[j][i] = 1
          return nextnode.list = i
     else
          for (k = j + 1 to N)
               if PLG[j][k] = 1
                    nextnode.list = k
                    createPaths (k, i, list) =0
```

Algorithm 2 for mapping RLG to PLG traverses the adjacency matrix of RLG to discover all paths in RLG. Then it passes two nodes and the list to the procedure *CreatePaths*. *CreatePaths* checks if there is the same connection in the PLG. If the path is found in PLG then it is added to the LLG. The complexity of Algorithm 2 is $M * N^2$ where M is number of nodes in RLG and N is the number of nodes in PLG.

5 Experiment

To test Algorithm 1 for constructing PLG, we created a home IoT network with a computer, a Samsung Galaxy phone, an Echo Dot, a Google Home, a Nest Thermostat, a router, a Smart Hub and a bulb. Using a python script utilizing bluetool for bluetooth scanning and the bash command iwlist for Wi-fi scanning, the program actively listened for the command to scan on each virtual machine, the virtual machine controller (VMC) gave the command and all the devices used Secure File Copy (SCP) to transfer their portion of the database to the VMC which then combined the database parts to form the PLG [6]. The PLG is shown in Figure 3. A snapshot of the database is shown in Figure 5.

To demonstrate the effectiveness of Algorithm 2 for mapping RLG to PLG, we use PLG constructed by Algorithm 1. For the RLG we used different types of connectivity requirements in IoT network that users may need. An example of RLG is shown in Figure 4. A snapshot of the output of Algorithm 2 is shown in Figure 6.

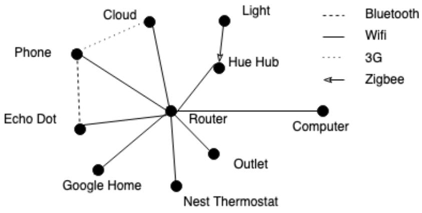


Figure 3: PLG Example

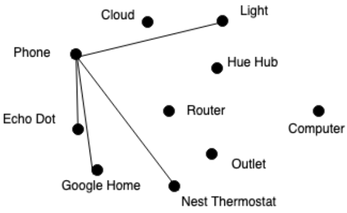
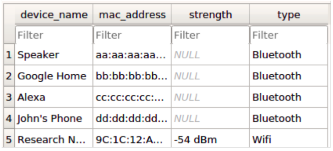


Figure 4: RLG Example

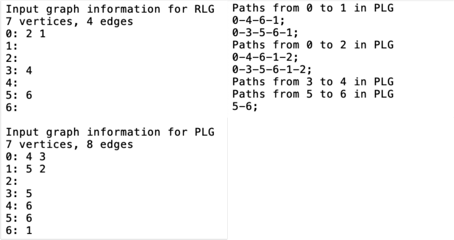
6 Conclusion

The number of IoT devices are growing rapidly. Many of these devices leave a plethora of security and privacy concerns for users when they are connected to their home network. In our research we propose a three layered abstraction of IoT Network - Physical Layer, Logical Layer and Requirement Layer. We propose algorithms for constructing Physical Layer Graph (PLG) and mapping of Requirement Layer Graph (RLG) onto Physical Layer Graph for IoT Network. Our ultimate goal is to create a Secured Logical Layer Graph (SLLG) of IoT



	device_name	mac_address	strength	type
	Filter	Filter	Filter	Filter
1	Speaker	aa:aa:aa:aa...	NULL	Bluetooth
2	Google Home	bb:bb:bb:bb...	NULL	Bluetooth
3	Alexa	cc:cc:cc:cc:...	NULL	Bluetooth
4	John's Phone	dd:dd:dd:dd...	NULL	Bluetooth
5	Research N...	9C:1C:12:A...	-54 dBm	Wifi

Figure 5: Output Algorithm 1



```
Input graph information for RLG   Paths from 0 to 1 in PLG
7 vertices, 4 edges               0-4-6-1;
0: 2 1                            0-3-5-6-1;
1:                                Paths from 0 to 2 in PLG
2:                                0-4-6-1-2;
3: 4                              0-3-5-6-1-2;
4:                                Paths from 3 to 4 in PLG
5: 6                              Paths from 5 to 6 in PLG
6:                                5-6;

