Proceedings of the
34th Annual Conference of
The Pennsylvania Association of Computer and Information Science Educators

April 12 - April 13, 2019

Applications of Machine Learning

Hosted by
Millersville University of Pennsylvania

TABLE OF CONTENTS

Faculty Abstracts

Graduate Student Abstracts

Undergraduate Student Abstracts

Birds of a Feather

Special Sessions

Student Posters

2019 PACISE Conference Schedule

Friday, April 12th, 2019

3:00-5:45PM	**Registration** SMC Lobby outside of MPR
3:00-4:00PM	**Poster Setup** SMC Lobby outside of MPR
4:00-5:45PM	**Poster Judging** SMC Lobby outside of MPR
4:00-5:00PM	**Workshop (Robotics)** SMC MPR (114)
4:15-5:45PM	**Student Presentations** SMC 18, SMC 24, SMC 118
4:00-5:45PM	**PACISE Board Meeting** SMC 202
6:00-7:30PM	**Dinner Reception and Keynote** Lehr Dining Room, Bolger Conference Center
7:45-???	**Programming Competition Practice** Roddy 130, Roddy 131
7:45PM	**Security Competition Kickoff** SMC MPR (114)

Saturday, April 13th, 2019

7:45-8:00AM	**Programming Competition Check-In** Roddy 130, Roddy 131
8:00-9:00AM	**Registration** SMC Lobby outside of MPR
8:00-11:45AM	**Programming Competition** Roddy 130, Roddy 131
9:00-11:45AM	**Poster Session** SMC Lobby outside of MPR
9:00-10:00AM	**BOF Session** SMC 118
9:00-10:00AM	**Presentations** SMC 18, SMC 24
10:00-10:15AM	**Break**
10:15-11:45AM	**Presentations** SMC 18, SMC 24
10:15-11:45AM	**BOF Session** SMC 118
11:45AM	**End of Security Competition**
12:00PM	**Lunch, Awards, and General Meeting** SMC 114 (MPR)

PACISE 2019 Millersville University *Friday, April 12, 2019*

	SMC Lobby		*SMC 18*	*SMC 24*	*SMC 118*	*SMC 202*	*SMC MPR*
3:00 PM	Registration	Poster Setup					
3:15 PM							
3:30 PM							
3:45 PM							
4:00 PM		Poster Judging	Student Presentations Start at 4:10 PM			PACISE Board Meeting	Workshop (Robotics)
4:15 PM			[UG] Day, Kim (Clarion) "A Comparison of Automatic Extractive Text Summarization Techniques"	[UG] Varone, Lee (Shippensburg) "Implementation of Cognitive Radio Techniques using Traditional Relay Network Principles"	[UG] Leinhauser, Burns (West Chester) "Physical Exercise Instructions: Unlocking the Key to Injury-Free Workouts using Natural Language Processing"		
4:30 PM			[G] Rhoads, Annadatha (Clarion) "A Comparison of Prediction Methods for Customer Churn Using SAS Enterprise Miner"	[UG*] Del Toro (East Stroudsburg) "Images Concentrated with Encrypted Data (ICED)"	[UG] Smale, Rush (Shippensburg) "Are Unit Tests Good Enough for Design Patterns? An Empirical Study"		
4:45 PM							
5:00 PM			[G*] Shick, Annadatha (Clarion) "Predictive Model for Automotive Insurance Losses"	[UG*] Markish (Lock Haven) "A Venture into Android Virtual Reality Development"	[G] Syed (Slippery Rock) "Analysis of Autism Spectrum Disorder Diagnosis in Children and Adults "		
5:15 PM							
5:30 PM			[UG] Groff (Millersville) "Data Preprocessing And Feature Selection for an Intrusion Detection System Dataset"		[G*] Kerle, Annadatha (Clarion) "Social Media Analysis Using Tableau, Facebook Analytics, and Facebook Insights"		
5:45 PM							
6:00 PM to 7:30 PM	Dinner and Keynote -- Lehr Dining Room						
7:30 PM							
7:45 PM	Programming Competition Practice Roddy 130/131				Security Competition Kickoff SMC MPR (114)		
8:00 PM							

PACISE 2019 Millersville University *Saturday, April 13, 2019*

	SMC Lobby	*SMC 18*	*SMC 24*	*SMC 118*	*Roddy 130/131*	*SMC MPR (114)*
7:45 AM					Programming Competition Checkin	
8:00 AM	Registration				Programming Competition	Security Competition
8:15 AM						
8:30 AM						
8:45 AM			Student Presentations in First Half			
9:00 AM	Poster Session	Pillai, Rublein (Lock Haven) "The Use of Heuristics for Robotic Guidance"	[UG] Rummel, Girard (Shippensburg) "Document Classification Problem: Does a Web Page Contain an Article?"	Birds of a Feather: BSE Computer Science Hardy, Liffick (Millersville)		
9:15 AM						
9:30 AM		Packard, Phillips (Clarion) "An Examination of the Ethics of Imitation Games"	[G] Hutchinson, Schauer, Seetan (Slippery Rock) "A Comparative Study of Data Mining Techniques Used to Test Predictive Accuracy of Autism Spectrum Disorder Screening Process"			
9:45 AM						
10:00 AM		Break				
10:15 AM		Packard, Phillips (Clarion) "An Examination of the Ethics of Asset Flips"	Ngo (West Chester) "Computing Infrastructures to Support Cybersecurity Education"	Birds of a Feather: Continually Updating Computer Science General Education for Informed Citizens Beyond 2020 Jones (Lock Haven)		
10:30 AM						
10:45 AM		Carelli (Kutztown) "A Musical Chord Player Application Build with Python/Kivy"	Menon (California) "Selecting an Appropriate Information System Development Approach for Undergraduate Capstone Courses"			
11:00 AM						
11:15 AM		Wynters (Bloomsburg) "Many Serial Programs Can be Easily Parallelized to Run Hundreds or Thousands of Times Faster"	[*] Kovalchick (California) "An Interactive Method for Teaching and Learning Cryptography"			
11:30 AM						
11:45 AM						
12:00 PM	Lunch, Awards, General Meeting SMC MPR (114)					
12:15 PM						

ACKNOWLEDGEMENTS

On behalf of the PACISE Board of Directors and the PACISE Editorial Board, I would like to thank the following people who have contributed to PACISE 2019.

Program Committee at Millersville University:

Conference Chair:	Stephanie Schwartz
Registration Chair:	Chad Hogg
Contests Chair:	William Killian
Editorial Board Chair:	Dave Mooney
Technical Liason :	Todd Echterling
Department Secretary	Tonya Pyles

PACISE Editorial Board:

Lisa Kovalchick	California University
Robert Montante	Bloomsburg University
Dave Mooney (Chair)	Shippensburg University

Reviewers:

Alawya Alawami	Clarion University
Jay Annadatha	Clarion University
Thomas Briggs	Shippensburg University
Weifeng Chen	California University
Dudley Girard	Shippensburg University
Chen Huo	Shippensburg University
Olaniyi Samuel Iylola	California University
Mark Jones	Lock Haven University
Soo Kim	Clarion University
Sangkook Lee	Shippensburg University
Pratibha Menon	California University
Brandon Packard	Clarion University
Dale Parson	Kutztown University
Carol Wellington	Shippensburg University

Sponsors:

For their financial support,

Office of the Provost Millersville University
College of Science and Technology Millerville University

For Awards and prizes for the programming contest, capture-the-flag contest, best paper, best student paper, and best poster,

Northrup Grumman

General Thanks:

Thanks to

Everyone who submitted papers, abstracts, birds-of-a-feather proposals, special session proposals, and posters.

The students who participated in the programming contest, the capture-the-flag contest, and their coaches.

The contest judges.

The Millersville students who assisted with the conference.

Special Acknowledgement

I would like to extend a special acknowledgment to David Huthchens of Millersville University, who is retiring this year. David has served as Millersville's representative on the PACISE Board for over twenty years. He has served as president and as chair of the Editorial Board.

The Board would like to extend its gratitude for his years of service and wish him well in his new life.

Dave Mooney,
Chair, PACISE 2019 Editorial Review Board

KEYNOTE ADDRESS

APPLIED MACHINE LEARNING

Dave Feltenberger
Staff Software Engineer
Google

ABSTRACT

Drawing on Dave's experience leading a machine learning organization, this talk will cover several applied case studies of machine learning on corporate ("Enterprise") data. After a brief introduction to machine learning concepts, a few hypothetical Enterprise case studies will be discussed. In particular, some work with patent classification; estimating demand to Cafes to help reduce food waste; finding organically forming groups for answering questions such as "who works with whom?" and "who works on what?"; automatically detecting issues with video conference equipment to reduce the time to a fix; and routing helpdesk tickets to the proper channels without laborious and repetitive human intervention. There will be ample time for questions after the talk.

BIOGRAPHY

Dave is a graduate of the Computer Science department at Millersville University (class of 2003). He also holds an MS in Computer Science from the University of Maryland, Baltimore County, with a focus on machine learning. Following and concurrent with his studies, Dave worked in a number of industries -- building some of the early telematics systems for large fleets of vehicles; an electronic medical records startup; distributed systems work on Goldman Sachs' equity and derivatives trading platform; and finally, leading multiple machine learning teams at Google. Currently, Dave leads a machine learning team in the location platform of Google Maps.

BEST PAPER AWARDS

Faculty:	Linh B Ngo, West Chester University
	COMPUTING INFRASTRUCTURES TO SUPPORT CYBERSECURITY EDUCATION
Graduate student:	Joshua Rhoads and Jay Annadatha, Clarion University
	A COMPARISON OF PREDICTION METHODS FOR CUSTOMER CHURN USING SAS ENTERPRISE MINER
Undergraduate student:	Nicholas Rummel and C. Dudley Girard, Shippensburg University
	DOCUMENT CLASSIFICATION PROBLEM: DOES A WEB PAGE CONTAIN AN ARTICLE?

REFEREED FACULTY PAPERS

A MUSICAL CHORD PLAYER APPLICATION BUILT WITH PYTHON/KIVY

John Carelli
Kutztown University of Pennsylvania
carelli@kutztown.edu

ABSTRACT
A preliminary design for a musical chord player had been prototyped using the Python[1]/Kivy[2] application development platform. The concept is to provide a touch-sensitive app[3] that can be used as an aid to songwriters in quickly developing musical ideas, by educators and students to develop an understanding of chord progressions and relationships and even, possibly, as an easy-to-play musical instrument in a performance setting. The app presents a set of most commonly used chords in western popular and classical music in a readily playable format that is reconfigured automatically based on a user selectable musical key. The selection of included chords is based on basic musical principles involving related keys as well as an analysis of an extensive database of representative songs, both of which are described. The use of Python/Kivy as a program development environment will allow the still-developing application to be rapidly modified and improved.

KEY WORDS
Musical instrument, chord, Python, Kivy, MIDI, song

1. Introduction

The impetus for this work grew out of a desire to create a simple musical instrument that could be easily played by anyone, particularly those who are not necessarily proficient with a standard musical instrument. The goal envisioned was to remove an individual's lack of virtuosity as an obstacle to the objective of creating, understanding, and/or performing music.

The creation of a touch-screen based computer application seems a natural approach to this. If an app could be developed that allows a user musical versatility without being difficult to navigate, it could find application in song writing, in musical education, and even, perhaps, in musical accompaniment.

The concept of a touch screen application for playing musical chords is, by itself, not a new idea. Neither is the idea of an easy to play instrument. Simple applications exist, and can readily found in app stores, for both computers and portable devices like tablets and phones that do something similar. Some of these mimic musical instruments like pianos or guitars and allow a user to play individual notes as well as chords. Others are chord based, generally offering a set of basic chords. In addition, some physical instruments, like certain electronic keyboards, or specially designed instruments like the Qchord [4], allow one button chord generation. Many of these instruments and apps are simple novelties or are targeted to users with limited musical knowledge or aspiration [5]. Other research has targeted the creation of web-based environments for instrument creation [6, 7].

This effort looks to extend on those by creating a musical environment, i.e. an instrument, which offers an opportunity for more serious musical expression and/or development while, at the same time, remaining relatively easy to use. In addition, taking a software based approach will allow for the expansion of capabilities as experience using the instrument reveals new needs and applications. This document presents a preliminary attempt at the implementation of such a device based on research into musical chord usage in songs.

2. Development Approach

The application presented here will be limited to playing musical chords - not individual notes. That is to say, the instrument, at least in this initial development, will not play melody lines. Narrowing the focus in this way is a first, but important, step in keeping the user interface simple and approachable while still retaining its usefulness for the targeted applications mentioned previously. It should also be noted that it is targeted to music that many would identify as in a western tradition: 12 tone scales, major and minor keys, and so forth.

Since this is a chord based application, the basic challenge is to provide a rich enough selection of chords and chord types to make the instrument usable in a musical context while, at the same time, keeping the interface as approachable as possible. Too narrow a selection of chords limits the utility. Too broad a selection can result in an overly cluttered and confusing user screen that would become difficult to navigate.

Two separate approaches were taken to guide the chord selection. The first was to consider which chords one

would expect to be most used based on basic musical principles [8]. Assuming a song is in a given key, this just involves making a list of chords found in that key as well as in related keys, or tonal centers. A key change would simply involve having the software make an adjustment to a new, but similar, list of chords in the new key. Of course, more complex songs may not be in what would be thought of as a well-defined key, or make use of more esoteric chords, but this approach should cover the vast majority of songs that most users would be interested in.

The second approach was to determine which chords are actually most used based on an analysis of a large number of representative songs. Toward this end, an extensive collection of over 6500 songs, available in Music-XML [9] format, was data-mined using scripts written in the Perl[10] programming language specifically for this purpose. The result produced a listing of chords and the frequency with which they were used.

With the information obtained from these two approaches, a basic chord playing app was written. This will be described later in this document, but it is viewed by the author as an initial attempt in an ongoing research effort.

The actual app itself was written in the Python programming language using the Kivy extension. Python is well suited to this task as its simple but powerful object-oriented syntax and capabilities allow for rapid prototyping. In addition, it has an extensive list of supporting libraries. Among these, a library for generating MIDI information, was especially useful for playing the chords.

The Kivy extension to Python facilitates the development of multi-touch enabled GUI-based applications. In addition, Kivy is designed to be portable to multiple operating systems. For these reasons, it was chosen for this project.

3. Chord Selection

Two different approaches were taken to identify a set of musical chords for inclusion in the player. The first involved starting with the set of chords found in a given musical key and adding chords in its closely related keys. These would be expected to be the chords most used by songs in that key. The second involved actually performing an examination of a collection of a large number of songs and identifying which chords are most used in practice. These two approaches will now be discussed.

3.1 Chords in related keys

The vast majority of songs in western tradition are in either a major or minor key. While it is not uncommon to modulate from one key to another, generally speaking, a song is usually in an identifiable key at any given point in time. Of course there are exceptions to this. Some more complex songs, certain jazz songs, for example, may not have a clearly identifiable “tonal center”, but that is less common. This analysis is less concerned with those cases.

Within a given key, one can list a set of chords built on the scale notes found in that key. For example, if one considers just basic triads and seventh chords in the key of C, the list of chords constructed in this manner would be [11]:

Triads
C, Dm, Em, F, G, Am, Bdim

Seventh Chords
Cmaj7, Dm7, Em7, Fmaj7, G7, Am7, Bm7b5

This could be extended to include 9^{th}, 11^{th} and 13^{th} chords [12], but those are less commonly used and, generally, exist more as embellishments. Therefore, for simplicity’s sake, they will not be considered here.

This collection of chords, while a good starting point, would almost certainly be insufficient to play anything but the most basic songs. To extend the list, one looks to include chords from related keys[13].

Related keys are those that have a strong overlap in terms of shared scale notes. Consider a given major key. One of its related keys would be its relative minor key, since it actually has the same scale notes as the relative major key and, thus, the same chord set. By itself, that doesn’t add anything new. However, two other closely related keys would be the sub-dominant and dominant keys – those major keys built on the 4th and 5th tones in the original major scale. In the key of C major, those related keys would be F and G major, respectively. The scale notes in each of those keys share 7 of the 8 scale tones with the key of C. These related keys might be considered to be common keys from which to “borrow” chords when writing in the key of C. Finally, the related (as opposed to relative) minor key is closely related in the sense that it is natural to transition between it and the major key. In this example, that would be C-minor.

If one adds the chords constructed from each of these related keys to the original list, the new list of unique chords grows in number from 14 (the list above) to 39. This, however, might still be regarded to be incomplete in the sense that it ignores the fact that there are three flavors of commonly used minor scales. The A-minor key referred to above is built on a scale that is known as natural minor. The other two variants, relative minor and harmonic minor, make adjustments to notes in that scale. Details are beyond the scope of this discussion but, suffice it to say, those adjusted scales result in an additional set of chords being added to the list. The final chord count grows to 58. Table I summarizes the chord

lists resulting from these approaches for the key of C-major. Note that the columns are additive. The second column contains the original 14 mentioned. Adding the column from related keys expands the total to 39 and all 3 columns contain 58 chords. A similar analysis can be done for the minor keys.

Table 1. Chord List

Scale Note (C-Major)	*Major Key Chords*	*Additional Chords from Related Keys*	*Additional Chords from Relative and Harmonic Minor Keys*
C	C, Cmaj7	C7, Cm, Cm7	Caug, Cmaj7#5, CmMaj7
D	Dm, Dm7	D, Ddim, D7, Dm7b5	
Eb		Eb, Ebmaj7	Ebmaj7#5, Ebaug
E	Em, Em7	Edim, Em7b5	E, E7
F	F, Fmaj7	F, Fm7	F7, Fm7b5, Fdim
Gb		Gbdim, Gbm7b5	
G	G, G7	Gm, Gmaj7, Gm7	Gdim, Gm7b5, Gdim7
Ab		Ab, Abmaj7	Abdim, Abm7b5
A	Am, Am7		AmMaj7
Bb		Bb, Bbmaj7, Bb7	Bbdim, Bbdim7, Bbm7b5
B	Bdim, Bm7b5	Bm, Bm7	

Note: There are no entries for Db/C# (minor second)

3.2 Chords derived from song analysis

The second approach to deriving a set of chords to include in the app involved data-mining a large number of songs representative of the targeted musical tradition. For this purpose, a collection of songs in Music-XML format was used. This collection contained approximately 6500 songs and ranged across a variety of musical styles, including classical, American Songbook, and pop, to name a few. The original source of the files was a, now defunct, website called Wikifonia[14,15], which originally published (with copyright clearance) music contributed by a user community. The information for each song consisted, essentially, of the song's lead sheet, that is to say, the melody and the chords for the song, which is exactly what is needed for this analysis.

The approach used was to parse the Music-XML file and extract pertinent information. Specifically, the song's key, the notes used in the melody and, of course, the chords were extracted from the song file. This was done with Perl scripts, written specifically for this purpose, as was the rest of the analysis to be described. Once the data were extracted, an analysis of chord usage could be performed. There were, however, some complicating factors.

The first major complication was that the stored information could not be relied upon to be entirely accurate. The primary reason was that the songs were submitted to the wiki site, as indicated earlier, by anyone who was motivated to do so. This, sometimes, resulted in multiple submissions of the same song and, more to the point, submissions that did not always correctly identify the key. One frequent mistake was to neglect to identify a song as being in a minor key – the key signature was correct, but the song was labeled as being in the relative major key. Other times the key signature was just wrong (sometimes not entered at all, which then defaulted the song to C-Major). Another issue was inconsistent syntax, despite the Music-XML standard being used. For example, a major chord might be identified as either "maj" or "major".

In order to perform a meaningful analysis, these issues needed to be addressed in some automated fashion, as it was impractical to edit some 6500+ songs by hand. Syntax problems could be handled with simple translations to a consistent format. Multiple song entries could be filtered. The biggest issue, however, was in how to identify and correct incorrect information, particularly involving keys. The approach taken to identify the correct key was to take into consideration a collection of factors. These factors included:

- The stated key
- Beginning and ending chords
- Beginning and ending notes (in the melody)
- A listing of notes used in the melody, which were then compared to scale notes in a given key

From these considerations, one can draw inferences as to the likely key the song is in. An heuristic was developed which involved assigning weights to the above factors and then inferring the most likely key together with a confidence score. Generally, a song in a given key might be expected to end on the tonic chord and/or note, or, less frequently, the fifth. Similarly, it is not uncommon for it to begin in the same way. In addition, one might expect most of the notes in the melody to be scale notes in the given key. Thus, comparing the notes used in the melody to the scale notes in a given key can give an indication of the likelihood that the song should be considered to be in that key.

None of this is strictly required, of course, and songs with complex tonality may not necessarily conform to this, but, generally, if a weighting is assigned to each of these considerations and an aggregate score generated for each possible key, one key will stand out as the most likely. An example of the output of this analysis for one song is as shown in Figure 1.

Figure 1. Song Key Analysis

```
################################
song:      ************
listed:    C   47.70
inferred:  G   98.72
key:       G   99.37
#------------------------------
chords:        C           G
13   G         V           I
8    D         II          V
6    C         I           IV
4    D7        II7         V7
2    A         VI          II
2    Em        IIIm        VIm
endchords:
################################
```

The figure shows the score, out of a maximum value of 100, for two possible keys, the listed key in the song file and the key deemed most likely based on analysis. For this example, the listed key in the Music-XML file was "C". The analysis indicates that the song should, more probably, be considered to be in "G". Some of the evidence can be seen in the actual chords used and the number of times each appeared in the song, which are shown in the first two columns. Note that, while actual chord names are listed in the second column, the more generic, and key independent, Roman numeral designation [8] is also shown, making it obvious that the most used chord is the tonic, or "I" chord in G major.

Armed with similar information for all of the songs, one can then consolidate the results and generate a listing of most used chords and chord types for songs in both major and minor keys. Results of this analysis are shown, for major keys, in Table 2.

The table shows the number of times the listed chord type was used in songs in major keys. Although not shown, a similar table has also been generated for songs in minor keys. In the table, the chord type is in the first column followed by the total number of occurrences for that type of chord. The last 4 columns provide a breakdown of usage for each of 4 levels of the key score confidence mentioned earlier. "max" reflects a score of 100. "high" is anything over 80, "medium" is over 66.7, and "low" is below 66.7. Note that, as in the first analysis involving related keys, higher order embellishments, i.e. 9th, 11th and 13th chords were not considered. They were reduced to their underlying 7th chords.

Table 3. Tonic Chord Type Usage in a Major Scale

I	38083	I7	4591	I6	2347
Imaj7	2077	Idim	600	Im	371
Iaug	340	Isus	236	Im7	128
I7#5	114	Im6	102	Idim7	89
Ipow	51	Isus2	28	I7b5	17
Iped	12	Im#5	11	Im7b5	8
Imaj7#5	3	ImMaj7	11		

Table 2. Chord Usage for Major Keys

Chord Type	*Usage Count*	*Confidence Score*			
		max	*high*	*medium*	*low*
maj*	77451	61574	6965	7558	1354
7*	54019	44616	3354	4610	1439
m*	20214	14916	1924	2872	502
m7*	18712	15026	1223	1816	647
maj7*	4714	3570	270	648	226
6	4448	3756	143	457	92
dim*	3465	3170	94	150	51
m6	1773	1492	61	191	29
sus	1496	1109	170	170	47
7#5	1404	1200	72	84	48
m7b5*	1306	985	65	211	45
dim7*	1207	1000	35	138	34
aug*	1167	1058	35	60	14
7b5	519	391	27	65	36
sus2	156	108	13	34	1
pow	112	80	12	20	0
mMaj7	77	71	4	2	0
m#5	55	45	0	10	0
m7#5	18	15	0	1	2
ped	13	12	1	0	0
maj7#5*	9	9	0	0	0
maj7b5	9	2	2	1	4

One final observation is pertinent. Table 2 details the overall chord type usage in major keys. The analysis also yielded a breakdown, for every note, of which chords types are most used with that particular note as the tonic. An example of this, in Table 3, shows the types of chords found that were built on the tonic note in a major scale, i.e. Roman numeral 'I'. As might be expected, the simple major chord is most used. Again, similar tables have been generated for all notes.

3.3 Observations

These analyses, based on both fundamental music theory and on an investigation of "real-world" usage provide interesting insight into chord usage when one compares the results from each. As might be expected, there is significant overlap between the two analyses.

This can be seen in Table 2. When one ignores the scale tones upon which chords are built, there are only 9 unique chord types that result from the related keys analysis

discussed earlier. In the first column, which lists the chord types resulting from analysis of songs, asterisks have been added to those chord types that overlap with the types produced from the related keys analysis. All 9 types are present and, for the most part, are near, or at, the top of the list of most used chords.

There are a couple of notable exceptions. In particular, 6th chords turn out to be used a significant amount in practice, but were not considered in the related keys analysis. Likewise, suspensions are also missing. Again, those were not considered in the related keys analysis.

Armed with this information, one can feel confident that any instrument that can play all, or most, of the listed chords types in Table 2, would be useful, under most conditions, for musical exploration and/or expression.

4. Application Development

4.1 Background

The application/instrument, described here, takes advantage of much, but not all, of the knowledge gained from the analyses discussed earlier. In particular, work on the software was begun after a preliminary version of those analyses was completed. The main differences between those analyses, as compared to what has been presented, are as follows.

1) The theory based, related keys analysis, did not include consideration of relative and harmonic variants of minor keys. As a consequence, augmented chords, for example, did not show up on the list of chord types. Specifically, none of the chords in the last column of Table 1 were initially under consideration.

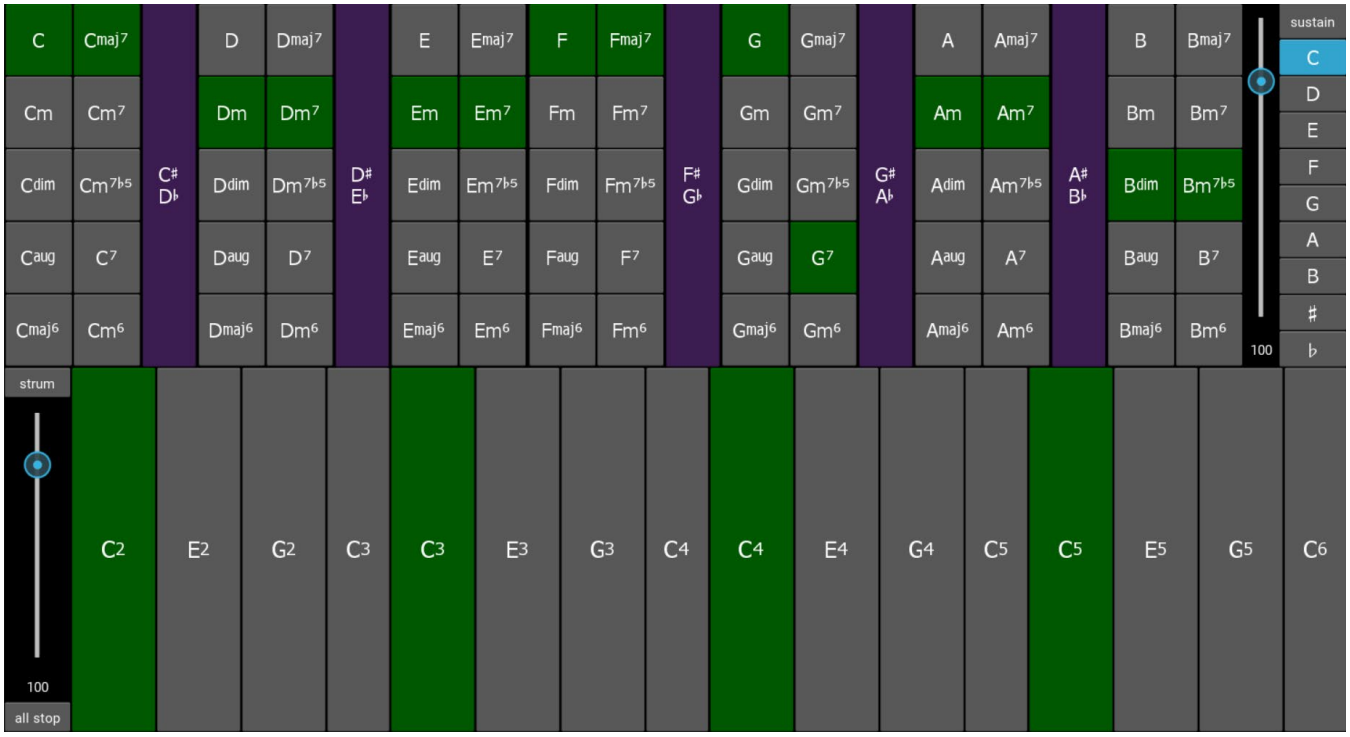
2) In the original data-mining exercise, all major keys were transposed to C-major and all minor keys to A-minor. However, the key recognition capability described earlier was far less advanced which led to a bit of a Catch 22 situation when it came to recognizing chords in the scale. The updated approach starts with a Roman numeral based approach and performs the key analysis as described earlier. The final result is independent of the actual original key, which is far more flexible for ongoing analysis. Still, the results were consistent, largely, with those reported above.