Input graph information for PLG
7 vertices, 8 edges
0: 4 3
1: 5 2
2:
3: 5
4: 6
5: 6
6: 1
```

Figure 6: Output Algorithm 2

network that will satisfy the user requirements, keep the secured connections and remove the unnecessary connections from the IoT Network. The next step in our research is to assess the vulnerabilities of each path in LLG and keep the least vulnerable path to create the SLLG. Then we will keep adding new devices to our IoT testbed to test the adaptability and scalability of our solution.

References

[1] C. Lee, L. Zappaterra, K. Choi, H. Choi, "Securing smart home: Technologies, security challenges, and security requirements" in *IEEE Communications and Network Security and Privacy*, 2014, pp. 67-72.

[2] C. Kolias, A. Stavrou, J. Voas, I. Bojanova,R Kuhn, "Learning Internet-of-Things Security Hands-On" in *IEEE Communications and Network Security and Privacy*, Vol 14, Issue 1, Jan.-Feb. 2016, pp. 37-46.

[3] E. Fernandes, A. Rahmati, K. Eykholt, A. Prakash, "Internet of Things Security Research: A Rehash of Old Ideas or New Intellectual Challenges?" in *IEEE Communications and Network Security and Privacy*, Vol 15, Issue 4, 2017, pp.79-84.

[4] J. Jung, S. Chun, K. Lee, "Hypergraph-based overlay network model for the Internet of Things" in *IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 104-109.

[5] E. Bertino, (2016) "Data Security and Privacy in the IoT" in *19th International Conference on Extending Database Technology* , March 2016.

[6] A. Aleksandrov (2017) "Bluetool" (Version 0.2.3) [Library]. Python Software Foundation.

Ranking Privacy of the Users in the Cyberspace[*]

Adrian Beaput, Shankar Banik, Deepti Joshi
Department of Cyber and Computer Sciences
The Citadel, Charleston, SC 29409
{abeaput,baniks1,djoshi}@citadel.edu

Abstract

When websites are accessed, online shopping is done, or social media is used, a user's privacy is assumed to be protected. However, the Internet provides a widely available and easily accessible way to discover a vast amount of personal data. If this information is aggregated to create a digital footprint for the user, there are many security concerns that arise with this to include identity theft and other malicious intents. In our research, we build a user profile based on the personal information of a user through Twitter, and additional information collected using a web crawler. The web crawler keeps track of the websites that contain user attributes. Using the web ranks of these websites, and the number of counts of the attributes, we propose a formal and novel model to rank the privacy of a user in the cyberspace. Our proposed model of privacy assigns privacy ranking of each user between 0 and 1 with 0 being more private.

1 Introduction

The Internet provides a widely available and easily accessible way to discover a vast amount of personal data. If this information is aggregated to create a complete profile of an individual, or a digital footprint, there are many security concerns that include identity theft and other malicious intents. However, with the ubiquitous use of Internet and our digital footprint ever increasing, understanding how exposed we are online is a critical problem to solve. When information is posted online on different websites open to the public, online shopping is done, or social media is used, a user assumes that their privacy is protected.

Privacy is defined as the state of being free from the public eye; however, when information is posted on the Internet it loses its state of privacy. Privacy on social media then translates to being the concealment of information that reveals personally identifiable information (PII) which include Social Security Number, Passport Number, Driving License Number, etc. Some of these PII are directly made available by the users or can be derived from information.

In this research, our goal is to build a digital footprint of a user and then analyze and rank the user's privacy using our own proposed novel privacy metric (see Section 3) on the scale of 0 to 1, with 0 being most private, and 1 being least private. To achieve this, we build a user profile based the personal information that an individual makes available on the web. The initial data points collected using a user's social media profile, are ran through a web crawler. The web crawler provides new information about the user, which also provides a feedback loop into the crawler to collect additional information. Based on the information collected and aggregated using the web crawler, we propose a formal model to rank the privacy of a user in the cyberspace. Industry and government agencies that deal with sensitive information can use our model to calculate the privacy of the users. This information will be useful when users are given access to sensitive information. If a user is less private, then the probability of leaking sensitive information through that user will be higher.

Our paper is organized as follows. Section 2 provides literature review. In section 3, we describe our proposed privacy ranking metric. Section 4 provides our system model. In section 5, we describe our experimentation setup and present the obtained results. Finally, section 6 outlines the conclusion and future work.

2 Literature Review

Cheung et al in [1] have evaluated the privacy risk of user shared images. They have explored the possibility of whether or not widely accessible user images invade the privacy of a user. This is done by de-anonymizing social graphs and online profiles. Zafarani et al in [4] have presented a behavioral-modeling approach for connecting users across social media sites. They have proposed a methodology called MOBIUS that identifies user's unique behavioral patterns, constructs features that exploit information redundancies, and employs machine learning for effective user identification. Narayanan et al in [3] have looked at de-anonymizing users across heterogeneous social computing platforms. They have showed that correlations in users' activity patterns across different sites provide enough evidence that the two profiles belong to the same user. Kamiyama et al in [2] have presented a unified metric for measuring anonymity and privacy with applications to online social network. They have developed a privacy metric that first measures the current anonymity for a specific question using the user's present blog posts, and estimates the user's future privacy by predicting future blog posts.

3 Privacy Metric

Our proposed Privacy Metric first identifies the attributes of a user that are present in the cyberspace. Next we check the web rank value for each of the website that contains these attributes. If the website is frequently visited, its web rank value will be higher which will make the attribute that appears at this website less private. We calculate the number of times the attribute appears in different websites. Based on the web rank values of all the websites that contain the attribute, we use a maximizing expression that finds the highest web rank value for the attribute. Next, we derive an expression for privacy ranking of the user using the maximum rank of value of each of the attribute and its count. We normalize the expression so that privacy ranking of each user will be between 0 and 1 with 0 being more private and 1 being less private. The derivation of the privacy ranking of the user is explained below.

Let us assume that user u uses Social Media SM from where we obtain the attributes $A = a_1, a_2, a_3, \ldots, \ldots, a_n$. We create a subset $S = a_1, a_2, a_3, \ldots, a_m$ where $S \in A$. For S, we find the derived set of attributes $DA(S)$. DA is the set of attributes that can be found about user u using the known attributes. Given S and u as input to the web crawler, we find $DA(S)$. The Web Crawler finds the webpages WP_1, WP_2, WP_3, …, …, WP_k. Let us assume that WP_1 reveals attributes b_1, b_2, b_3; WP_2 reveals attributes b_2, b_3, b_4; WP_3 reveals b_3, b_4, b_5; … ;…; WP_k reveals b_1, b_2, b_3, b_4, b_5, b_6, …, b_t. We obtain the Web Rank values for each of WP_1, WP_2, WP_3, …, WP_k. Let us assume that Web Rank of WP_i is $WR(WP_i)$ where $1 \leq i \leq k$. We define Matrix B where $B[i][j]$ $= 1$ if b_j appears in WP_i. $1 \leq i \leq k$ and $1 \leq j \leq t$. We define vector R where $R[i][1] = WR(WP_i)$, $1 \leq i \leq k$.

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 1 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 1 & 1 & \ldots & 0 \\ \vdots & & & & & & \\ 0 & 0 & 0 & 0 & 1 & \ldots & 1 \end{bmatrix}, R = \begin{bmatrix} WR[WP_1] \\ WR[WP_2] \\ WR[WP_3] \\ \vdots \\ WR[WP_k] \end{bmatrix}$$

We find vector $Count$ where

$$Count[j] = \sum_{i=1}^{k} B[i][j], 1 \leq j \leq t$$

$Count[j]$ calculates how many times attribute b_j appears in the web pages returned by the web crawler. We calculate Matrix P where $P[i][j] = B[i][j] * R[i]$, $1 \leq i \leq k$ and $1 \leq j \leq t$.

$$P = \begin{bmatrix} B[1][1] * R[1] & B[1][2] * R[1] & \ldots & B[1][t] * R[1] \\ B[2][1] * R[2] & B[2][2] * R[2] & \ldots & B[2][t] * R[2] \\ \vdots & & & \\ B[k][1] * R[k] & B[k][2] * R[k] & \ldots & B[k][t] * R[k] \end{bmatrix}$$

We calculate vector M where

$$M[j] = \max_{i \leq 1 \leq k} P[i][j], 1 \leq j \leq t$$

$M[j]$ finds the maximum value of web ranks for the webpages that reveals attribute $b[j]$. Finally, Privacy Ranking PR of user u, $PR(u)$ is defined using the following expression.

$$PR(u) = \sum_{j=1}^{t} (Count[j] * M[j])/(t * k * 10)$$

where t is the total number of derived attributes, k is the total number of webpages returned by the web crawler and 10 is the highest web rank value.

4 System Model

Given a user and his/her social media information, we collect the publicly available attributes of the user. We then run these attributes with the user's first and last name through a web crawler to determine how much PII information is available on each website that is found by the web crawler. Combining a formalized version of the Alexa Page Ranking [6] in conjunction with the count of found attributes, we use our privacy metric to calculate the privacy ranking. Figure 1 presents the overview of the system model.

5 Experimentation and Results

Initial attributes of a user are gathered from a selected social media. The attributes gathered are then mixed together to create unique combinations.

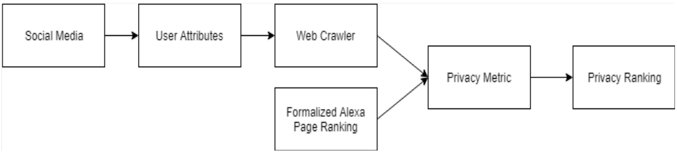


Figure 1: System model for ranking a user's privacy on the web

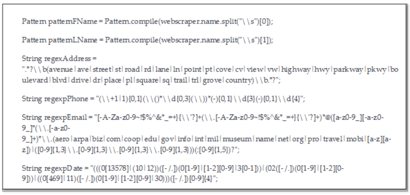
These combinations are then added to a search queue that is ran through our web crawler. Our web crawler starts off by entering the search entry from the queue into a Google search. We then collect all of the URLs from the Google search. Each URL collected will be scraped for URLs in the web page and added to a separate queue. The separate queue gives our breadth first search a distance of one to limit the number of websites not pertaining to the user searched. We then combine the two queues and crawl through all the websites again. The next time we crawl through the websites we search for PII using the formulated regular expressions (see Figure 2).

In concurrence with the search for PII in each website, we also calculate our formalized version of the Alexa Page Ranking [6]. The Alexa Page Ranking is calculated based on the number of visits to a website. This number can range from 0 to infinity. In order to formalize the Alexa Page Ranking, we grouped webpage rankings based on similarity to their Google Page Rank. This allows us to limit the website rankings from 1 to 10, where 1 is the least visited. For each website searched we create a row in our table and indicate if a PII attribute is found, and its cardinality (number of times the attribute is found in the different pages visited) based on our formulated privacy metric. After all the websites are crawled for PII, we take the cardinality of the PII attribute and highest ranking of the websites that contain this attribute, apply these values towards calculating the user's privacy ranking based on our proposed privacy metric.

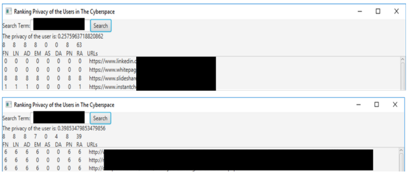
For our data collections, we were able to successfully collect data about searched users using our regular expressions and web crawler. The level of accuracy varies based on user attributes and websites crawled since our crawler relies on the Google search algorithm for accuracy. Then privacy metric was successfully implemented in combination with the web crawler to produce results for the user searched. Figure 3 shows the example of calculating the privacy values of two users where the first user has privacy value of 0.25 and second user has a privacy value of 0.39.

6 Conclusion and Future Work

Users share information about themselves in social media. This information can be used to derive more attributes about the user using web crawlers. We



```
Pattern patternFName = Pattern.compile(webscraper.name.split("\\s")[0]);

Pattern patternLName = Pattern.compile(webscraper.name.split("\\s")[1]);

String regexAddress =
".*?\\b(avenue|ave|street|st|road|rd|lane|ln|point|pt|cove|cv|view|vw|highway|hwy|parkway|pkwy|bo
ulevard|blvd|drive|dr|place|pl|square|sq|trail|trl|grove|country)\\b.*?";

String regexpPhone = "(\\+1|1){0,1}(\\()*\\d{0,3}(\\))*(-){0,1}\\d{3}(-){0,1}\\d{4}";

String regexpEmail = "[-A-Za-z0-9~!$%^&*_=+}{\\'?]+(\\.[-A-Za-z0-9~!$%^&*_=+}{\\'?]+)*@([a-z0-9_][-a-z0-
9_]*(\\.[-a-z0-
9_]+)*\\.(aero|arpa|biz|com|coop|edu|gov|info|int|mil|museum|name|net|org|pro|travel|mobi|[a-z][a-
z])|([0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}))(:[0-9]{1,5})?";