4.2 Application Interface and Design

Using this information, a basic application was written that displays a common set of chords for each scale note in a selectable key. In this application, only major keys are selectable but, of course, the same chords would exist in the relative (natural) minor key. The application screen is displayed in Figure 2 for the key of C-major (A-minor).

The operation is straightforward. The user simply touches a box in the upper portion of the screen that contains a chord name and the chord plays. In this application, chords are all played in root position. The layout is

Figure 2. Application Screen



reminiscent of a piano keyboard with the root note, C, in this case, to the left, and notes in the major scale arrayed to the right. Non-scale, accidental, notes are interspersed in between. The non-scale notes do not have chords listed. To play those, one needs to press both the accidental box and a chord in either adjacent scale note section to the immediate right or left.

Whenever a chord is pressed, the "keyboard" section in the lower portion of the screen remaps to notes in that chord. The keyboard covers 4 octaves and can be used to play the individual notes in the chord. So, one can either strum or arpeggiate the chord.

Key selection and key changes are accomplished using the vertical list of notes on the right. Whenever a new key is selected, the chord listing remaps so that the tonic note in that key is at the left, and all other scale notes are adjusted accordingly. Thus, the positions of the tonic, sub-dominant, dominant notes, and so forth, stay in the same relative positions regardless of the selected key.

The sustain and strum controls allow smooth transitions from one chord or note to another by holding what is currently being played until something new is played. Finally, volume and global note "all stop" features are included.

For each scale note, the basic triad and seventh chord that result from using only notes in the scale are highlighted (in green). Overall, the chords selected for inclusion began with the four basic triad chords: major, minor, diminished, and augmented (even though this last one was not indicated in the original related keys analysis). Four of the most commonly used seventh chords were also included as were two flavors of 6^{th} chords, based on results from the initial song analysis effort.

4.3 Implementation

As mentioned, the program is written in Python using the Kivy application development extension. Kivy supports multi-touch which enables the features described above to be implemented. Python is object oriented, so the code is organized taking advantage of that.

First, an object, called ChordInfo, was defined for managing music-related information. It contains the basic information needed for constructing the musical scales and chords supported by the app given key and chord names and types. A second object, MidiManager, handles communication with an external MIDI enabled player, which will actually play the selected chords. MidiManager uses the mido[16] Python library, which has routines for connecting to an external player and sending it messages via the MIDI protocol (Musical Instrument Digital Interface). MIDI is a well-established standard for controlling digital musical instruments [17].

The basic object used in the chord playing section is the ChordButton. These are the chord playing buttons that are visible to the user and each is assigned its own unique chord to play. ChordButton inherits the Kivy supplied Button object. In general, whenever a Kivy Button is touched on the screen, it calls a method in the Button object for processing that touch event. This method is overloaded in ChordButton which then determines what notes need to be played for its assigned chord, using the ChordInfo object, and then sends those notes to the MidiManager which, in turn, sends them to the external player.

ChordButtons are grouped together in ChordGroup objects, each containing 10 chords. When a ChordGroup is constructed, it is assigned a specific tonic note which is, in turn, passed along to each ChordButton in the group together with the type of chord assigned to that particular ChordButton. The net result is a grouping of 10 buttons each of which plays a different chord on a common tonic note.

The ChordBlock object assembles the ChordGroups into the playable chord section in the upper half of the app. Interspersed, where "black keys" would be on a piano, are TonicNoteButton objects whose purpose, when touched, is to change the tonic note of adjacent ChordGroups. Thus, if a TonicNoteButton and held, and a ChordButton in an adjacent ChordGroup is touched simultaneously, the chord played is transposed up or down one half step depending on whether the TonicNoteButton is to the right or left, respectively. In this manner, chords built on both scale and non-scale notes can be played.

The ChordBlock also contains a volume slider and sustain button for adjusting the volume and sustaining chords. When sustain is engaged, a chord will continue to play after the touch is removed and until either a new chord is triggered or sustain is deactivated (or "all stop" is touched).

Finally, the ChordBlock contains an array of buttons for key selection, with sharp and flat qualifiers. When a new key is selected, the chords, and their labels, are remapped to the new key, with the tonic note in the scale be assigned to the leftmost ChordGroup and the remaining scale notes assigned relatively by scale position. Thus, scale notes in any key are in the same relative positions regardless of key.

The keyboard section in the lower half of the app uses a similar approach, but plays individual chord notes rather than full chords. Here, there are 4 octaves of playable notes, but each octave has only 4 notes each. Each note is managed using a KeyButton object which inherits a Kivy Button object. Whenever a chord is touched in the chord playing section, the 4 notes in each octave are remapped to notes in that chord allowing the user to articulate the chord as they desire.

The implementation of the keyboard required special attention in Kivy in order to make it responsive to strumming type actions. Basically, the KeyButton objects need to be responsive not only to when a touch event occurs on the button, but also to touch movement events. The object needed to recognize when a touch point on the screen "collided" with the button, playing and stopping notes on collisions and exits. Kivy has methods for detecting such events, which facilitated the desired operation.

The keyboard section has its own volume and sustain capabilities which work similarly to those in the chord section. Finally, the "all stop" button stops any playing notes.

5. Conclusion

The stated goal, when undertaking this effort, was to create a musical application, or instrument, that would provide a workable set of playable musical chords in an approachable user presentation. As such, it would serve as a usable instrument for exploring musical ideas, learning about chords, or for basic accompaniment. Overall, the current implementation largely achieves that goal.

There are some limitations, however, and many opportunities for improvements and expansions. One limitation is that it is primarily set up for major keys. This could, and should, be expanded to include selection of minor keys. Also, in light of the more extensive analyses done subsequent to the initial development, a more expansive list of chords should be included. That analysis suggests too that all notes should be directly available for chord generation, not just the primary scale notes.

It is not clear that the keyboard, as implemented, is a useful feature. It takes a large amount of screen real estate and it is debatable that it is "playable". Perhaps eliminating it, or shrinking it into a "strum plate" might be advisable.

Some other possibilities include: making a configurable interface that allows a user to select which and how many chords to display; adding the ability to have different chord voicings[18] or root notes; adding embellishments such as 9^{th}'s, 11^{th}'s and 13^{th}'s to the chords; building in chord arpeggiations.

A more advanced idea would involve following the chord progression in a song and adjusting the chord presentation based on it. This would require additional theoretical and song analysis beyond what has been done so far.

Thus, it is submitted that this effort both achieved its targeted goal and has opened up many opportunities and directions for further research and development.

References:

[1] F. Romano, *Learning Python* (Birmingham UK: Packt Publishing Ltd., 2015).

[2] M. Vasilikov, *Kivy blueprints: build your very own app-store-ready, multi-touch games and applications with Kivy!* (Birmingham UK: Packt Publishing Ltd., 2015).

[3] A. Bellucci, M. Romano, I. Adeo, P. Diaz, Software Support for Multitouch Interaction: The End-User Programming Perspective, *IEEE Pervasive Computing*, 1(15), 2016, 78-86

[4] Suzuki QChord (electronic musical instrument), *Canadian Musician*, Sept-Oct, 1999, Vol 21 Issue 5, 71

[5] Y. Wu, Musicking with an interactive musical system: The effects of task motivation and user interface mode on non-musicians' creative engagement, *International Journal of Human-Computer Studies*, 39(1), 2015, 27-40

[6] C. Roberts, G. Wakefield, M. Wright, J. Kuchera-Morin, Designing Musical Instruments for the Browser, *Computer Music Journal*, 122, 2019, 61-77

[7] Web Audio API, https://www.w3.org/TR/webaudio

[8] M. Hewitt, *Music theory for computer musicians,* (Boston MA: CENGAGE Learning, 2008).

[9] Music-XML home page, https://www.musicxml.com

[10] T. Christiansen, N. Torkington, *Perl Cookbook,* (Sebastopol CA: O'Reilly, 2003).

[11] J. Clendinning, E**.** W. Marvin *The Musician's Guide to Theory & Analysis*, (NY, NY, W. W. Norton, 2016)

[12] Altered Chords, *Electronic Musician*, 35(1), 2019, 32-33

[13] F. Kinney, Beyond major and minor: A composer's understanding of chords and scales, *Clavier Companion*, 7(5), 2015, 38-42

[14] Wikifonia, https://en.wikipedia.org/wiki/Wikifonia

[15] A. Chan, J. Hsiao, Information Distribution within Musical Segments, *Music Perception*, 34(2), 2016, 218-242

[16] mido, https://github.com/mido/mido

[17] J. Rothstein, *MIDI: a comprehensive introduction* (Madison WI: A-R Editions, 1992).

[18] T. Gerken, Creative Chord Voicings, *Acoustic Guitar*, 22(8), 2012, 16-20

SELECTING AN APPROPRIATE INFORMATION SYSTEM DEVELOPMENT APPROACH FOR UNDERGRADUATE CAPSTONE COURSES

Pratibha Menon
California University of Pennsylvania
menon@calu.edu

ABSTRACT
This paper provides a basic framework for choosing the features of the waterfall, and the iterative methods of software system development in an undergraduate level senior capstone project. The paper explains some of the main constraints of a capstone project that differentiates it from a software development process in the industry. The contribution of this paper is to provide a framework that maps key features of waterfall, agile and hybrid waterfall-agile approaches, to its applicability in the software development process of a capstone project.

KEY WORDS
System, Development, Waterfall, Agile, Capstone

1. Introduction

Many undergraduate programs in computing or related disciplines such as information systems, software engineering, informatics etc. require students to complete a capstone project in their senior year. In these capstone projects, students work in groups to complete a significant project, which involves design and development of information systems for a real client, or business. The capstone give students an opportunity to bring together the wide range of skills they have acquired during their degree program. The capstone generally need students to apply hard skills such systems analysis, design, and programming skills that are required to develop computing solutions to a realistic and unstructured problem. In addition, a capstone also requires students to apply soft skills such as teamwork, conflict resolution, and interacting with project sponsors.

The capstone projects usually go beyond meeting student and program assessments, as they involve industry sponsors who have some real need for the software under development. The capstone experience prepares students for real-world experience before they move into the workplace. In a capstone project, the choice of development methodology is often a perennial subject for debate, along with the choices of programming languages, and the development environment. Selection of a systems analysis, design, and development methodology is pivotal to the projects, and this choice is typically based on a myriad of factors such as the size of the project, the budget, the team size etc. [1]. The key criteria in selecting the appropriate development approach in a capstone project is finding not only the best methodology to fit the nature of the project, but also be the best approach for teaching it.

Central to selecting an appropriate system development methodology lies the 'process versus product tension' [2] - whether the focus of the capstone course should be on the development process and the associated learning outcomes or, on the product of the development that will lead to a usable and working system. Since the capstone emulates work as it is performed in a professional environment, adoption of an appropriate design and development method is important. However, studies have [3] emphasized that students shouldn't think that following the process alone is enough even if it doesn't meet all the project's requirements for a workable product.

The key to reconcile the 'process versus product' tension may be to select appropriate project goals and development method. Traditionally, most capstone courses have used a derivation of the sequential approach called the waterfall approach to Software Development Life Cycle [4]. In recent years however, instructors have mixed components of traditional waterfall approach with the newer iterative development models and prototyping [5].

This paper is organized in the following way. First, the constraints on capstone projects are explained. Then, the sequential (waterfall) approach, and the iterative (agile) methods of software development are reviewed with respect to the constraints of capstone projects. This paper provides a basic framework for choosing between the waterfall and the iterative methods for a software development capstone project. The paper considers the possibility that senior capstone projects can be customized to meet the capstone constraints by accommodating useful features of both iterative and waterfall approaches to software development.

2. Constraints on Capstone Projects

While a capstone tries to provide students with real world systems design and development experience, it does not fully replicate a professional environment due to its teaching imperative. There are a number of constraints on capstone due to the fact that the project is carried out in an educational environment. These restrictions include the project duration, time and commitment of students, experience level of students and instructors, scope and complexity of project, technology available to students, and the need to meet assessment criteria of the educational program. Each of these constraints are discussed in detail.

A typical capstone can be a semester long project, where a typical semester might be 12, or 15 weeks in duration. In some cases, the capstone may be completed within two semesters. Here, the first semester may be used for systems analysis and design process, and the second semester may be dedicated towards systems development. The project deadlines are firm, unlike in real world projects where the project could be extended.

Experience levels of students are relatively low since most students would have little or no exposure to working on such long term projects. Students also have very little experience working on unstructured problems. In many cases, the projects will also require them to learn new programming languages and methods. Students are mostly inexperienced in applying project management, and in working in teams in a development environment. For many students this is also the first time that they have to work with project sponsors, who are generally not part of the academic environment.

Experience of Instructors will play a major role in selecting the scope of projects. This process typically takes place prior a project is assigned to students. The scope of the project should be such that the project can be completed by a team of 3-5 students, meet the assessment requirements of the course, and that it should satisfy the requirements of the project sponsors. Once the scope of the project is set, it is advisable to not change it during the project duration. Increase or decrease of scope during the project due to unforeseen conditions should be managed adequately by the instructor, so that the assessment criteria remains equally valid among all projects.

The fact that the capstones are used as an assessment tool for a program, adds additional complexity to the projects [6]. In addition to producing the product or system as per the sponsors' requirements, the capstone will also require students to produce extensive documentation for assessment purposes. Such documentations may not always be required in a real world project setting, where much of the time may be spent to development, prototyping, and user acceptance testing. Documents such as a comprehensive specification document, class presentations, work logs, essays on technology adoption etc. add additional time constraints on the students. A typical student may not devote 100% of their time on the capstone, as they may need to balance other courses as well.

Use of technologies due to budgetary constraints of the program may pose additional constraints on the capstone. If a project cannot be funded by a sponsor, the technologies and development environment are created within the academic setting with whatever funds and support are available. Instructors may not be proficient in all the technologies and their willingness to provide support on technologies not covered as part of the curriculum might be limited. Most software capstones commonly use object-oriented languages, web application development, and use of relational databases. Other technologies that appear are cloud technologies, mobile devices, several development platforms, and programming languages.

A project sponsor in a capstone differ in characteristics compared to the clients in a larger project. A typical sponsor in a capstone may be a small business, or a non-profit with limited funding for the project. The knowledge of the sponsors about information systems might be limited. For example, some sponsors who might have been used to storing data in spreadsheets, might not realize the importance of having databases. Not incorporating a database into the project might not be an option for the project if the assessment requires students to create one. Additionally, sponsors may not have the time to devote to evaluating prototypes, and that might not help towards creating usable systems.