String regexpDate = "(((0[13578]|(10|12))([-/.])(0[1-9]|[1-2][0-9]|3[0-1]))|(02([-/.])(0[1-9]|[1-2][0-
9]))|((0[469]|11)([-/.])(0[1-9]|[1-2][0-9]|30)))([-/.])[0-9]{4}";
```



Ranking Privacy of the Users in The Cyberspace
Search Term:
Search

```
Search Term: [redacted]
The privacy of the user is: 0.2575963718820862
8   8   8   8   0   0   8   63
FN  LN  AD  EM  AS  DA  PN  RA  URLs
0   0   0   0   0   0   0   0   https://www.linkedin.c[redacted]
0   0   0   0   0   0   0   0   https://www.whitepag
8   8   8   8   0   0   8   8   https://www.slideshar
1   1   1   0   0   0   0   1   https://www.instantch
```

Ranking Privacy of the Users in The Cyberspace
Search Term:
Search

```
Search Term: [redacted]
The privacy of the user is: 0.39853479853479856
8   8   8   7   0   4   8   39
FN  LN  AD  EM  AS  DA  PN  RA  URLs
6   6   6   6   0   0   6   6   http://[redacted]
6   6   6   0   0   0   6   6   http://
6   6   6   6   0   0   0   6   http://
```

Figure 2: Regular expressions to discover and match user attributes

Figure 3: Privacy ranking obtained for two sample users – anonymity maintained

analyze the websites that contains these derived attributes. If other users frequently visit the website, then the derived attribute contained at that website becomes less private. Based on the number of times these attributes appear at the websites, and the web ranking of the websites themselves, we propose a novel privacy ranking method to calculate the privacy of users. The range of the privacy rank for each user is between 0 and 1, where 1 means less private and 0 means more private. Industry and government agencies that deal with sensitive information can use our model to calculate the privacy of their employees. If an employee is less private, then the probability of leaking sensitive information through that employee will be higher. As part of future work, we will categorize attributes based on their weights toward privacy.

References

[1] M. Cheung, J. She, "Evaluating the Privacy Risk of User-Shared Images" in *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, Volume 12 Issue 4s, Article 58, November 2016.

[2] K. Kamiyama, T. Ngoc, I. Echizen, H. Yoshiura, "Unified Metric for Measuring Anonymity and Privacy with Application to Online Social Network" in *6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2010, pp. 506-509.

[3] A. Narayanan, V. Shmatikov, "De-anonymizing Social Networks" in *30th IEEE Symposium on Security and Privacy*, 2009, pp. 173-187.

[4] R. Zafarani, H. Liu, "Connecting users across social media sites: a behavioral-modeling approach" in *19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 41-49.

[5] E. Bertino, (2016) "Data Security and Privacy in the IoT" in *19th International Conference on Extending Database Technology* , March 2016.

[6] https://www.alexa.com/siteinfo.

One Department, Four Undergraduate Computing Programs[*]

Tony Pittarese, Brian Bennett, Mathew Desjardins
Department of Computing
East Tennessee State University
Johnson City, TN 37614
{pittares,bennetbt,desjardins}@etsu.edu

Abstract

While most departments accredited by the ABET Computing Accreditation Commission offer only one undergraduate program, the Department of Computing at East Tennessee State University houses three such programs: Computer Science, Information Systems, and Information Technology. In fall 2019 a fourth undergraduate program in Cybersecurity and Modern Networks will be added. Combining multiple programs in a single department allows for increased student opportunity, improved faculty utilization, efficient course scheduling, streamlined accreditation and assessment management, improved student retention, and enhanced financial stability to support growth.

1 Introduction

The East Tennessee State University Computer Science Department was founded in 1975 as one of the first computing-focused departments in the Tennessee higher education system and the region overall[8]. In subsequent years the Department has added multiple undergraduate programs making it, as of this writing, the only ABET-accredited institution to house three or more ABET-accredited computing programs in a single academic department served by a single team of faculty that reports to one department chair [2]. Reflective of its unique structure, in 2010 the department changed its name to the Department of Computing. The department employs 16 tenured/tenure-track faculty, 5 full-time non-tenured faculty, 19 graduate students, and 5 full-time support staff. The Department currently enrolls 500 students, graduates 80-85 students per year, and generates over 17,500 credit hours each academic year. (Roughly 45% of those hours come from service courses offered to the university as a whole). Rather than operate as four distinct academic departments, or even as four distinct entities merged into one unit such as a college or school, the department has structured itself as a single, unified department with four integrated programs. This structure provides many positive opportunities for students, faculty, staff, and college/university administration. The efficiency such a structure affords has proven useful to ensure consistent funding to support growth in a time when overall university attendance has been stagnant or in decline.

2 Curriculum and the "Computing Core"

The Computer Science (CS) program has a traditional theoretical approach, relying heavily on advanced math and disciplines such as computer architecture and operating systems. The Information Systems (IS) program focuses on business applications of computing and requires students to complete a 15-hour application emphasis area in management or accounting. The Information Technology (IT) program serves students interested in applied computing fields such as web development and system administration. The Cybersecurity and Modern Networks (CSMN) program focuses on advanced topics in computer security and contemporary networking applications such as Internet of Things and cloud computing. The structure of the Department's curriculum has evolved over its 44-year history to facilitate its current integrated format. Initially, CS and IS programs shared an introductory programming sequence based on C++ and data structures, while IT students learned Visual Basic and took courses in web design and advanced database [4]. As all programs required at least 3 computing electives, the intent was for students to be able to take courses required in other departmental programs to meet this need. In reality, course prerequisites often made this difficult. The first major step towards unifying the department around an integrated set of core courses was made in 2007 when all undergraduate programs moved to require Java as the introductory language. While this change was heavily debated at the time [6], it reduced any potential setbacks for students who wished to switch programs their freshman year and also established a programming lingua franca among all programs to serve as a foundation for later coursework[8]. This went a long way toward resolving prerequisite issues, providing students with a robust set of elective course choices. Given the success of the above, the department worked to restructure all programs such that they built off a common set of core courses that became known as the "Computing Core." This Computing Core encompasses all the coursework needed to fulfill the ABET "a through k" outcomes required for all computing programs at that time. All programs were designed such that (1) their freshman year courses were similar and (2) during their senior year all programs completed the same set of software engineering courses. This strategy starts all students together and then brings all students back together—allowing inter-program teams to be created for projects, a structure which is often employed in engineering programs [1]. The set of courses that compose the Computing Core is listed below:

CSCI 1250 Introduction to Computer Science I (Java I)

CSCI 1260 Introduction to Computer Science II (Java II)

CSCI 1400 PC Set-Up and Maintenance

CSCI 1510 Student in the University (Freshman Orientation)

CSCI 1900 Math for Computing (Discrete Math)

CSCI 2020 Fundamentals of Database

CSCI 2150 Computer Organization

CSCI 3400 Networking Fundamentals

CSCI 3500 Information Security and Assurance

CSCI 4250 Software Engineering I

CSCI 4350 Software Engineering II

To represent the organization of the programs, a recommended program of study showing student progression through coursework is shown in Figure 1.

3 Staffing and Course Scheduling—Student Benefits

A number of benefits have been realized from creating multiple computing programs that build from the Computing Core. While freshmen are asked during the university admission process to select a program, they generally are ill equipped to judge the difference among various computing programs prior to beginning coursework. By making the freshman year fairly uniform for all computing students, they have the opportunity to experience the Core for two or three semesters before committing to a program. When students do elect to change programs, courses which were taken that are not required in their new program can be counted as in-program electives. Migration from one computing program to another is facilitated by the Core, and the Department benefits from retaining students after the program change.

Not all students have the desire or ability to excel in the Computing program they initially choose upon entry to the university. Approximately 25For students who wish to go beyond the completion of a single program of study, the Computing Core facilitates students completing a “double concentration” by using program and free electives toward completing a second program. (IS students, for example, can use their IS major electives to take courses required for IT students. They then have to complete only 1-2 additional courses to complete a second concentration in IT.) This “de-siloed” approach allows students to achieve cross disciplinary study from the other programs, diversifying their knowledge and experience in computing. The ease with which students can transition from one Department program to another and combine elements of multiple programs are key drivers in recruitment and retention.

4 Staffing and Course Scheduling—Faculty Benefits

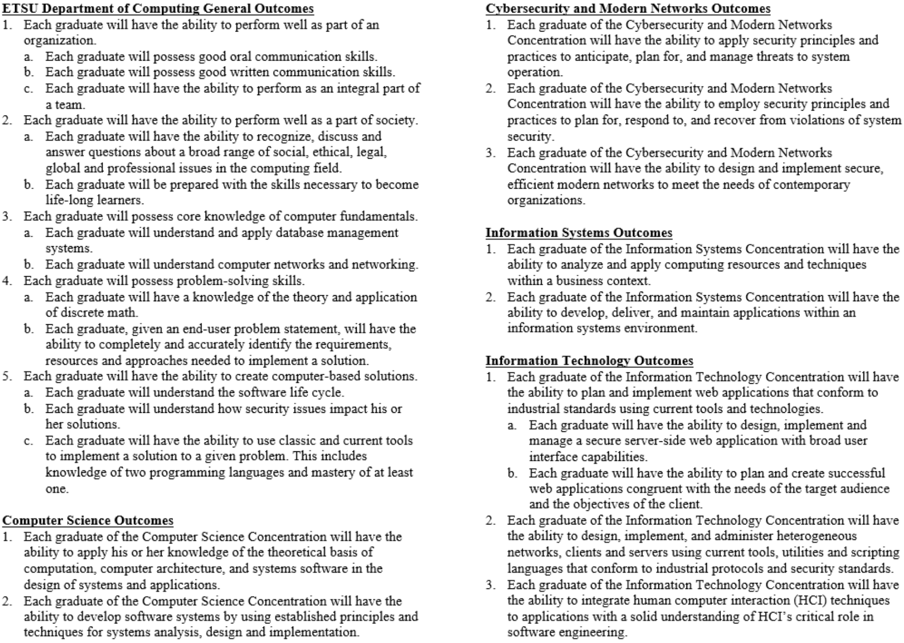
One of the most common questions raised regarding this structure has come to be realized as one of the structure’s prime benefits: How can one team of faculty teach four distinct programs, particularly given that most other institutions conduct such programs in different departments? While this is a readily acknowledged challenge, it provides an opportunity to create synergies among the faculty that are similar to the benefits afforded students. Instead of faculty members serving only one program, faculty members instead teach in their area(s) of expertise across all programs. A study of the spring 2019 schedule revealed that every full-time faculty member taught one or more courses that spanned multiple programs. Due to the combination of faculty skills present in the department, a survey of faculty in spring 2017 determined that every course offered by the department could be taught by at least 2 faculty members, with an average of 4.5 qualified instructors being available to teach all required courses. By housing all Computing courses in a single department, course enrollments are increased, making it feasible to offer a diverse set of special topics courses and other special course offerings while having confidence that these courses will attract sufficient student enrollment—since students from all four programs may elect to enroll. This structure allows a diverse team of faculty members to teach a diverse body of students in their field of expertise. It promotes esprit de corps among the students as they frequently work on projects with students from other Departmental programs, and it also fosters a similar spirit among the faculty as all programs are viewed as a shared effort.

5 Staffing and Course Scheduling—Administrative Benefits

Housing all four programs in one department allows one department chair to administer all programs, one system manager and one assistant system manager to administer computing systems for all programs (7 labs, over 225 machines, 1 unified lab image), and two full time student advisors to work with freshmen in all four programs. This has resulted in reduced administrative overhead and expenses, providing more funds for professional development of faculty and staff, travel, equipment purchases, and other Departmental priorities. While such a structure promotes efficient topic coverage within the Department, it introduces challenges. As faculty members tend to not think of themselves in terms of individual programs, it has proven helpful to recognize program champions to be advocates for each program and fulfill program-specific responsibilities in tasks such as accreditation visits. During accreditation visits, time must be taken to explain the unique Department structure and align faculty interviews and other activities so as to provide evaluators the opportunity to become comfortable with the staffing structure employed.

6 Program Outcomes and Assessment

Building off the structure provided by the Computing Core, a set of five "General Outcomes" has been defined. These General Outcomes apply to all graduates, regardless of their program. Program-specific outcomes extend the General Outcomes similar to how program-specific courses extend the Computing Core. These outcomes are shown in Figure 2. A key strength of this structuring of outcomes is how it facilitates program assessment [7]. General Outcomes are measured either (1) via course-embedded activities and rubrics in Computing Core courses—most particularly Software Engineering I and II, which serve as the Computing Core capstone courses—or (2) by a senior exit exam taken by all graduates. Regardless of which of these two methods are employed, General Outcome data is collected for all programs at one time. Although all scores, rubrics, results, etc. are anonymous, each of these are tagged with the student's program of study to facilitate data reporting and analysis. Program-specific outcomes are collected in courses unique to individual program. To ease assessment workload, program-specific data is collected only for students in the particular program. (No program-specific data is captured for students taking elective courses outside of their program.)



ETSU Department of Computing General Outcomes

1. Each graduate will have the ability to perform well as part of an organization.
   a. Each graduate will possess good oral communication skills.
   b. Each graduate will possess good written communication skills.
   c. Each graduate will have the ability to perform as an integral part of a team.
2. Each graduate will have the ability to perform well as a part of society.
   a. Each graduate will have the ability to recognize, discuss and answer questions about a broad range of social, ethical, legal, global and professional issues in the computing field.
   b. Each graduate will be prepared with the skills necessary to become life-long learners.
3. Each graduate will possess core knowledge of computer fundamentals.
   a. Each graduate will understand and apply database management systems.
   b. Each graduate will understand computer networks and networking.
4. Each graduate will possess problem-solving skills.
   a. Each graduate will have a knowledge of the theory and application of discrete math.
   b. Each graduate, given an end-user problem statement, will have the ability to completely and accurately identify the requirements, resources and approaches needed to implement a solution.
5. Each graduate will have the ability to create computer-based solutions.
   a. Each graduate will understand the software life cycle.
   b. Each graduate will understand how security issues impact his or her solutions.
   c. Each graduate will have the ability to use classic and current tools to implement a solution to a given problem. This includes knowledge of two programming languages and mastery of at least one.

Computer Science Outcomes

1. Each graduate of the Computer Science Concentration will have the ability to apply his or her knowledge of the theoretical basis of computation, computer architecture, and systems software in the design of systems and applications.
2. Each graduate of the Computer Science Concentration will have the ability to develop software systems by using established principles and techniques for systems analysis, design and implementation.

Cybersecurity and Modern Networks Outcomes

1. Each graduate of the Cybersecurity and Modern Networks Concentration will have the ability to apply security principles and practices to anticipate, plan for, and manage threats to system operation.
2. Each graduate of the Cybersecurity and Modern Networks Concentration will have the ability to employ security principles and practices to plan for, respond to, and recover from violations of system security.
3. Each graduate of the Cybersecurity and Modern Networks Concentration will have the ability to design and implement secure, efficient modern networks to meet the needs of contemporary organizations.

Information Systems Outcomes

1. Each graduate of the Information Systems Concentration will have the ability to analyze and apply computing resources and techniques within a business context.
2. Each graduate of the Information Systems Concentration will have the ability to develop, deliver, and maintain applications within an information systems environment.

Information Technology Outcomes

1. Each graduate of the Information Technology Concentration will have the ability to plan and implement web applications that conform to industrial standards using current tools and technologies.
   a. Each graduate will have the ability to design, implement and manage a secure server-side web application with broad user interface capabilities.
   b. Each graduate will have the ability to plan and create successful web applications congruent with the needs of the target audience and the objectives of the client.
2. Each graduate of the Information Technology Concentration will have the ability to design, implement, and administer heterogeneous networks, clients and servers using current tools, utilities and scripting languages that conform to industrial protocols and security standards.
3. Each graduate of the Information Technology Concentration will have the ability to integrate human computer interaction (HCI) techniques to applications with a solid understanding of HCI's critical role in software engineering.

Figure 2: General and program-specific outcomes.

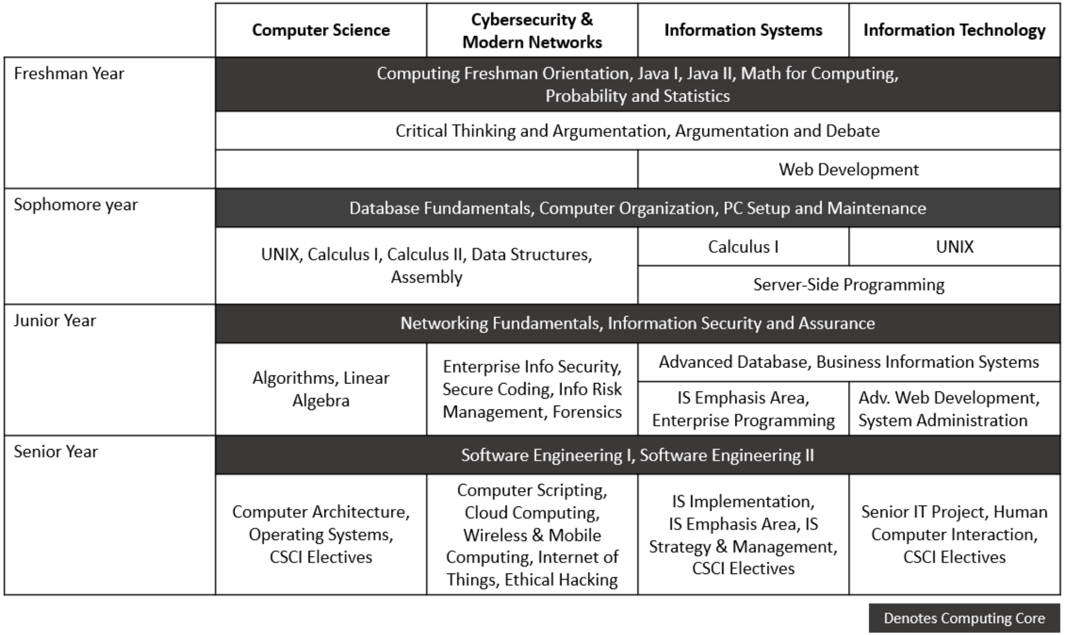
It is common for assessment-driven improvements implemented for one program to also result in improvements for the other programs. In situations where assessment data reported for one program is lower than that of the other programs, it is possible to compare the course outcomes across all programs to more precisely pinpoint ways to implement improvement. For example, in 2015 it was noted in IT assessment reporting that students were doing poorly in oral communication. As this was investigated further, it was discovered that performance of students varied greatly based on what general education speech course they selected. A change was made in all programs to require students to take Argumentation and Debate—thereby having an improvement needed in the IT program to drive improvement in all programs. In 2014, tools such as Slack were introduced to improve teamwork skills of CS students in Software Engineering work, which also resulted in improvements across all programs. These examples show how changes introduced to improve performance for one program can also benefit students in all programs. While conducting activities such as Self-Studies for accreditation purposes, the multi-program character of the department facilitates reporting. Only one ABET Self-Study document is required to maintain accreditation (although outcome reporting and other document elements differentiates the programs), and only one ABET team visits campus to evaluate the programs.

7 Four-in-One Benefits