All these constraints that are typical to a capstone influences the 'process versus product' decisions that need to be resolved for a successful capstone project. Selection of an appropriate system development methodology for a given capstone project will require substantial decision making on part of the instructors on the suitability of the existing development methods in light of the constraints presented by the capstone.

3. Sequential versus iterative approaches to systems development

Over the years, information systems have dramatically evolved in terms of its complexity. An information systems development methodology is a collection of particular systems development assumptions, a set of strategies, principles and guidelines, a multi-step procedure of what to do and how to do things. The development of computer information systems went

through revolutionary advances over the years owing to the advances in computing and the changing IT needs of organizations. This evolution has resulted in the creation of various development methodologies designed to address various development issues. The development methods can be broadly classified into two types of models – the sequential, and the iterative development models.

The sequential model, also known as the waterfall approach to Software Development Life Cycle (SDLC), was one of the earliest system development methodologies to apply engineering to software system development. Variants of this model still continues to be used today. Opposed to the sequential method, is the iterative method of development that emerged during the 80s and 90s, and continues to dominate the software and systems development to this day.

Figure 1 depicts a version of the modern SDLC/waterfall model [7] that considers the four systems development phases - Project Planning and Selection, Systems Analysis, Systems Design, and Systems Implementation & Operation.

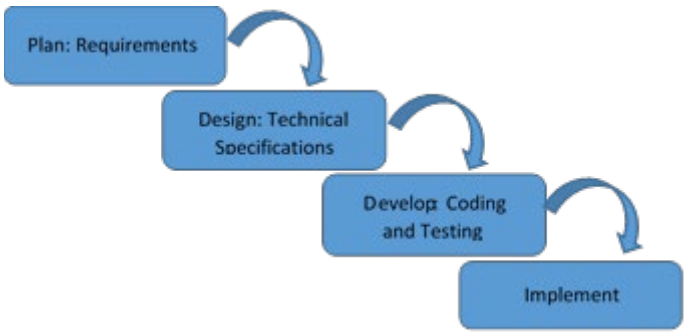


Figure 1. Waterfall model

Sequential life cycle models assume a certain context for their use. They assume that the software requirements can be well understood during the systems analysis stage that errors made in a phase will be caught in the next phase, that the systems design pattern and application technologies have been used for the same or similar applications, or that a complete life cycle development involving all four phases is necessary. If none of these contexts applies, then an iterative life cycle model may be more applicable. Iterative development models have several cycles of a prototype phase and a corresponding evaluation phase. The results of an evaluation phase are then input into the prototype phase for an iteration, or to an engineering phase to formalize the prototype into the system. Using an iterative model, the information system is developed incrementally after several cycles of prototyping and evaluation.

In 2001, a group of software engineers working in alternative development methodologies signed the so-called manifesto for Agile software development. Unlike the traditional waterfall methods that focuses on process, extensive documentation, extensive task planning and sequential phases, agile manifesto focuses on teams, working and usable software, strong customer collaboration and responsiveness to changes according to a plan. Agile manifesto suggests that using short iterative cycles called 'sprints', during which developers work closely with customers to achieve better communication, to manage design changes, and for quick completion of products and systems. Agile avoids 'up-front' requirement gathering from stakeholders, who often could not provide all requirements in sufficient details for development to occur at the beginning of a project [8]. Figure 2 depicts the iterative sprints of an agile method.

One of the key features of Agile method is that it significantly reduces the time required to document the design, unlike what a waterfall SDLC does. Despite improving the efficiency of the development process, lack of extensive planning and documentation could pose difficulties for novice developers, who may need more clarification from the experienced developers on the requirements and processes. On the other hand, a traditional waterfall SDLC with its detailed specifications will provide clarification to communicate the requirements and guidelines about the project to the development team.

Yet another key feature of agile is that it requires strong communication among team members and frequent meetings with the customer to test and evaluate the prototype. Furthermore, the time frame allocated for each iteration is typically short, which is usually in the range of a few weeks. This would make the development time short and several times the iteration delivery can be dragged. Managing such uncertainties for each iteration cycles requires establishing efficient and strong communications with the customers. On the other hand, a traditional sequential SDLC process collects detailed requirements before the development begins. The customers may not be involved during the development process at all and the developers need not be concerned about frequent meetings with the customers.

There are several of recognized Agile methods [8] which generally includes Extreme Programming (XP) [9], Scrum [10] and Crystal methods [11]. While experience reports on the use of XP and Scrum can be commonly found, reports on other methods are less available.

4. Using waterfall model in a capstone

Despite having a variety of development methods to choose from, most academic programs use the traditional,

sequential, waterfall model to teach a course in Systems Analysis and Design. The basic components of SDLCs function as the building blocks for various systems development methodologies. From a training stand point, use of the building blocks of the traditional waterfall based sequential SDLC model provides the fundamental knowledge to developers, who do not have the experience, or knowledge of best practices to address various systems development situations. Because of this nature of waterfall model, it dominates most of the popular textbooks on Systems Analysis and Design.

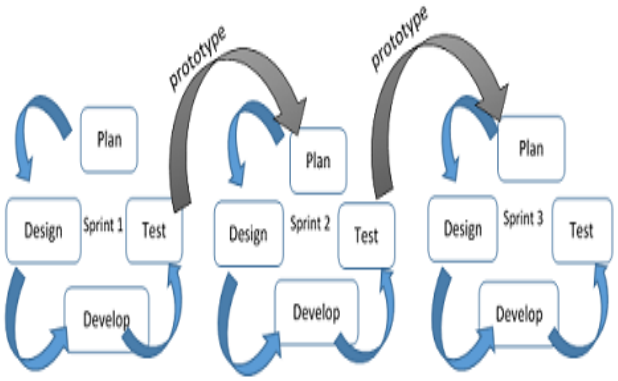


Figure 2. Agile model

While the waterfall continues to be the prominent methodology used in several capstone courses, it problems are apparent in several areas [12]. One of the drawbacks of the sequential SDLC method is the late delivery of product that only happens towards the end of the cycle. Students may spend a large amount of time producing the analysis and design documents, and sponsors will begin to see the product only once the development process has begun.

Students may not be experienced in identifying the functional and user requirements appropriately and therefore, the additional requirements that may crop up during the development stage, may need revision of the analysis and design documents. In many cases, if the sponsors are not tech savvy, they may not even know about the capabilities that technology might provide. Therefore, once the product begins to take shape the sponsors may require additional useful functionalities, or usability features that they might not have been aware of previously. Given the inexperience of students and sponsors, a scheduled revision of requirements, design and code would be helpful to produce a workable and usable product. To meet this goal, it require the students to building in iterative cycles of prototype delivery and user feedback, within the development process.

5. Using agile model in a capstone

Report on the use of agile methods for capstone projects are available in the literature. Methods such as the Scrum and the XP have been used for one or semester-long capstones. Studies by Dubinsky and Hazzan [13] reported increased awareness of customer needs and testing issues when using XP. Keefe and Dick [14] have identified the need to teach the XP techniques in earlier programming courses, so that students can be coached further during the capstone project. The key features of XP are [9]:

- continuous and concrete feedback from short cycles of development,
- an incremental approach to planning,
- an adaptive approach to the way businesses requirements change,
- using testing, source code and oral communication as a means to communicate the systems structure and changes,
- an emphasis on test-driven development,
- a continuous and evolving design.

There are several advantages of XP that makes it an attractive option for software development in capstone. It clearly defines techniques for using coding standards, test-driven development, refactoring and continuous integration. XP is designed for small teams and heightens communication using pair-programming. The XP approach has clearly defined roles for team members that can also be adapted to the given project context.

Despite its advantages, the XP method has certain drawbacks when applied to capstone projects. Most of the documentation is embedded in the code, which may not be suitable for assessment. While XP methods requires customer feedback every three weeks into the project, this would not be possible for every capstone team. In such cases, the instructor would need to be a proxy for the customer. An instructor's experience and comfort in using development tools for testing and continuous integration of modules during the development process will play an important role in adoption of XP method. Some instructors are averse to running a senior project without adequate documentation and consider agile methods to be an ad-hoc development method that does not take the long term view of system [15]. In general, there is a consensus among instructors that the adoption of XP practices as such, in senior project is too complex.

The adoption of XP in senior capstone would require the instructors to rethink about the goals of the capstone. If the goal of the capstone is to teach students about software engineering practices that can be used to deal with large and complex systems that requires documentation for compliance purposes, then XP clearly doesn't meet these goals. However, if the goal of the

capstone is to teach students how to work in small teams , how to design, develop and test functioning and usable software in small time frame, then the studies indicate that development project of a capstone. A scrum [16] project is made up of sprints, which are short durations of time, from about 2 to 4 weeks, where potentially deployable features must be completed.

XP might be a good choice. Besides XP, another agile model called the scrum has been used to manage the

Table 1: Comparison of systems development approaches and their implications in a capstone.

Methodology	**Unique Characteristics**	**Implications in a Capstone**	**Artifacts used for Assessment**
Waterfall SDLC - with four phases: Planning, Analysis, Design, Implementation	Suited for large projects Extensive Documentation Limited Prototyping Several models are used Longer planning and requirements gathering phase	Student familiarity - Commonly taught in IS based courses. Assumes requirements can be accurately obtained before developing system. Schedule slippage is possible with less time on implementation and testing. Project duration will be spent on detailed documentation. Easier to assess and to replicate assessments due to common documentation.	Process diagrams , Entity Relationship Diagrams, Activity diagrams etc. Comprehensive Requirements documentation. Scope document. Project Management Plan, Work Breakdown structure. Database Script. Software Code. Product documentation
Iterative : Agile Methods: Design--Develop--Test cycles. XP Scrum	Suited for smaller teams. Suited for projects with uncertain requirements. Prototype driven - early production of code. Greater emphasis on quality and testing. Very little documentation Requires high collaboration and team work. Client involvement is very important. Can accommodate late changes in requirements.	Inherently provides a framework for good team support. Will require additional documentation for assessments purposes. Client involvement is important. Students should have previous experience with Pair-Programming, use of software standards and test driven development methods. Instructors might need to give greater weightage to team work and communications during assessment.	Task backlogs Prototypes Test plans Design documents in the form of UML, wireframes diagram for web pages etc. User feedback and evaluation.

	A typical Scrum sprint can last about 3 weeks		
Hybrid model - Iterative + Waterfall	Ability to customize development method. Focus on business value and quality of products. Use of iterative cycles, prototypes. Use of initial project planning and requirements specifications. Planning + Sprinting method (Baird, 2012) : The initial sprints focus of project planning and documentations .Later sprints focus on a draft prototype based on critical path and later a final prototype. Blending Agile and Waterfall documentation(Rahmamian, 2014): Incorporating "just enough planning" upfront" and implementation of agile method during the development testing and implementation phase	Incorporation of sprints during development phase creates a structured method for students to work as a teams. Use of software design and testing practices will help established common and disciplines work habits. Prototyping allows students to gain expertise on incremental development methods. Initial planning phases will help students get an overall understanding of system requirements. Documentation will make assessment possible. Suitable assessment methods are required to assess activities during the sprints.	Initial documentation of process diagrams, ERD, requirements specification and identification of critical path. Sprint plans that include a documentation of backlogs, results and improvements. A draft prototype that includes tasks in the critical path. Interim and final prototypes that includes a bug free code/system.

Goal of each sprint is to complete backlogs, which are prioritized lists of tasks that are waiting to be completed. Within each sprint, a small team of developers selects a subset of prioritized activities from the backlog that they believe they can complete within the duration of the sprint. Each day during the sprint, the team gets together to individually answer the following questions: 1) what each member has completed, 2)what each member plans to do between now and next meeting, and 3) identify obstacles that will prevent each member from completing the tasks. At the end of the sprint, the potentially deployable features are demonstrated to the product owner, who will also typically manages a product backlog. Studies have also shown that due to lack of experience, task estimation and planning during each sprint is what many student teams struggle with [17].

6. Using hybrid –waterfall- agile model

A blended methodology that combines the key strengths of waterfall and agile approaches have been proposed in the industry due to various reasons. One of the reasons for blending the two approaches is due to the documentation requirements for ISO compliant organizations that practice agile methods [18]. McGovern [19] suggests use of lightweight documentation for agile practices and this is primarily for large projects transitioning from waterfall to agile practices. Rahmamian [20] suggests blending the two methods by having the traditional upfront method during planning, and the use the Scrum cycles for design, implementation and testing. Initial planning and documentation can reduce the risk of confusion in project objectives and deliverable and the agile method can speed up design and development.

Scrum has been adopted in capstone courses by including features of both waterfall and scrum models [5]. The idea of sprints have been modified such that the first two sprints are used towards systems analysis, planning and documentations. The later sprints have been used towards developing the system and to create prototypes at the end of each sprint. The clients were not as heavily involved as would be the case in a typical Agile method, rather the clients were often communicated with as necessary. The study showed that client involvement did have a big

impact on the success of the project, as did the involvement of students to update their backlogs often.

7. Conclusion

Information systems development methodology is an important aspect of systems development projects. Table 1 provides a comparative analysis of several of the methodologies, along with their implications in a capstone course. We have shown that capstones can choose between sequential waterfall based models, iterative XP or Scrum models, or their combinations in the form of a blended waterfall- agile approach. Each approach has benefits and drawbacks when applied in the capstone environment. Choice of an appropriate methodology requires considerations of capstone constraints, and the context of the project. An optimal combination of sequential and iterative methods may also be required to mitigate the 'process versus product' tension of the capstone project.

References

[1] Beasley, R. E. (2003). Conducting a successful senior capstone course in computing. *Journal of Computing Sciences in Colleges,* **19**(1): 122-131.

[2] Clear, T., Goldweber, M., Young, F. H., Leidig, P. M. and Scott, K. (2001): Resources for instructors of capstone courses in computing. *ACM SIGSE Bulletin,* **33**(4): 93-113.

[3] Chamillard, A.T., & Braun, K.A. (2002). The software engineering capstone: structure and tradeoffs. *SIGCSE*.

[4] Hoffer, J.A., J.F. George, and J.S. Valacich (2005) *Modern Systems Analysis and Design*, (4th ed.), Upper Saddle River, NJ: Prentice Hall.

[5] Baird, A. & Riggins, F. J. (2012). Planning and sprinting: Use a hybrid project management methodology within a CIS capstone course. *Journal of Information Systems Education, 23(3), 243-257*. ISSN: 10553096

[6] Mann, S. and Smith, L. (2005). Technical complexity of projects in software engineering. *Proc. 18th Annual Conference on Computing Qualifications (NACCQ 2005)*. Tauranga, New Zealand.

[7] Valacich. J, Hoffer, J & George. J. (2005). *Essentials of System Analysis and Design*.

[8] Strode, D. E. (2006): Agile methods: a comparative analysis. Proc. 19th Annual *Conference of the National Advisory Committee on Computing Qualifications* (NACCQ 2006), Wellington, New Zealand. *Wellington, New Zealand.*

[9] Beck, K. (2000): *Extreme programming explained: Embrace change*. Boston, Addison-Wesley.

[10] Schwaber, K. and Beedle, M. (2002): *Agile software development with Scrum*. Upper Saddle River, New Jersey, Prentice Hall.

[11] Cockburn, A. (2002): *Agile software development*. Boston, Addison-Wesley.

[12] Mann, S. and Smith, L. (2006). Arriving at an agile framework for teaching software engineering. *Proc. 19th Annual Conference on Computing Qualifications (NACCQ 2006)*. Wellington, New Zealand.

[13] Dubinsky, Y. and Hazzan, O. (2005): The role of a project- based capstone course. *Proc. 27th International Conference on Software Engineering*. St. Louis, MO, USA, ACM Press.

[14] Keefe, K. and Dick, M. (2004). Using Extreme Programming in a capstone project, *Proc. 6th Conference on Australasian Computing Education - Volume 30*. Dunedin, New Zealand, Australian Computing Society

[15] Schneider, J. and Johnston, L. (2003): Extreme programming in universities: An educational perspective. *Proc. The 25th International Conference on Software Engineering*, Washington, DC, USA, 594-599, IEEE Computer Society.

[16] Rising, L. and Janoff, N.S. (2000) The Scrum Software Development Process for Small Teams. *IEEE Software*, 17, 26- 32. http://dx.doi.org/10.1109/52.854065

[17] Mahnic, V, "Teaching Scrum through team-project work: students' perceptions and teacher's observations," *Int. J. Eng. Educ.*, vol. 26, no.1, pp. 96–110, 2010.

[18] Binder, J., Aillaud, L. IV, & Schilli, L. (2014). The project management cocktail model: An approach for balancing agile and ISO 21500. *Science Direct*, 119, 182-191. http://dx.doi.org/10.1016/j.sbspro.2014.03.022

[19] McGovern, F. (2010). Blending traditional and agile project documentation. http://www.visiblethread.com/wp-content/uploads/Lean-Documentation BlendingTraditional-and-Agile-Project-Documentation.pdf

[20] Rahmamian, M. (2014). A comparative study on hybrid IT project management. *International Journal of Computer and Information Technology*, 03(05), 1096-1099. ISSN: 2279 – 0764.

http://www.ijcit.com/archives/volume3/issue5/Paper030535.pdf

COMPUTING INFRASTRUCTURES TO SUPPORT CYBERSECURITY EDUCATION

Linh B. Ngo, Liu Cui, and Si Chen
Computer Science Department
West Chester University of Pennsylvania
{lngo, lcui, schen}@wcupa.edu

ABSTRACT
As cybersecurity education emphasizes hands-on learning activities, significant efforts have been expended in developing and publishing course materials that satisfy these requirements. These materials cover a wide range of topics and require significant computing infrastructure support. At large institutions with emphasis on research activities, there are redundant facilities to provide students with an isolated environment where they have access to adequate infrastructures and personnel support from teaching assistant and professional IT staff. These resources are not always available for regional teaching-oriented institutions. In this paper, we discuss our recent experience at West Chester University in utilizing various local and national computing infrastructures to teaching computer security in different courses at different levels.

KEY WORDS
Computer security, computer education, computing infrastructure, cloud computing, CloudLab, XSEDE.

1. Introduction

The creation of the National Initiative for Cybersecurity Education [1] and the two National Center of Academic Excellence (CAE) programs in Cyber Defense and Cyber Operations [2] has signaled an official focus of national efforts to the promotion of cybersecurity awareness through education, training, and workforce development. This has resulted in a rapid growth of various security-related educational components at academic institutions across the nation, with 182 colleges and universities already acquired CAE in Information Assurance Excellence designation by 2014 [3].

One critical component of cybersecurity education is the inclusion of hands-on learning activities. In [4], the Department of Homeland Security recommended improving qualifying criteria for its CAE programs to promote the addition of more hands-on content. This presents a challenge to academic institutions in balancing between training, which requires more practical contents, and education, which requires more theoretical contents. This balance is most notable at institutions that provide 4-year Bachelor and Masters degrees [5]. As a result, recent literature in developing and publishing courses for cyber-security focuses on the developments of lab-based materials that can illustrate fundamental computer security concepts using hands-on learning activities on various relevant computing infrastructures. Adopting these materials, in many cases, requires a close replication of host institutions' infrastructures. For teaching institutions with limited resources, this represents a significant challenge.

In this paper, we discuss our recent experience incorporating hands-on learning activities in different computer security courses with the utilization of various local and national computing infrastructures. The remaining of the paper is as follows. Section 2 provides a brief overview of previous work in cybersecurity educational contents, with a focus on how the accompanying computing infrastructures differ over time. Section 3 describes the settings and materials for cybersecurity courses at West Chester University. Section 4 presents different approaches in provisioning computing infrastructures for students in these courses. We discuss the teaching experience, challenges, and student feedback in Section 5. Section 6 concludes the paper and elaborates on future work.

2. Related Work

Cybersecurity education literature typical follows several major approaches in providing computing resources for hands-on activities. In the first approach, students have direct access to a physical compute resource, which could be a personal computer (PC) or a shared network of computers. In [6], the authors described the most preferred laboratory exercises as ranked by students for introductory cybersecurity courses at two liberal-arts colleges. These exercises cover areas such as cryptography, network monitoring and analysis, secure connection, network security configurations, and network programming. Exercises in the area of cryptography are doable on students' PC, and others in a dedicated computer lab. Laboratories with focus on network security, including eavesdropping attacks, dictionary attacks, SSL usage, man-in-the-middle attacks, and firewalls/VPN were described in [7]. The accompanying computing infrastructure was

consisted of four computing clusters, each of which had three individual computers. In the paper, the authors describe having eight students per cluster as adequate. The work of [8] described an extensive laboratory network that enables various advanced network security exercises including network mapping and sniffing, vulnerability assessment, rootkit analysis, backdoors, honey pot, and worms. Students were also provided with removable hard drives that could be detached and taken home at the end of the labs.

The rise of virtual technologies, particularly virtual machines, has enabled the second approach for provisioning computing environment for cybersecurity education [9]. One example of this approach is the SEED Lab [10], which provides an extensive suite of security labs based on pre-made VMs. Students can download and deploy these VMs directly on their laptops to set up these labs. The creation of large-scale virtual networks such as Emulab [11] across a federation of institutions gives rise to platforms such as Deter-Lab [12], which supports virtual, isolated environments for network-related security courses. In recent years, the wide spread availability of public and private cloud computing infrastructures also contribute to a variety of new course materials [13-14].

The above overview demonstrates that while there exists a wide variety of contents for cybersecurity education, these contents are also designed and customized for specific computing infrastructures. It is critical for instructors to adapt materials such that the corresponding infrastructure requirements are accessible to students and also suitable and sustainable at the local institution.

3. Educational Settings

At West Chester University, the cybersecurity career track includes three courses that directly cover cybersecurity. In CSC 301, students learn about basic topics in computer security and the ethical underpinnings of security. CSC 302, Computer Security, introduces critical and diverse topics in general computer security. Topics in computer security (CSC 497/583) is a topic course cross-listed both at undergraduate and graduate level that focuses on specific advanced security problems. The actual topic is going to be announced at the time of offering. Currently, we offer two topics: software security and modern malware analysis. The majority of hands-on activities in computer security are covered in CSC 302 and CSC 497/583.

3.1 CSC 302 – Computer Security

CSC 302 is taught in two sessions, each of which covering different aspects of cybersecurity, including security principles, concepts (software, system, and network security and cryptography), components, and architectures. It is not a required course for computer science major students. However, it is a required course for the computer security certificate, which is accredited by NHS/DHS' National Centers of Academic Excellence program. Prerequisites for CSC 302 includes CSC 301 Computer Security and Ethics, and three Java courses (CSC 141, CSC 142, and CSC 240). In addition to learning about ethical issues related to cybersecurity, students in CSC 301 also learn authentication, attack models, access control, malware, risk analysis, intellectual property, information warfare, and security issues in distributed systems, such as database, email, payment systems, social networking, and cloud computing. By finishing all Java courses, students are capable of writing programs with generics, inheritance, and master all linear data structures such as Array, Array List, and Linked List. At the same time, the core prerequisite courses do not always provide students with Linux system administration skills and experience in working with a command line interface.

3.1.2 Sessions 1 of CSC 302:

This session of CSC 302 covers topics in network security, system security, and cryptography. The major hands-on lab in this session, called **game-below**, involves several teams trying to gain administrative access on computers administered by other teams. The intent of this game is to give students greater insight and understanding into the nature of computer security issues by allowing them to act as both "hackers" and "administrators" in a live network environment. The entire environment in which this game takes place is known as the sandbox. Students are divided into eight teams, with four students per team for this lab. The teams have three major tasks, which spread throughout the semester.

The first task is for each team to set up their own server in the sandbox. This requires Linux system administration skills. The teams are able to choose and install their own Linux distribution on a server and deploy it in the sandbox. At a minimum, this server must support SSH, Web service, and Mail service. The instructor offers extra credits for additional security services such as two-factor authentication, flash, firewall, and SQL server.

In the second task, the team begins their exploration of the sand box and attempt to gain unauthorized access to systems belong to the other teams. While doing these activities, the students also need to avoid detection while engaging in these activities. The second task will be carried out at the same time as the third task.

The third task is to defend and maintain each team's individual system. As part of this task, each team is to set up and observe their own activity monitoring mechanisms. Through the monitoring logs, if an attack is detected, the defending team is responsible for carrying out a proper respond. During the process, if there is any security related updates for the Linux distributions or installed software component released. It should be noted that as part of the rules, all students have standard user accounts on every other teams' systems. Therefore, the defending team needs

to discern between normal authorized activities and unauthorized attempts to gain entry to the system.

Students are to maintain rigorous documentation for all activities related to the three tasks. More specifically, they are to provide detailed descriptions regarding their own exploits as well as detection of other teams' exploits. A final written report and in-class presentation accounts for a significant part of the course's final grade.

3.1.2 Sessions 2 of CSC 302:
The second session of CSC 302 focuses on software security and web security. The hands-on lab materials are from the SEED lab series [10]. Taking into consideration the lack of Linux experience and C programming knowledge, introductory and hands-on practices for these topics are interleaved between the security lectures and labs. For software security, the session covers set-uid, environment variables, buffer overflow, and return-to-libc attacks. The topics selected for web security are cross-site request forgery, cross-site scripting, and SQL injection.

While the lab materials are provided by SEED, the accompany VM is limited at 32-bit with an older distribution of Ubuntu (16.04). To help students in understanding the technical aspects the lab, an additional team project is assigned. In this project, students are to first re-implement the lab in a 64-bit Ubuntu 18.04 VM. More specifically, they are to implement the set-uid and environment variables lab and then to select an additional lab from either software security or web security for re-implementation. To be able to create the memory-based vulnerabilities in the new 64-bit environment, students will have to leverage their understandings of the SEED materials for the 32-bit VM and apply them to the new 64-bit architecture. Similarly, in web security, the students will need to understand aspects of the web software stacks in the SEED VM that lead to the existence of the exploits for reimplementation purposes. Next, students are to automate these implementations so that their VM can be dynamically deployed in CloudLab, an academic cloud computing environment [15].

3.2 CSC 497/583 – Topics in Computer Security

Topics in computer security (CSC 497/583) is a free elective course for students major in computer science. This course carries a prerequisite of computer organization (CSC 242) that teaches the basics of assembly language, C programing language, bitwise operations and the concept of pointers. The expected demographic for this course includes students with basic programming concepts, familiarity with Unix/Linux environments including command-line shell and gdb, and with web programming concepts.

3.2.1 Topic in Software Security
The primary objective of the software security course is to let students understand how to assess software vulnerability; this process includes learning how to dissect an application, discover security vulnerabilities, and assess the danger each vulnerability presents. The course focuses on teaching practical offensive security skills in binary exploitation and reverse engineering through a combination of interactive lectures, hands-on labs, and it also offers students an opportunity to explore the state-of-the-art subjection in both the industry and academic world.

The course first covers legal aspects of reverse engineering, assembly language for IA-32 compatible processors, vulnerability analysis, and Linux-based ELF program exploitation. It will then transition into several protection mechanisms found on modern systems (e.g., DEP, ASLR, Canaries, and PIE) and the techniques used to defeat them. The course also covers other subjects in exploitation including advance heap exploitation and kernel-land exploitation.

3.2.2 Topic in Modern Malware Analysis
The primary objective of the modern malware analysis course is to let students learn how to detect, analyze, and eradicate malware since it becomes an increasingly important issue of economic and national security. The course introduces students to malware analysis techniques through lectures, readings and hands-on interactive analysis of real-world samples. Students should be able to analyze contemporary malware using both static and dynamic analysis after taking this course.

In this course, the instructor first covers the basic concepts of malware analysis, including the experiment environment setup, debugging concepts and tools, definition of static analysis and dynamic analysis, and PE/ELF format. Next, the course moves on to study malicious activities, techniques, and countermeasures (e.g., DLL Injection, API Hooking). Rootkit techniques including packing, patching, and direct kernel object manipulation are also discussed.

4. Supporting Infrastructures

The varieties in contents of the courses described in Section 3 lead to different infrastructure requirements for students' learning activities. These infrastructures include on-site computer networks, bare-metal cloud computing environments, and virtual machines deployment on individual students' laptop.

4.1 Session 1 of CSC 302

In this session, students have access to a sandbox infrastructure consisting of eight computing servers, connected through an Ethernet switch. The instructor commands one additional machine to manage and oversee the sandbox. The sandbox itself is deployed within a CS-specific computer lab, with a portion of the physical machines in this lab is dedicated to the sandbox. Students not enrolled in the session are not allowed to use these

machines during the semester. The overall architecture of the sandbox is shown in Figure 1.