The flexibility of the programs' structure makes the Department's offerings more attractive to students who are unsure of their major. This improves initial enrollment as students have more flexibility in their planning. The Department is more resilient when it comes to changes in the popularity of one major over another. The Department has a much higher rate of retention than the University as a whole (50% vs. upper 30%'s). This diversity of programs plays a role in the higher retention rate for the Department when compared to the University, as students can change their program of study without changing their home department, classmates, advisor, etc. Faculty research benefits from a diverse student base. Many research projects within the department involve students from more than one program. For example, one ongoing computer security research project uses CS students as developers, IT students as system administrators, and IS students as data analysts [5]. The breadth of skills that faculty have access to is much deeper than that often found in more traditional, siloed departments. The diversity among the students benefits the Software Engineering I and II sequence that serves as the capstone of the Computing Core. Through various course projects, students complete the full development life cycle of a software product. Having students with diverse programs of study work together provides diverse expertise, resulting in a more robust software solution. In addition, it allows for the introduction of other concepts like DevOps in Software Engineering [3], which relates to each of the disciplines. The department's graduate program also benefits from the common Computing Core. Students from any of the Department's undergraduate program who move on to do graduate work with the department find they have no need to take undergraduate foundational courses to succeed with their program of study.

8 Conclusion

While the combination of four computing programs into one department is atypical, there are curriculum, assessment, staffing, and scheduling benefits as outlined. Student retention is stronger than the university norm, and students gain more flexibility in tailoring their coursework to their interests and career goals. Additional studies on the benefits of this program structure are ongoing.



	Computer Science	Cybersecurity & Modern Networks	Information Systems	Information Technology
Freshman Year	Computing Freshman Orientation, Java I, Java II, Math for Computing, Probability and Statistics			
	Critical Thinking and Argumentation, Argumentation and Debate			
			Web Development	
Sophomore year	Database Fundamentals, Computer Organization, PC Setup and Maintenance			
	UNIX, Calculus I, Calculus II, Data Structures, Assembly		Calculus I	UNIX
			Server-Side Programming	
Junior Year	Networking Fundamentals, Information Security and Assurance			
	Algorithms, Linear Algebra	Enterprise Info Security, Secure Coding, Info Risk Management, Forensics	Advanced Database, Business Information Systems	
			IS Emphasis Area, Enterprise Programming	Adv. Web Development, System Administration
Senior Year	Software Engineering I, Software Engineering II			
	Computer Architecture, Operating Systems, CSCI Electives	Computer Scripting, Cloud Computing, Wireless & Mobile Computing, Internet of Things, Ethical Hacking	IS Implementation, IS Emphasis Area, IS Strategy & Management, CSCI Electives	Senior IT Project, Human Computer Interaction, CSCI Electives

Denotes Computing Core

Figure 1: Recommended progression through the four programs of study

References

[1] ABET. Criteria for accrediting engineering programs, 2018–2019. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2018-2019.

[2] ABET. ABET accredited programs, 2019. http://main.abet.org/aps/Accreditedprogramsearch.aspx.

[3] B. T. Bennett and M. L. Barrett. Incorporating DevOps into undergraduate software engineering courses: A suggested framework. *Journal of Computing Sciences in Colleges*, 34(2):180–187, 2018.

[4] T. Countermine and P. Pfeiffer. Implementing an IT concentration in a CS department: content, rationale, and initial impact. In *Proceedings of the thirty-first SIGCSE technical symposium on Computer Science education*, pages 275–279. ACM, 2000.

[5] B. Franklin J. Dangler M. Lehrfeld, A. Ogle. Dimensioning spam–an in depth examination of why users click on deceptive emails. *Journal of Computing Sciences in Colleges*, 30(2):213–219, 2014.

[6] ETSU Department of Computing. Proposal for using Java as our teaching language. unpublished internal document, 2006.

[7] D. Sanderson. Assessment in the department of computer and information sciences at East Tennessee State University: An overview. *International Journal of Engineering Education*, 25(5):920–927, 2009.

[8] D. Tarnoff. Interview with Don Bailes, founding chair of the department of computer science: History of the computing department at East Tennessee State University.

Examining Strategies to Improve Student Success in CS1[*]

Janet T. Jenkins and Mark G. Terwilliger
Computer Science and Information Systems
University of North Alabama
Florence, AL 35632
{jltruitt,mterwilliger}@una.edu

Abstract

With the pervasiveness of the necessity of computational thinking across fields, more universities are requiring courses that build students' ability to think computationally. Computer Science 1 (CS1) is one such course where computational thinking is required. This paper summarizes the work of two CS faculty who co-taught separate sections of CS1 for five semesters. Course modifications were made to augment CS1 with support inside and outside of the classroom for students to be successful. The use of in class tutors and requiring design documents were two of the primary modifications made to the course. A variety of data was collected in areas such as student planning, program design, frustration, and resources used to determine what relationships impact student success. One of the main benefits observed was an increase in the student pass rate.

1 Introduction

There have been many efforts to seek out better ways to teach Computer Science 1 (CS1), from lecture, to lab, to pair programming, to flipped classrooms. However, Watson and Li claim that a "pass rate of CS1 courses of 67.7%, and comparable results were found based on course size, and institutional grade level" [10]. Additionally, they assert "contributions of this study have been to show that CS1 pass rates vary by different countries, have not improved over time, and they are largely unaffected by the programming language taught in the course." [10]. There is no shortage of research in efforts on improving CS1, and yet pass rates have not made a turn for the better.

There is a global need for an increasing number of skilled programmers. The United States Bureau of Labor Statistics show that software developer jobs are predicted to grow at a rate of 24% as far out as 2026. This growth rate is much faster than the average for all occupations reported [6]. Additionally, the National Center for Education Statistics and Code.org cited that in 2015, there were just under 530,000 open computing jobs and only 60,000 students graduating with computer science degrees [4]. The gap between needed skilled developers and actual skilled developers could lead to somewhat of a crisis without attention to mass preparation for current students. Already, we see an uptick in the number of states in the US requiring at least some level of computer science education in K-12 [1].

This gap provides motivation for CS1 educators to provide a course with embedded support to help students master the rigorous material in CS1. Although K-12 state CS requirements are on the rise, there are still schools that do not offer CS curriculum [1], meaning some students will still lack the necessary background in computational problem solving. Shein posits that learning to think like a programmer is important for students in a variety of fields, even if they do not need the coding skills in their work. In effect, computational thinking helps students to break larger problems down into smaller problems in order to solve them [8].

2 Overview and Methodology

2.1 Background

In our CS1 course, the same programming language has been used for over 15 years, taught by a variety of professors, with a variety of teaching styles. We define pass as a student earning a grade of A, B, or C. Our institution shows a range of CS1 pass rates from lows under 45% to highs that rarely exceed 70%. In fact, before we began our study, a pass rate of 60% or higher was achieved in only 35% of the semesters. Additionally, assessment of our program showed that our upper level students were not prepared to build an appropriate design for their larger projects. Specifically, too many of the students did not develop algorithmic thought prior to writing code. Often, when advising our students who are struggling or are considering changing their major from CS, frustration is dominant in their demeanor.

We want students to be able to transition from critical thinking to computational thinking so computational thinking can become a tool for their problem solving [3]. We also want to provide support to meet student frustration to help them through the discomfort of solving a problem without clear solutions. The high attrition rate in CS1 due to recognizing students not persevering through problems, seeing our upper division students struggling to appropriately build a design prior to coding large projects, and wanting to have consistency among distinct sections, were the motivations for adding support for and modifying our CS1 course. This paper summarizes findings from two faculty who co-taught separate sections of CS1 for five semesters, supported by tutors who provided assistance to students inside and outside of the classroom.

2.2 Design Document

It is important for students to build mental models from their code, and we propose doing this in the form of representing algorithms in pseudocode and flowcharts. Ramalingam et al. [7] state "the student's mental model of programming influences self-efficacy and that both the mental model and the self-efficacy affect course performance." When our assessment process showed a weakness in design in upper level courses, we added design documents into CS1 in an ad hoc fashion. At the beginning of this study, we began formally requiring a specified design document for every major project in CS1. The design document has iteratively been improved from each semester to better meet our goals. A successful design document in our course will completely and unambiguously describe all of the following items: program requirements, program inputs, program outputs, a test plan with specific test cases, a solution algorithm, and a flowchart of the algorithm. The design document is considered complete if someone who is not familiar with the problem could read the document and implement the solution in any language without input from the individual who designed the plan.