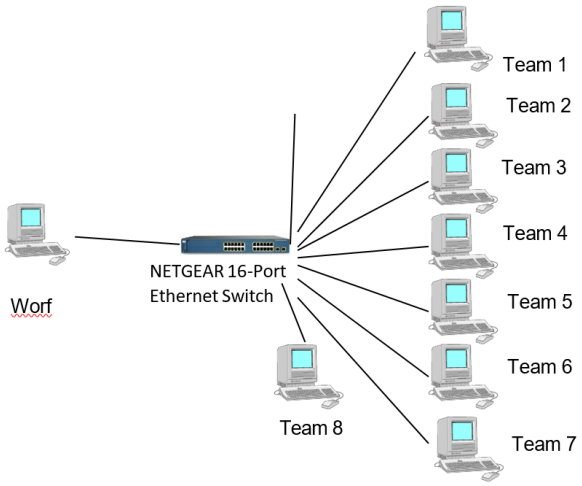


Figure 1. Sandbox Infrastructure

4.2 Session 1 of CSC 302

For the SEED labs, students download the original VM to run on their laptop using VirtualBox. To help integrating lecture materials and lab materials, students also download and install Anaconda inside their downloaded VMs. Anaconda's Jupyter server can be launched from inside a VM and has its port forwarded to host computers. With the lecture contents and lab activities converted to Python Jupyter notebooks, students can access the lab environment inside the VM and the lecture materials through a web browser.

To support the creation of an additional 64-bit VM for the reimplementation of SEED, students are granted access to a cloud-based experimental computing environment called CloudLab. Funded by the National Science Foundation in 2014, CloudLab's goal is to provide researchers with a robust cloud-based environment for next generation computing research [15]. These resources are distributed across several U.S. institutions. As of August 2018, CloudLab boasts an impressive collection of hardware. At the Utah site, there was 785 nodes, including 315 with ARMv8, 270 with Intel Xeon-D, and 200 with Intel Broadwell. The compute nodes at Wisconsin included 270 Intel Haswell nodes with memory ranging between 120GB and 160GB and 260 Intel Skylake nodes with memory ranging between 128GB and 192GB. At Clemson University, there are 100 nodes running Intel Ivy Bridges, 88 nodes running Intel Haswell, and 72 nodes running Intel Skylake. All of Clemson's compute nodes have large memory (between 256GB and 384GB), and there are also two additional storage-intensive nodes that have a total of 270TB of storage available.

In order to provision resources using CloudLab, a researcher needs to describe the necessary computers, network topologies, start-up commands, in a resource description document. CloudLab provides a graphical interface inside a web browser that allows users to visually design this document through drag-and-drop actions. For large and complex profiles, this document can also be automatically generated via Python in a programmatic manner. Starting Fall 2017, CloudLab supports a direct integration between publicly readable git repositories and their profile storage infrastructure. This significantly minimizes the effort needed to modify existing profile while still maintaining a complete history of previous changes.

4.3 CSC 497/583

For the software security course, a virtual machine disk image is created. This image runs the Manjaro Linux system which has all the tools, challenges, and coding libraries. Students need to download the image and import it to their computer via VirtualBox (or VMware). A remote server with Ubuntu Linux, and Docker is set up to create a capture-the-flag (CTF) environment for labs and the final project. The student can connect to the remote server through a Python script, and the goal is to reveal the secret stored in a file.

For the modern malware analysis course, a Windows XP 32-bit virtual machine is set up. Unlike the Linux system, the Windows system is not free and cannot be distributed this VM to the student. Instead, the instructor provides a comprehensive list of tools and installers involved in the course. All the labs can be done in the Linux environment by choosing a cross-platform Python library.

5. Teaching Experience, Challenges, and Student Feedback

Since CSC 302 and CSC 495/583 are elective courses, students enrolled in these courses have with different technical background. As a result, students' learning experience, challenges, and course feedback varied across the courses and the sessions.

5.1 Session 1 of CSC 302

One of the authors have taught Session 1 of CSC 302 for three semesters. The game-based assignment enabled students to learn using a project-based learning approach. Recognizing that many students were not familiar with Linux, the instructor first sent out a survey with Linux and networking related questions and then used the results of the survey to divide students into groups with similar aggregated level of technical skills. The instructor made sure that at least one student is very familiar with Linux and one or two of them has/have working knowledge about Linux.

5.1.1 First semester

At the beginning, the instructor prepared and taught the class under an ideal assumption that as long as adequate

directions were provided, students would engage in the project and learn by themselves. However, it turned out to be not the case. The instructor received feedback for both logistic and technical issues. Logistic issues mainly focused on time and collaboration. Although enrolled students in this semester were at junior and senior level, they complained about not having enough classroom time for the project. This was contradictory to the intention of the project, which was to be done after the class in lieu of homework assignments. Some group also complained about team-members do not show up, etc.

The students also encountered more technique issues than expected. One of the primary learning activities for students in the first task was to research through different Linux systems, select one, and follow online documentation to install the machine. Several groups emailed the instructor to require explicit instructions regarding the installation of Linux operating system. There had also been requests for in-class demonstrations of this process. Similar issues happened with the installation and configuration of the required services such as SSH, Web, and Email servers. Fluency in programming techniques, which came from the prerequisites, did not translate into experience with system administration knowledge. For example, students could quickly create and publish a web page, but not able to setup the corresponding web servers. During the intrusion and defend stage of the second and third tasks, the most common feedback was that the students could not understand and find out what type of tools they can use to defend and attack. As a result, several teams never attempted any intrusion, and there was no meaningful intrusion detection and defend. The lack of prior experience in Linux was emphasized in many feedback that the instructors received.

While the purpose of the sandbox is to promote free exploration in order to facilitate attacks and defends, the feedback highlighted typical students' hesitation in navigating a fully project-based learning environment and their preference for step-by-step instructions. Taking this into account, the instructor adjusted the course's structure to provide students with more gradual support.

5.1.2. Second and third semesters

In the subsequent semesters, the instructors enacted a number of changes to better coordinate the project activities. While the project still pushed students to research and explore the system and network security, there are more support structures in the materials. First, the entire game was divided into three complete stages, and there was one checkpoint for each stage to make sure everyone got the minimum requirement. Second, the instructor provided an initial reference list of intrusion and defend tools for the students. The list does not include installation, configuration, and usage documentation. As a result, if students decided to follow the list, they will need to perform research on how to use them. Furthermore, they can also use this list as a reference point for additional tools. Third, each group was to submit a game plan to illustrate the responsibilities for each team members and the time line for their implementation. Finally, the instructor presented and emphasized the teaching/learning approach for the class, which required two concurrent thrusts on concepts and technology. The lecture time focused on the conceptual parts, which illustrate how to solve problem. The project and homework were implementations that use certain technology to solve the real problem. Those technologies change all the time, and students are responsible to this knowledge on their own with some assistance such as an introduction lecture that provided during the lecture time. These changes resulted in an improved experience for both students and instructor in the following two semesters, although there are still students that ask for systematic instruction on how to do the lab.

Throughout the three semesters, the dedicated on-site local computers for this session enabled students to have a stable computing environment for the course's projects and assignments. However, working with this infrastructure encountered several administrative challenges. One challenge having to do with reserving the individual computers within the computer lab. To ensure no external interference, the instructor isolated the selected computers and placed a semester-long hold on these computers. This inadvertently reduced the number of laboratory computers available to other students. The setting of the room itself was also not conducive to team-based activities, as students struggled to find seating space around their team's computer. In the long term, this represented a scalability issue for increasing the instances of this session for the course. Second, even if other students do not use the reserved servers, they may use the Ethernet cables, which disable students in CSC 302 from remote login.

5.2 Session 2 of CSC 302

This session of CSC 302 was taught for the first time in Fall 2018. Learning from the experience in Session 1, the instructor for this session provided a series of supporting materials that aimed to introduce students to the Linux environment and C programming. Step-by-step instructions on setting up the Python's Anaconda environment, Jupyter server, and port forwarding were provided with the goal to ease students into working with. While students' feedback was positive, interesting technical challenges arose with the usage of the SEED VM.

The instructors observed that students in the class had different personal laptops, many of which were older models. As the uncompressed SEED VM required more than 12GB in size, several students struggled to find enough disk space to decompress and setup the VM. The age of the hard drive also impacted the integrity of the VM, and throughout the semester, the class encountered instances when the VM file corrupted, resulting in having to download and setup the computing environment again.

Even with SEED's modest hardware requirements (1 core and 1GB of memory), students experienced lag in working with the VM, particularly for the web security exercises.

After the first half of the semester working on a local VM, the introduction of the CloudLab-based project was well received by the students. As CloudLab provided a significantly better hardware performance, students were able to smoothly deploy and work on their individual VM regardless of their individual laptops' configurations. Near the end of the semester, the instructor also attempted to introduce two network security exercises regarding packet sniffing and spoofing. Without having to reserve a prior laboratory, the instructor deployed an online environment in CloudLab [17]. This environment allows the instructor to specify a number of VMs corresponding to the number of students, set up and equip the VMs with relevant software packages and source codes for the lectures, and deploy them on CloudLab. The entire process took approximately thirty minutes and was done prior to the start of the class.

5.3 CSC 497/583

Students came into CSC 497/583 near the end of the undergraduate career or at the graduate level. Therefore, issues related to prior experience working with Linux were rare. The instructor for CSC 497/583 observed several advantages and disadvantages in using local VM in this course.

5.3.1 Advantages

The use of local VM provided students with a safe, reliable, and consistent experiment environment. It was easy to use and students did not need to set up the initial Linux environment by themselves. This was especially important as many tools and libraries in the Linux environment are not easy to compile and install and require the student to have some level of debugging skills when facing with compiling error. Second, the local VM provided a GUI interface to the students, thus reducing the learning curve to the student who is not familiar with command line operation. Third, the local VM is cross-platform, and it supports both the Windows and Linux system, which is ideal for studying various binary file structure on the different operating system (e.g., PE and ELF format). Last but not least, the VM software provided snapshots which guarantee to revert the VM to an earlier state completely. This feature is extremely useful for the student who studied the malware analysis course.

5.3.2 Disadvantages

First, some students have faced compatibility issues when installing and using the local VM. Si found that some laptop does not enable the VT-D feature in their BIOS settings by default and therefore cannot boot up the VM image. Moreover, some laptop with NVidia graphics card and runs Linux Mint system needs to manually disable secure boot to install VM software (e.g., VirtualBox). Second, the Chromebook operating system does not support installing and running the local VM software. Students who own Chromebook have to use remote secure shell to connect to a remote Linux server and setting up everything by their own. These compatibility issues make student feel discourage or frustrated and may fall behind.

5. Conclusion and Future Work

In this work, we highlighted our experience using various computing infrastructures in cybersecurity courses. These infrastructures included on-site local network of computers, VMs installed on students' personal laptops, and computing environments deployed in the cloud. The former two presented various logistical, technical, and administrative challenges in ensuring a seamless and transparent hands-on learning environment for students. Early experience in the last approach using CloudLab, a national computing infrastructure, was positive and provided indications that cloud computing environments like CloudLab can provide significant support for teaching cybersecurity courses.

At the same time, using a cloud-based infrastructure is not without its hurdles. It places a burden on the instructor to install, configure, and deploy the environment, with the added caveat of having to learn how to work with a cloud provider. CloudLab is a research-based environment, and thus is not appropriate for scenarios where a dedicated infrastructure needs to be maintained throughout the semester. To address these challenges, the authors are pursuing future work in preparing cloud profile templates to help with automated installation and configuration and exploring additional resources, including the JetStream Cloud infrastructure inside XSEDE (Extreme Science and Engineering Discovery Environment) for provisioning sustained infrastructure.

4. Acknowledgements

The hands-on project in session 1 of CSC 302 borrows the game administration plan that was developed for the previous graduate security course (CPSC665) in Computer Science Department at Texas A&M University. Thanks Dr. Udo Pooch and Dr. Bin Lu for their generous help in the development of this game. They will be always remembered.

References:

[1] Paulsen, C., McDuffie, E., Newhouse, W., & Toth, P. (2012). NICE: Creating a Cybersecurity Workforce and Aware Public. *IEEE Security & Privacy, 10*(3), 76-79.

[2] Sobiesk, E., Blair, J., Conti, G., Lanham, M., & Taylor, H., Cyber education: a multi-level, multi-discipline approach. *Proc. 16th ACM Annual Conference on Information Technology Education*, Chicago, IL, 2015, 3-47.

[3] McDuffie, E. L., & Piotrowski, V. P., The future of cybersecurity education. *IEEE Computer, 47*(8), 2014, 67-69.
[4] DHS Task Force on CyberSkills, CyberSkills Task Force Report, *D.o.H. Security, Editor.* Washington, DC, 2012, 1-41.
[5] Conklin, W. A., Cline, R. E., & Roosa, T., Re-engineering cybersecurity education in the US: an analysis of the critical factors. *Proc. 47th IEEE International Conference on System Sciences (HICSS)*, Big Island, HI, 2014, 2006-2014.
[6] Weiss, R., Mache, J., & Nilsen, E., Top 10 hands-on cybersecurity exercises. *Journal of Computing Sciences in Colleges, 29*(1), 2013, 140-147.
[7] Brustoloni, J. C., Laboratory experiments for network security instruction. *Journal on Educational Resources in Computing (ACM Transactions on Computing Education)*, *6*(4), 2006, 5.
[8] Abler, R. T., Contis, D., Grizzard, J. B., & Owen, H. L., Georgia tech information security center hands-on network security laboratory. *IEEE Transactions on Education, 49*(1), 2006, 82-87.
[9] Bullers Jr, W. I., Burd, S., & Seazzu, A. F., Virtual machines-an idea whose time has returned: application to network, security, and database courses. *ACM SIGCSE Bulletin 38*(1), 2006, 102-106.
[10] Du, W., SEED: hands-on lab exercises for computer security education. *IEEE Security & Privacy, 9*(5), 2011, 70-73.
[11] Stoller, M. H. R. R. L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., & Lepreau, J., Large-scale virtualization in the emulab network testbed. *Procs USENIX Annual Technical Conference*, Boston, MA, 2008.
[12] Mirkovic, J., & Benzel, T., Teaching cybersecurity with DeterLab. *IEEE Security & Privacy, 10*(1), 2012, 73-76.
[13] Salah, K., Hammoud, M., & Zeadally, S. Teaching cybersecurity using the cloud. *IEEE Transactions on Learning Technologies, 8*(4), 2015, 383-392.
[14] Park, Y., Hu, H., Yuan, X., & Li, H. Enhancing Security Education Through Designing SDN Security Labs in CloudLab. *Procs 49th ACM Technical Symposium on Computer Science Education*, Baltimore, MD, 2018, 185-190.
[15] Ricci, R., Eide, E., & CloudLab Team, Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. *; login:: the magazine of USENIX & SAGE, 39*(6), 2014, 36-38.
[16] Watson, J., Virtualbox: bits and bytes masquerading as machines. *Linux Journal, 2008*(166), 2008, 1.
[17] Ngo, L., SEEDCloud. *https://github.com/linhbngo/SEEDCloud*, 2019.
[18] Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G.D., & Roskies, R. XSEDE: accelerating scientific discovery. *Computing in Science & Engineering, 16(*5), 2014, 62-74.

AN EXAMINATION OF THE ETHICS OF IMITATION GAMES

Brandon Packard , Jamie Phillips
Clarion University of Pennsylvania
bpackard@clarion.edu, jphillips@clarion.edu

ABSTRACT

The famous philosopher Friedrich Nietzsche once said, "The bad gains respect through imitation, the good loses it especially in art." It only takes a glance at the gaming market today to see that this holds true in the realm of game development as well. Almost all games draw inspiration from some source material, but games overall run the gambit from drawing a little inspiration to being straight up rip-offs of their source. In this paper, we will describe the 4 general categories of imitation games, and use theories of ethics to examine the morality of creating (and profiting from) each type of game.

1 Introduction

Herman Melville once said, "It is better to fail in originality, than to succeed in imitation" [1]. Why then, do so many people attempt to imitate success? It seems that wherever success goes, imitation follows. Whether because they are afraid of failing themselves or desire desperately to succeed, imitation is a tool used by many. With media like movies or art, copyright is usually highly enforced and prevents too much imitation from reaching broad audiences. When it comes to video games, however, imitation runs much more rampant. This is because it someone can create their own code and underlying structure but make the same or a very similar end product (making it harder to provide pure evidence of infringement), and many game creators don't have deep enough pockets to pursue legal cases against similar games. Note that although we will be exclusively discussing video games in this paper, the conclusions drawn can be applied to other areas in which copyright is tricky to enforce as well, such as music which is derivative of other music [2] or software that is derivative of other software.

The bigger and more successful a game is, the more imitations it spawns. Take, for example, the game Five Nights at Freddy's [3]. This game came out during a dry period in the horror genre, and took the gaming community by storm. Shortly afterwards, so many games that imitated the games mechanics and premise emerged that "Five Nights at Freddy's"-like games became its own horror subgenre. These games covered a large range of quality. There were a few that are widely regarded to be high quality (such as One Night at Flumptys and One Night at Flumpty's 2 [4]), and many that are regarded to be... less so (such as Five Nights at Thomas's [4]). Some have been so close they received copyright notices [5] while most are different enough to avoid this.

In this paper we will be discussing various categories of imitation game, specifically those which are a product, not a project. That is to say, games which are primarily intended to make money – not just games meant to showcase a concept or share something with the community. When things are a personal project, released for free, it is harder to fault them. When that line of project to product is crossed however, game design goes from a simple hobby to a business, and therefore should be held to a different set of standards. Specifically, we will discuss four different categories of imitation game, and comment upon the value and morality of each type.

2 Categorizing Imitation Games

For distinguishing between the categories of imitation games, we will be heavily relying on two aspects of the games: premise and mechanics. The premise of a game is essentially the overall idea of the game. For example, Five Nights at Freddy's is a horror game in which you are stuck in a room and have to somehow prevent enemies from getting into your room and killing you. Therefore, we would assert that any game which shares the overall idea that the player is immobile and has to prevent enemies from entering the room while stylized in the manner of a horror game shares the same premise. Mechanics, on the other hand, refer specifically to the actual gameplay. In Five Nights at Freddy's, there are a few core mechanics. You can check cameras to see where the enemies are at, light a light to see if they are right at the door, and close the doors to the room to prevent them from getting in. However, every one of these actions drains power, and if you run out of power you can no longer do any of these things (which is essentially a death sentence). Therefore, any game that allows you to see where the enemies are, allows you to close doors or otherwise prevent them from getting into your room, and does not allow the player to move around would be said to share the same or similar mechanics. Bearing these definitions in mind, we can classify imitation games into 4 broad categories:

- Inspired Game: The game draws inspiration from the

original but is a unique game with its own premise and mechanics.

- Inspired Imitation: The game was clearly inspired by the original and shares many of the same mechanics/the same general premise. However, the game has unique or creative elements that extend beyond that inspiration, and it can be considered its own entity.
- Pure Imitation: The game is basically just a reskinned version of the original it is trying to imitate. The mechanics and premise are basically exactly the same. There is very little, if anything, new or creative about the game.
- Namesake Imitation: The game is a completely unique game, nothing like the original, only using the name of the original or images/assets ripped from the original to generate hype and get people to buy/download the game.

It should be noted that the line between the second and third type, inspired imitations and pure imitations, can sometimes be a bit blurred. The main difference between the two is how "new" or "different" the game is from its inspiration, which can be subjective in some cases (what if it shares exactly half the mechanics and the same premise? We would likely think that it would depend on the exact situation in this case!). For the remainder of this paper we will be using inspired imitation to refer to those games which are clearly new and different from their source material, and pure imitation to describe those games that are clearly *not* new or different from their source material.

3 Value of Imitation Games

The first category certainly has value to the gaming community. Drawing inspiration from something in order to create something new an exciting is not only widely accepted in society, but it is also the way that most new things are created. We believe it is fairly safe to say that making inspired imitations is a morally acceptable process under almost any theory (although we will still analysis it under several!).

The next category also has a clear value to the gaming community. Although this type of game shares the same overall mechanics and often the same or a very similar premise, it builds upon it by adding new or unique elements. By doing so, it adds something new to the gaming community, and therefore can be said to have value (we would assert that even if the new elements are not particularly good, they still add value because then others can be inspired by/build upon those elements, and eventually something good will be born from that process – not every single brick in a wall is great in and of itself, but they add up to create something significant over time).

Next, we have the pure imitation games. These games change very little about the original game. They have the same premise (for example, you are trapped in area X and Y is chasing you down) and the same gameplay mechanics as the original. They are basically just remakes of the original game but with slightly different themes. The logic behind this is that if the original is making a ton of money, then a game that is almost the exact game but retextured or rethemed could also make some money. Interestingly enough, this does tend to be the case. The exact reasoning for this comes down to the mentality of how fandoms work, which is outside of the scope of this paper, but suffice it to say that when a game gathers a large enough fanbase, they tend to build hype off of each other and latch onto anything related to that game that they can. We can confirm that this is true via personal experience. As one of the authors is a huge Pikmin© fan, if something were to come out that was a Pikmin themed ripoff of another game, they would be very tempted to buy it just because it is Pikmin.

Finally, we have the most egregious type of imitation game – namesake imitations. We consider these the worst type because they actually have very little or nothing to do with the original they are trying to imitate. Rather, they are trying to use the hype around the original in order to make money. There are two major ways to do this: using the name and using the images. Sometimes game developers will name their game after a popular game, when in reality it has nothing to do with it. For example, Cuphead Plane [6] is a game that uses the name Cuphead, a very popular game from 2017, but in reality is completely unrelated to it. The gameplay is different, the character Cuphead isn't even in the game, and it is incredibly obvious that they just named it that to build up the hype and get people to download it (and then show them ads for money). Sometimes developers also use sprites/characters in the screenshots for the game for the same reason. In particularly egregious cases, developers will actually use snapshots of the original game as the screenshots for their game, which looks or plays nothing like the original (Such as Fort Troops, which uses screenshots from the very popular game Fortnite [7])!

4 The Morality of Inspired Games

As previously mentioned, Inspired Games definitely add value to the community. They are of varying quality, but overall this type of game definitely tends to hold value with gamers and the gaming community. Furthermore, these games do not participate in any level of deception, generally speaking. In fact, the vast majority of game designers today don't just pull game ideas out of a void. Often inspiration is drawn from other media including video games (and often that media has been inspired from other media, and so on). For example, Destiny draws inspiration from Borderlands [8], which in turn was artistically inspired (not from the start, but in the end) by CodeHunters [9], which was in turn inspired by graphic-novelists such as Bilal or Moebius [10].

As such, it is easy to see that it is at least morally permissible to create this kind of game. Something of value is put into the community, and everyone benefits as a result, which easily

satisfies Utilitarianism [11] – A theory which states that actions which result in the highest overall happiness are morally permissible. In fact, we would say that this type of imitation game provides the most overall utility to the community as a whole, as it leads to truly new creations.

Kant's Categorical Imperative [12] states that humans must always be treated as an end, never as a means. For example, Kant would condemn employers who treat their employees less and less like humans in order to make more and more money; at that point they are using humans as a means to their own ends instead of treating them as people. Inspired games do not use humans as a means. They are forthcoming and honest about what they are, and they provide value back to the humans whilst making the developers money. Therefore, Kant would judge the process of making Inspired Games as morally permissible.

Garrett provides a framework for business ethics that more or less combines these aspects, the Proportionality Framework [13]. This framework states that business decisions can be split into three components: intentions, means, and ends. If a decision is lacking in one aspect but is strong in the other two aspects, (say it has good intentions and good means but the ends fall flat), it can still be considered a moral action overall. For example, in the medical field, if a doctor performs a surgery with the intentions of saving a patient's life, and he/she performs the surgery without error but the patient still passes away due to factors beyond his control, we would not condemn his actions as immoral just because the ends turned out bad. Since Inspired Games have good intentions (using something old as inspiration to create something new), good means (creating a game without purposefully copied mechanics) and varying ends (from rough products that don't really add much to masterpieces that are of immense value), the Proportionality Framework would definitely accept this type of game as morally permissible.

Laczniak [14] gives three ethical frameworks specifically targeted at businesses: the "professional ethic", the "TV test", and the Prima Facie Duties framework. In the former, business actions are considered morally permissible only if a disinterested panel of professional colleagues would deem them to be moral. Although we have never technically published a game for sale, the authors have created a couple of games and have worked with helping students learn game design, and we would definitely define Inspired Games to be moral – they aren't a blatant rip-off, they have effort put into them, and they add some value to the community. We would expect a panel of peers to feel similarly. The second, the TV test, states that an action is only morally impermissible if you would feel comfortable explaining to a national TV audience why you chose that action. Not only do we think developers of these games wouldn't feel ashamed to explain on national TV that they took inspiration from X and made Y, but we would also argue that they shouldn't. Taking something as inspiration and using it to create something new is what art and game design are all about. Inspiration can come from anywhere, such how the game Pikmin was inspired by watching ants carry leaves in a garden [15], but artists and game developers also often draw inspiration from other artists and video games (this can easily be confirmed with a quick glance at any gaming marketplace).

4.1 Prima Facie Duties

Laczniak [14] then goes on to describe what he believes to be more comprehensive frameworks for business ethics, such as the Prima Facie Duties framework. This framework, based on the work of W.D. Ross, is based on the notion that there are several "at first sight" duties which are self-evident and constitute moral obligations under most circumstances. There are 8 duties Ross puts forth, although they can be condensed into 7 [16]: Beneficence, harm-prevention (which can be seen as part of beneficence), non-maleficence, self-improvement, gratitude, reparation, fidelity, and justice. The first states that one should do good, if given the opportunity, and the second states that one should not harm others. Duties of self-improvement state that actions should be taken which promote one's own self interest. The duty of gratitude says that, when possible, you should show benefactions to those who show benefactions to you, and the duty of reparation says that you need to make up for the injuries you have done for others. The duty of fidelity says you need to keep your promises and contracts and not engage in deception. Finally, the virtue of justice requires you to act so as distribute benefits and burdens fairly (note that this seems to dis-include being a parasite, or soaking up benefits for yourself while lauding burdens upon others). These duties all stem from moral intuition – that is to say, we can just naturally *see* that non-maleficence is a good rule to follow. Similarly, when duties conflict (as they often will), moral intuition tells us what the priorities are. For example, we can just *see* that non-maleficence takes precedence over beneficence in the general case.

This framework is typically used for medical or business decisions. For example, say you were going to bring a new medicine to market. Using the first three duties for simplicity (in reality, all seven duties would have to be considered), the questions you should ask yourself are (1) is this medicine actually going to help anyone, (2) is this medicine going to hurt anyone, and (3) is this medicine a step in improving your own medicine-creating skills? If all three duties are met, then bringing the medicine to market is a moral action. If not all of the duties are met, then you need to look at the priorities of the duties to determine if the action is moral (for example, other factors being equal, non-maleficence normally overrides other duties). This can be applied to business actions as well. If you are the CEO of a large company and are going to roll out a new policy, you have to consider (1) if it is improving yourself and the company, (2) if it is going to benefit others, and (3) if it is going to harm others.

Of the seven duties, the creation of Inspired Games meets every single one. The first, duties of beneficence, states that actions should improve the intelligence, virtue, or happiness of others. As Inspired Games are creations which hold value in

the community and are viewed overall very positively in the community (and in fact are seen as the way that most good new games get created), it is safe to say that they meet this duty. The second is duties of self-improvement, stating that actions should be taken which increase our *own* virtue, intelligence, or happiness. As the creators of these games are taking an idea and pushing themselves to create something new and interesting from it, Inspired Games also meet this duty. Thirdly, the duty of non-maleficence states that actions should be taken which do not injure others. Being that these games do not hurt the community, and usually help it, it meets this duty as well and creating these games would be considered morally justified under this framework as well. Gratitude is followed because Inspired Game developers provide value to those who purchase their games, and fidelity is met because these game developers are very honest and upfront about what their games are. Justice is also upheld, because Inspired Game developers aren't leeching off of the community – they are putting out something of value and getting rewarded for it. Finally, the duty of reparations is mostly irrelevant for this category of game. This is because, for the most part, these games do not harm the community, so they have nothing to make up for. The main factor of harm that could be caused is from a game being incomplete or buggy, and in the general case these developers will take care (reparate for) these issues in order to make their game the best it can be. Therefore, since Inspired Games meet all seven duties, there can be no question that the Prima Facie Duties framework would find them to be morally permissible as well.

Finally, let us examine these games under a lens that cares only about one's own pleasure – that of Hedonism [17; 18]. Hedonism, much like utilitarianism, is about maximizing pleasure. However, Hedonism cares about maximizing one's own pleasure, and asserts that happiness is the only "true" pleasure. As such, creating a game is not intrinsically valuable. In fact, even the money garnered from a successful game is not considered intrinsically valuable. In terms of Hedonism, the only value that comes from creating a game is the happiness that you get as the developer. Does making Inspired Games bring happiness to game developers? We would argue that it almost certainly does. Inspired games are games that developers tend to feel proud about and that they take an ownership to, as opposed to being something that is merely churned out for money. In addition to any money they may make, which will buy them some happiness, there is also a sense of accomplishment with a finished game that one has spend so much time and effort on. We have experienced this feeling myself on my own creations, even though we have never actually sold any of them, and we can tell you that it is indeed a happy feeling. Therefore, and perhaps unsurprisingly, Hedonism would also justify the creation of Inspired Games as an ethically permissible act.