Some students, especially those who have some programming experience, struggle to delay coding to develop a design. However, we want students to use the design document as a planning tool to prepare for larger projects. This activity helps the student think about the solution before they begin coding. The model also encourages students to build test cases before they develop their solution, helping them to gain a better understanding of what their program should achieve [5]. It is our goal that through analyzing the problem carefully and considering important aspects of the problem prior to coding a solution, the student will completely understand the problem before attempting a code solution.

2.3 Tutoring

We began our first semester of co-teaching with upper level CS students serving as tutors outside of class. Every semester since, at least one tutor has served to assist the instructor during the class time of each CS1 section. With a classroom of 30 students, each with a computer, this provided opportunities to expand classroom activities. Having another person to assist students makes it more manageable to add more essential, hands-on, in-class exercises where immediate feedback is available. It was our goal that attending class would allow the tutor to deepen his or her understanding of the material, the tutor knew exactly what the students had been taught during class, and students would be more comfortable visiting tutoring hours outside of class from the familiarity of the tutors. Other studies have shown that in-class tutors and peer-mentor tutors have a positive impact on both students and the tutors [2][9].

2.4 Data Collection

In order to assess the impact of our changes, as well as gather input on student perceptions, we collected data from three sources. First, on the week-long projects, we looked at student scores on the design documents, scores on the project source code, as well as questions we asked associated with each project. These questions asked students to rate the frustration level associated with the project and what resources they used when they encountered difficulties. Second, we looked at student scores on the three semester hourly exams, the final exam grades, and the final semester course grades. Finally, we administered an end-of-semester survey that asked students to provide background information (academic major, mathematics experience, programming experience), their perceptions of the helpfulness of design document components (algorithm, flowchart, test plan), where they turned to when stuck on a project, the helpfulness of various resources (textbook, professor, tutor, Internet, class notes), and tutoring usage.

3 Results and Observations

With all of the data collected from ten course sections of CS1, we looked for data trends, relationships between variables, as well as student responses from open-ended questions. In this section, we will provide a few results we found both interesting and practical.

3.1 Design Document

We feel strongly that students need to spend some time thinking and planning a problem solution before they start hacking away at the keyboard. Figure 1 compares the mean project source code grade (out of 70 points) compared to the mean design document grade (out of 15 points) on six projects. It is easy to see the strong relationship between design success and coding success.

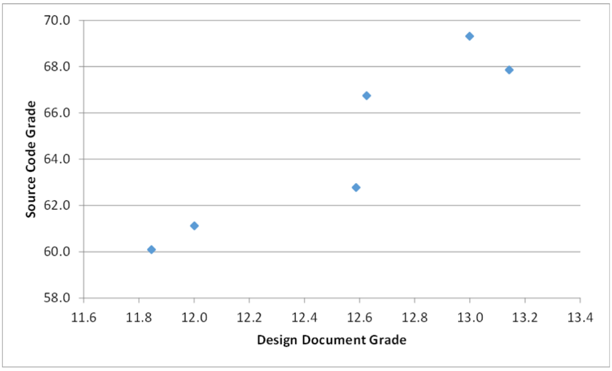


Figure 1: Project Source Code vs Design Document Grades

Another piece of data we collected was student estimates of time spent on the design document as well as the source code. In Figure 2, we show an example of student scores on the first hourly exam compared to the amount of time (in minutes) that students spent on the design document for their first project. Students that invested more time on design performed better on the exam.

3.2 Tutoring

After incorporating tutors into the CS1 classroom experience, a noticeable difference in the classroom atmosphere was obvious. In addition to improvements in the classroom environment, there were several unexpected results. As shown in Figure 3, the usage for our out-of-class, drop-in tutoring increased dramatically. We believe students became familiar and comfortable with the tutors in the classroom and felt less threatened to visit the drop-in tutoring on their own. We also noticed an observable change in our tutors, as they would frequently stop by our offices to discuss student issues, offer alternative strategies to solving problems, and suggest activities to try in upcoming classes.

Paying student tutors to attend course lectures costs money, but we believe

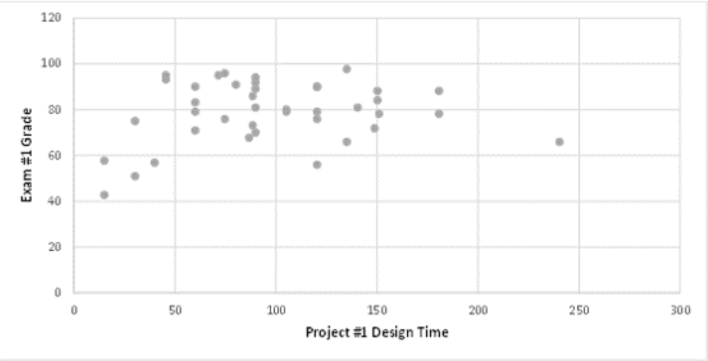


Figure 2: Exam Grade vs Time Spent on Project Design

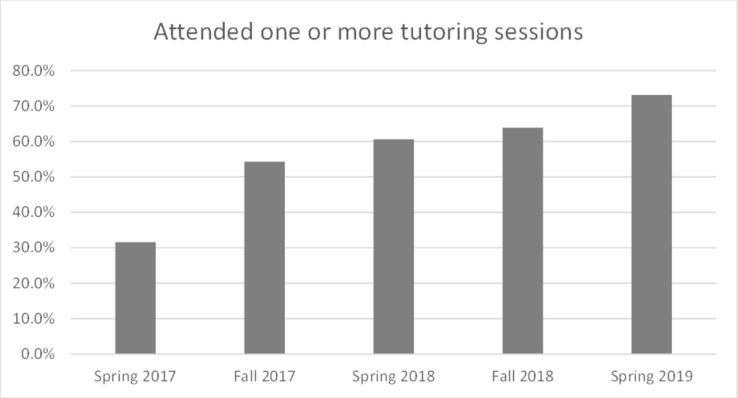


Figure 3: Student Attendance for Open Tutoring

the benefits to the classroom atmosphere, the improved students' mastery of concepts, and unique feedback to instructors are worth the investment. The unexpected benefits to the tutors acting as near-peer mentors are also positive side effects.

3.3 Project Frustration Levels

While seeking to provide support to help students, we wanted to hone in on frustration levels students experienced while completing the source code for their project. Students rated the frustration associated with each project from

1 to 10, higher values meaning more frustration. In Figure 4, we looked at the scores on the first hourly exam compared to the students average frustration level on the first three projects. In this example, you could actually predict a student's exam score with some degree of accuracy just by looking at their coding frustration level

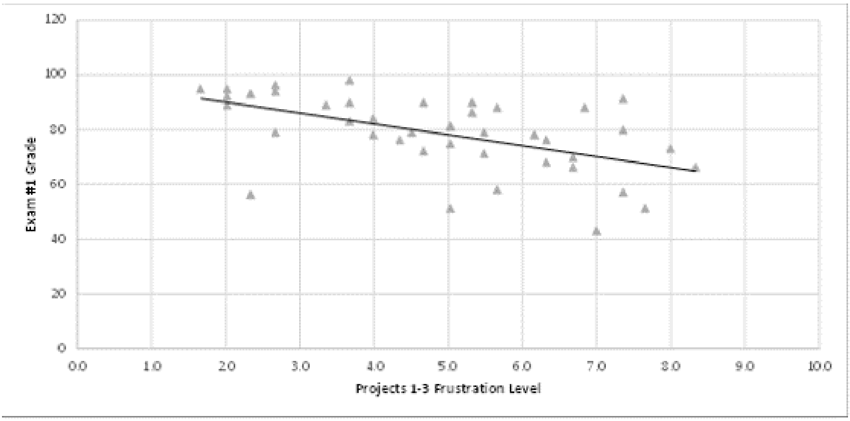


Figure 4: Exam Grade vs Frustration Level on Projects

We were also interested in where they went for help first when they were frustrated or hit a road block when they were coding. In each of the five semesters, the resource students used when encountering difficulties was to search the Internet. This was followed by looking at class notes, visiting the tutor, going to the professor's office hours, referring to the textbook, and asking a classmate for help. As shown in Figure 5, the student practice of searching the Internet for the answer first is trending downwards. Over recent semesters, we have seen a spike in frequency of visits to tutor sessions and professor office hours.

3.4 Course Success Rates

One of the most important data points we are following is the success rates of our students, not only in CS1 but also in the subsequent course CS2. In the 12 years before our study, a 60% pass rate in CS1 was achieved during 35% of the semesters. In the five semesters of this study, a 60% pass rate was achieved in all five semesters.

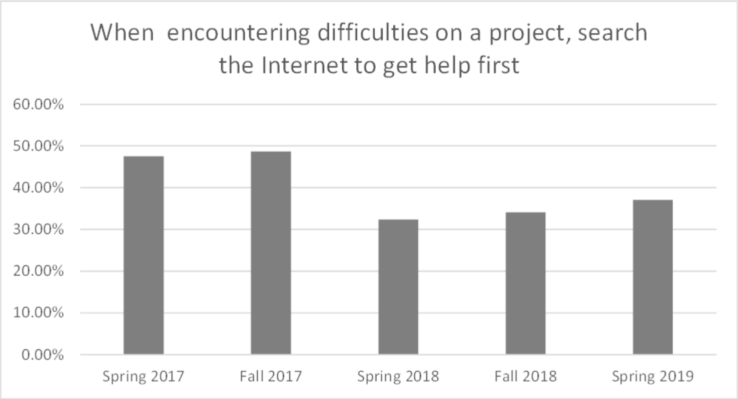


Figure 5: Students Seeking Internet Help When Stuck on Projects

4 Conclusions and Future Work

Between the three data sources mentioned, we have collected an enormous amount of both quantitative and qualitative data over five semesters and ten recent sections of CS1. We feel the changes we have made over the past three years have contributed to the increased success rates in CS1. We are also very interested in digging more into what happens when students get stuck and become frustrated while working on a project. This inevitably happens with all students at some point and we want to provide students with the necessary tools to persevere when they do encounter obstacles. More work is needed to continue exploring our data and collecting new data to help determine where students are still struggling and which of our course changes have made the most impact. This will help to inform how future CS1 courses may be developed or augmented, and what teaching methods, learning strategies, and course resources will make the most significant impact for the success of our students.

References

[1] Code.org. 2018 state of computer science education, policy and implementation. `https://code.org/files/2018_state_of_cs.pdf`.

[2] P.E. Dickson. Using undergraduate teaching assistants in a small college environment. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 75–80, New York, NY, USA, 2011. ACM.

[3] H. Fleenor. Establishing computational thinking as just another tool in the problem solving tool box. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 1253–1253, New York, NY, USA, 2019. ACM.

[4] S. Kessler. You probably should have majored in computer science. `https://qz.com/929275/you-probably-should-have-majored-in-computer-science`.

[5] W. Marrero and A. Settle. Testing first: Emphasizing testing in early programming courses. In *10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '05, pages 4–8, New York, NY, USA, 2005. ACM.

[6] US Bureau of Labor Statistics. Us bureau of labor statistics. computer and information research scientist. `https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm`.

[7] V. Ramalingam, D. LaBelle, and S. Wiedenbeck. Self-efficacy and mental models in learning to program. In *9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '04, pages 171–175, New York, NY, USA, 2004. ACM.

[8] E. Shein. Should everybody learn to code? *Communications of the ACM*, 57(2):16–18, 2014.

[9] G. Trujillo, P.G. Aguinaldo, C. Anderson, J. Bustamante, D.R. Gelsinger, M.J. Pastor, J. Wright, L. Márquez-Magaña, and B. Riggs. Near-peer stem mentoring offers unexpected benefits for mentors from traditionally underrepresented backgrounds. *Perspectives on Undergraduate Research and Mentoring*, 4(1):1–13, 2015.

[10] Christopher Watson and Frederick Li. Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education*, ITiCSE '14, pages 33–34, New York, NY, USA, 2014. ACM.

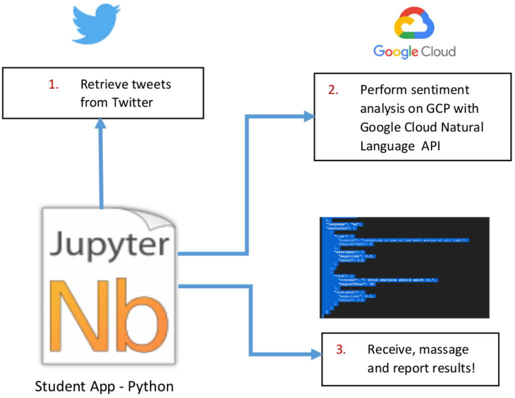
+, − or Neutral: Sentiment Analysis of Tweets on Twitter*

Nifty Assignment

Robert Lutz and Evelyn Brannock
Georgia Gwinnett College
1000 University Center Lane
Lawrenceville, Ga 30043
{rlutz,ebrannoc}@ggc.edu

1 Introduction

Sentiment Analysis is a popular application of Natural Language Processing (NLP). This exercise offers the capability to perform opinion mining in the political arena by feeding data into a cloud natural language processor, without in-depth proficiency in machine learning (ML) algorithms. It is an engaging mechanism for interesting students in using ML to extract information from voluminous amounts of text found in Twitter to understand the structure and meaning of text.


Google Cloud
1. Retrieve tweets from Twitter
2. Perform sentiment analysis on GCP with Google Cloud Natural Language API
Jupyter
Nb
Student App - Python
3. Receive, massage and report results!

```
  },
  "language": "en",
  "sentences": [
    {
      "text": {
        "content": "Inception is one of the best movies of all time.",
        "beginOffset": 0
      },
      "sentiment": {
        "magnitude": 0.9,
        "score": 0.9
      }
    },
    {
      "text": {
        "content": "I think everyone should watch it.",
        "beginOffset": 49
      },
      "sentiment": {
        "magnitude": 0.1,
        "score": 0.1
      }
    }
  ]
}
```

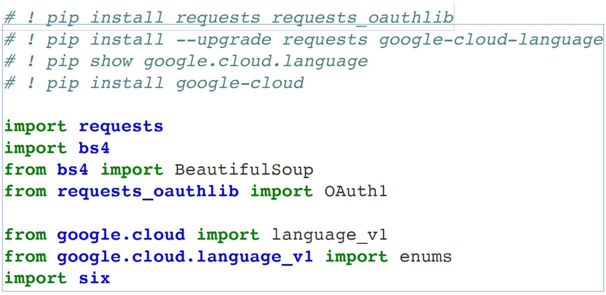
2 Materials

- Educational codes for access to Google Cloud Platform (GCP)
- Credentials to access API
- Jupyter Notebook

3 Summary

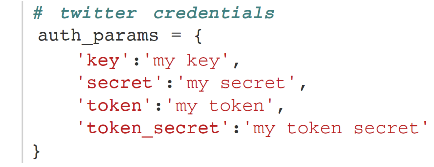
Students are asked to provide an app that provides results of a sentiment analysis of tweets on some current “hot” political subject, such as the Mueller report or tweets from President Trump as shown below.

Step 1: Load required libraries by running the install commands.

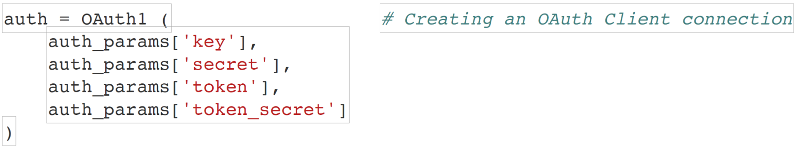


```
# ! pip install requests requests_oauthlib
# ! pip install --upgrade requests google-cloud-language
# ! pip show google.cloud.language
# ! pip install google-cloud

import requests
import bs4
from bs4 import BeautifulSoup
from requests_oauthlib import OAuth1

from google.cloud import language_v1
from google.cloud.language_v1 import enums
import six
```

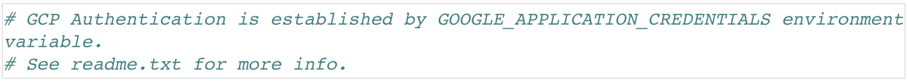
Step 2: Provide credentials to access APIs.



```
# twitter credentials
auth_params = {
    'key':'my key',
    'secret':'my secret',
    'token':'my token',
    'token_secret':'my token secret'
}
```

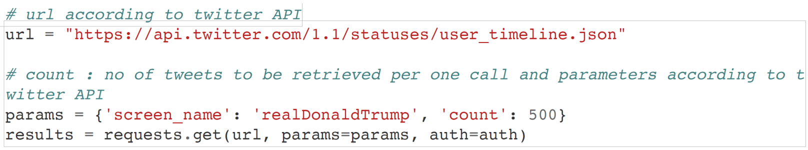


```
auth = OAuth1 (                                # Creating an OAuth Client connection
    auth_params['key'],
    auth_params['secret'],
    auth_params['token'],
    auth_params['token_secret']
)
```



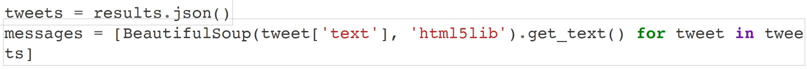
```
# GCP Authentication is established by GOOGLE_APPLICATION_CREDENTIALS environment
variable.
# See readme.txt for more info.
```

Step 3: Establish calling endpoint, call parameters and make the request.



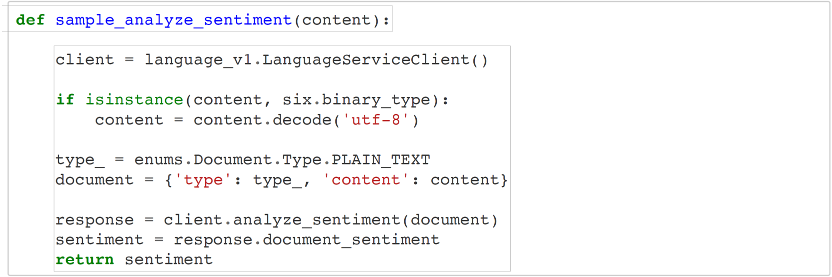
```
# url according to twitter API
url = "https://api.twitter.com/1.1/statuses/user_timeline.json"

# count : no of tweets to be retrieved per one call and parameters according to t
witter API
params = {'screen_name': 'realDonaldTrump', 'count': 500}
results = requests.get(url, params=params, auth=auth)
```

Step 4: Coerce the response into a list of messages.

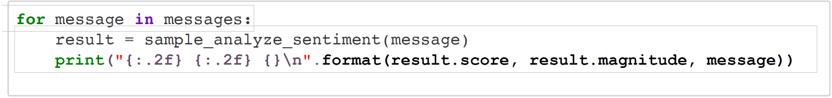


```
tweets = results.json()
messages = [BeautifulSoup(tweet['text'], 'html5lib').get_text() for tweet in twee
ts]
```

Step 5: Create a (reusable) function for sentiment analysis using Google's Natural Language Processing.

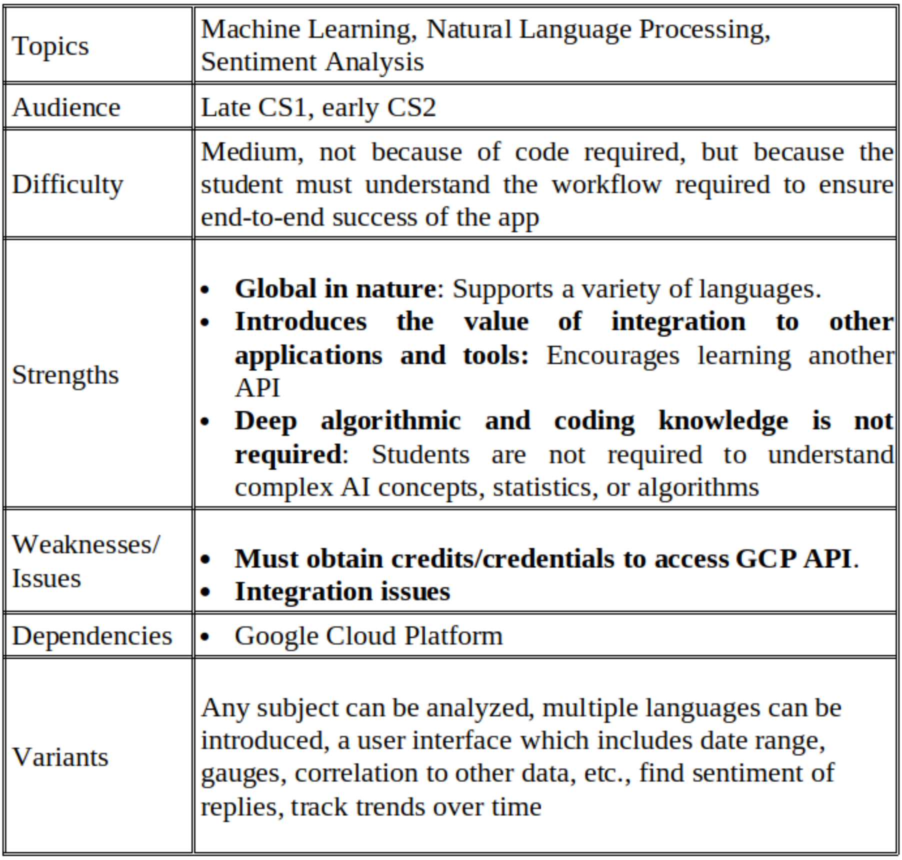


```
def sample_analyze_sentiment(content):

    client = language_v1.LanguageServiceClient()

    if isinstance(content, six.binary_type):
        content = content.decode('utf-8')

    type_ = enums.Document.Type.PLAIN_TEXT
    document = {'type': type_, 'content': content}

    response = client.analyze_sentiment(document)
    sentiment = response.document_sentiment
    return sentiment
```



```
for message in messages:
    result = sample_analyze_sentiment(message)
    print("{:.2f} {:.2f} {}\n".format(result.score, result.magnitude, message))
```

4 Metadata



Topics	Machine Learning, Natural Language Processing, Sentiment Analysis
Audience	Late CS1, early CS2
Difficulty	Medium, not because of code required, but because the student must understand the workflow required to ensure end-to-end success of the app
Strengths	• **Global in nature**: Supports a variety of languages. • **Introduces the value of integration to other applications and tools:** Encourages learning another API • **Deep algorithmic and coding knowledge is not required**: Students are not required to understand complex AI concepts, statistics, or algorithms
Weaknesses/ Issues	• **Must obtain credits/credentials to access GCP API**. • **Integration issues**
Dependencies	• Google Cloud Platform
Variants	Any subject can be analyzed, multiple languages can be introduced, a user interface which includes date range, gauges, correlation to other data, etc., find sentiment of replies, track trends over time