5 Morality of Inspired Imitations

As previously discussed, this category of games also has some value to the community. It tends to be a smaller, more incremental type of value, but value nonetheless (and no deceptive practices that might hurt the community!). Progress isn't always made one huge leap at a time, it's often made in much smaller steps that add up (for example, cars have been making incremental changes for over 100 years, but if you compare the car of today to the car of a century ago, the difference is immense). This is true of video games as well. Someone may make a inspired imitation that builds on the original but adds a couple unique features. Someone else may then like one of those features and make a whole game focused upon it, or a new game inspired by that one aspect. For example, Warcraft is a real-time RPG game that was heavily inspired by (some would say ripped off of) WarHammer 40k [19]. Warcraft has a robust map editor allowing for changing maps around to the level of detail that you could significantly change how the game plays. Steel Crate Games took that idea of changing the game and basically built their own game, Keep Talking and no one Explodes, around that feature [20]. Additionally, Warcraft's expansion into multiple games including the wildly popular World of Warcraft (which had 5.5 million subscribers in 2015, 10 years after it came out, and peaked around 12 million subscribers) shows that even though Warcraft could be called an inspired game, it started a chain of events which has drastically affected the gaming community. As such, it is also easy to see that it is at least morally permissible to create this kind of game. Something of value is put into the community, and everyone benefits as a result, satisfying Utilitarianism [11]. Strictly speaking, this type of game has a less significant impact than Inspired Games on average. Therefore, according to strict utilitarianism, it would *not* be morally permissible, as there is an alternative which has a higher overall benefit. However, these games do satisfy the general utilitarianism principle that actions are moral if they provide an overall net benefit, so these games could be considered morally permissible by utilitarianism in general.

Furthermore, these games are not using humans as means to an end. These games add value to the gaming community and people tend to enjoy and appreciate them overall. Because of this, it cannot be said that they are using humans purely as a means at any point. Sure, the point of a game is to make money, and to some extent they are using people to do that. However, they are making money by creating games that people enjoy – this means that they are using humans as a means but doing so bearing in mind that they are humans, and treating those humans as an end (providing them entertainment). As such, they are not using humans as *purely* a means to an end, Kant's Categorical Imperative [12] would surely find this type of game to be morally permissible.

Given these previous deductions, Garrett's Proportionality Framework [13] naturally follows as morally permissible. Inspired Imitations have good ends (adding value to and advancing the community), good means (doing business in a way that doesn't harm anyone and usually ends up benefiting everyone), and good intentions (making money, but doing so by releasing products worth their price tag that will help advance the community and provide enjoyment for those in it

as well) – the result being a process of creating games that would certainly be considered morally permissible under the Proportionality Framework.

Next, we will take a look at Inspired Imitations according to Laczniak [14]. The first test that we will examine is the "TV Test". Most developers of these games would not be ashamed to stand in front of a national TV audience and talk about their games, since they finished products and do add some value to the gaming community. However, the part that developers may not want to talk about is just how "inspired" their game is by the inspiration. However, we believe this is more of a legal issue than a moral issue. If they admit that they were too heavily inspired, it could cause them legal trouble (whether warranted or not). We have no doubt that, if not for the looming threat of legal action, they would be happy to talk about their inspiration and exactly how they have modified or added onto it to create this new and interesting product. The second test, the "professional ethic", would have similar results. If a panel of game development companies were to be assembled, they would have 1 of 3 viewpoints on how heavily the game uses its inspiration. The first would be that its perfectly fine, this sort of heavy inspiration is just how the field works. The second would say that drawing so heavily on their inspiration isn't desirable and that they should really do it less, but since they add something new or interesting and it's not just a blatant copy, it's fine. The last would say that these games are too close to the original games and condemn the actions of their developers. However, this last opinion would definitely be in the minority. As we have already discussed, many, many games are made by drawing inspiration from previous things. There just aren't a lot of truly "unique" games that don't draw inspiration from *somewhere*. As such, a panel of professional game developers is likely to have a majority of those who sympathize with the use of inspiration and would have one of the first two opinions (We are making the reasonable assumption here that these game developers would hold others to the same standards that they hold themselves), and therefore we believe Inspired Imitations would pass the "professional ethic" test.

As with Inspired Games, Inspired Imitations satisfy all seven duties of the Prima Facie Duties framework. In fact, they satisfy the duties for all of the same reasons. They add value to the community and do not harm it in any way, and although they heavily draw from their source of inspiration, they still improve upon it (and therefore improve their own skills) by adding new or different features. Inspired Imitation developers give value back to those who purchase their games, and would be likely to fix any errors or incompleteness in their games. They are upfront and honest about what their games are, and they spread out benefits in the community by distributing something of value. Therefore, since the process of creating Inspired Imitations meets all relevant duties, it can be said that the Prima Facie Duty framework would consider it morally permissible as well.

The final theory from which to consider Inspired Imitations is that of Hedonism. As previously discussed, hedonism states that things which are intrinsically good to oneself are good, and you should take actions to maximize these intrinsic goods for yourself. The true intrinsic good is happiness (possessions provide happiness, money buys things which then provide happiness, and so on). Do Inspired Imitations increase the happiness of their developers? We would say yes. Not only do they make money, which allows the game developers to make a living or extra money to buy things which will make them happy, but they do it in a "good" way. What we mean by this is that most Inspired Imitation game makers *enjoy* creating games. So in addition to the happiness that the product will bring, the process also brings them joy – hedonism would see value in the process itself for these games! How do we know that these developers enjoy the game creation process? Because they put time and effort into it. You can make money by cobbling together terrible games and selling them (a process known as asset flipping [21]), but Inspired Imitation game developers are putting time and effort into their project. They aren't seeing it as a means to the end of making money, but rather as something they enjoy doing (making money from games is often what allows them the time to be able to create games, rather than the reason they make games). As such, both the product *and* the process bring the developers happiness, making the creation of Inspired Imitations a morally permissible process according to Hedonism.

6 Morality of Pure Imitation Games

Pure Imitation games are those which are basically "reskins" of another game. They change very little about the game, having the same gameplay mechanics but different graphics or themes. These games really don't offer much to the gaming community in terms of value. Granted, there is something to be said about being able to play your favorite characters from one game in another game, but these games don't do anything new or unique, and don't really advance the gaming community in any significant way. They also sometimes participate in the deception of their customers, although that depends on the specific instance. However, it can be argued that these games can sometimes detract from the community. Take the revolutionary game Five Night's at Freddy's (FNAF). As previously discussed, FNAF took the horror gaming community by storm. Its immense popularity and fervid fanbase led to some Inspired Imitations, and many, many Pure Imitations. So many Pure Imitations, in fact, that the Steam marketplace was flooded with them. This is where the harm from this type of game can be seen. It's not that these games are harmful in and of themselves, but rather because they are relatively low effort and are often mass produced, they can have such a large presence that other games don't get seen. Inspired Games and Inspired Imitations that are new and unique sometimes get missed completely because of the influx of Pure Imitations. Since Pure Imitations don't really add anything to the community, and they can block some of the games that actually do, they can be harmful to the community in some cases.

As with the other categories of games, we shall begin by looking at Utilitarianism [11]. Can these games be said to increase

overall happiness? Well, they arguably increase the happiness of the game developers themselves, as they are earning money. It is debatable whether creating these games makes them happy or not, but in the end, it does not matter. As we have said before, these games indirectly hurt the community as a whole, both people who are looking for new and unique games and the developers of these new and unique games definitely suffer due to the flood of Pure Imitations. Therefore, since the developers are really the only people who even may get some happiness, utilitarianism would definitely condemn the practice of making Pure Imitation games.

Next up is Kant's Categorical Imperative [12]. Whether or not Pure Imitation games would be morally acceptable based on this theory is a much harder question, because it is based on the intentions of the developers. If the developers are intending to just make this games for the sake of making money, then they are using humans as an end – they aren't really considering the players or the community, but rather how they can make more money. On the other hand, imagine if the developers are making this game because, say, they love FNAF and they love Sonic the Hedgehog so they want to combine the two. Since they are hoping that others are able to get some enjoyment out of it as well, then they are considering other people as people and not just a means. However, in this case it would seem odd that they are charging money for the game if they just want others to be able to enjoy it. Many developers release short experiences or games just for the sake of the community, so charging money for it indicates that, at least to some extent, the developers intent is to make money from it. Logically, this also indicates that they are using gamers as a means to some extent. However, are they being used *purely* as a means? In this case, it depends on the marketting. If the game is honest and upfront in the description about what it is ("Play the Sonic game you know and love, but this time as Mario!") then gamers are able to provide informed consent for what they are buying. If the game tries to deceive the player by claiming it is something bigger or more unique than it is, then it is not really considering the player, only the money being made. In that case, they would definitely be using the gamer as purely a means to an end. Therefore, Kant's Categorical Imperative would have to be viewed on a case-by-case basis, as it would certainly permit some Pure Imitation game developers and condemn others.

Garrett's Proportionality Framework [13] is another case that is not so cut and dry. The ends of Pure Imitation games are bad overall; they don't really add anything to the community and tend to drown out those games which do. As we discussed previously, the means of the developers vary based on the individual game development teams. Similarly, the intentions of the developers vary: are the making Pure Imitation games purely for money (a bad intention), or so that others can enjoy their work and just charging money to make up for their time (a much better intention). However, the idea of this framework is that if two of the factors are good enough, they can make up for a 3rd factor that is bad. If one of the factors (ends) is bad, and the other two can vary between somewhat good and very bad, it can be safely concluded that overall there is not enough goodness overall to make a morally good action, even when considering all three factors.

The next set of ethical lenses from which to examine Pure Imitations come from Laczniak [14]. Just as with the Categorical Imperative, whether or not the TV Test would find Pure Imitations morally acceptable depends purely on the intentions of the creators. If the creators are developing the game as a labor of love, then they would surely have no problem saying that on national TV and explaining that they charge money in order to be able to keep producing games. If the creators are just churning out games to make money however, one would imagine that if they have any sense of shame they would feel quite uncomfortable admitting that to a national TV audience. The professional ethic test would go similarly. Granted, there are some game developers who would definitely get caught up on the fact that Pure Imitations in general flood online gaming platforms, but for the most part game developers would look at the studio's intentions to see whether their actions are moral. This is because creating games is, for most, a labor of love. There are very few ways to make "a quick buck" in the world of game design (Asset Flips [21] being a key exception), so most game design is a labor of love. A panel of randomly selected Game Development professionals would almost certainly consist of a majority of people who love what they do, and in turn would judge Pure Imitation developers based on whether those developers love what *they* do.

In terms of Prima Facie Duties, beneficence and non-maleficence are tough to determine in this case. Some Pure Imitation games make some people happy, and some make no one happy. Some people don't like any Pure Imitations, and there are likely people who like all Pure Imitations. We would say overall that the duty of beneficence is kind of a wash – these games make some people happy but make some people frustrated. The same is true for non-maleficence. These games help some people (letting them play a character they like in a world they like, when this would normally never happen) but also hurt some people (flooding gaming marketplaces and reducing visibility of other games). A big issue this framework would have with these games, however, is the duty of self-improvement. With one exception, those who make Pure Imitation games are not really improving upon themselves or their skills. They are re-skinning games and not adding anything new or creative – not really putting enough effort into the craft to actually improve their skills. The one exception to this would be those who make a Pure Imitation as their first ever game – in that case they *are* learning new skills and improving themselves. However, it is commonly accepted that if you are that new to game design, you should be releasing your games as a project (i.e. free) instead of a paid-for-product – and that is the usually (but definitely not always) followed rule. Since this paper only deals with categorizations of paid-for games, this exception is small enough that overall game developers who make Pure Imitations (especially if they are making multiple of them) are definitely violating the duty of self-improvement. Fidelity, justice, and gratitude also rely on the intentions of the developers. If the

developers are being honest and giving back some value for the money gamers spend on the games, all 3 are clearly satisfied. However, if the developers are clearly hoping to make a profit and attempt to harm and deceive the company in order to do so, then all 3 are clearly violated. Reparation is a tough sell. Even though these games are adding some value to the community, it doesn't seem to make up for the harm they cause. Flooding the market and drowning out games which are much more valuable cannot be made up for by the small amount of value that these games add to the overall community. Overall, it seems intuitively clear that if the developers' intentions are pure, then this framework would deem their actions to be morally permissible; if they are just in it for the money then the Prima Facie Duties framework would deem them impermissible.

Finally, we shall examine the process of creating Pure Imitations via the lens of Hedonism. For those developers whose intention is to make a labor of love, certainly this would be creating their own happiness. Additionally, that is what they are spending their time on, to the point where they actually have to charge money to be able to afford to create these games. That shows dedication and love, and it can easily be said that this would give them some sort of intrinsic happiness. But what about if the developers' intentions are purely to make money? You could argue that since money can buy pleasure (the one true intrinsic good), maximizing profits would be maximizing pleasure, and therefore be morally acceptable. However, life in the real world isn't quite so simple. There has been a lot of debate over whether money buys happiness. Graham [22] attempts to examine the complexities underlying this debate, and has found that the issue is a lot more complicated than most studies will say. Her take-home point is that there are many factors in the issue and although making more money helps, it will not on its own increase happiness. Therefore, you have a person (or people) toiling away at something they don't like doing to earn money that may also not make them happy. This definitely seems like a situation in which one isn't maximizing their own happiness, or even increasing it in many cases. Therefore, as with many of the previous theories and frameworks, Hedonism would commend or condemn the creators of Pure Imitations based on their intentions and goals.

7 Morality of Namesake Imitations

The final category of imitation game is that which we have dubbed "Namesake Imitations" since they imitate the original in name/identity only. As previously discussed, this type of game actively uses the name or images of a popular game purposely in order to get people to download/buy the game – that is to say, developers of Namesake Imitations actively and purposely rely on deception in order to make money. Even if the game is free, once it has been downloaded, they can try to push in-app purchases or just flood the user with ads to generate revenue (making it still qualify as a product not a project). More examples of this are the highly numerous amount of games using Mario's name (or very close to it, like "Hario") or likeness on google play in order to sell their own poorly made games [23]. Another example is "The Legend of Zelda Twilight Sword", which used models and the name of "The Legend of Zelda Twilight Princess" in order to try and raise money on Kickstarter, or "Pocket Monster Saga" which uses the original name of Pokemon along with models and many other story elements to push their own game [23]. These games have very little value to the community, as they basically only exist to make money off of misinformation/deception and prey on gamers.

It doesn't take much to see that Utilitarianism would heavily condemn the actions of Namesake Imitation game developers. As previously discussed, it is debatable whether having more money alone makes one happier. Let us assume for a moment that it does. In this case, the developers would be made happier by their creations, but would anyone else be? This certainly does not seem to be the case. The vast majority of gamers who buy/download these games will be sorely disappointed when the game is not the game they are expecting. These games also hurt the community as a whole, by making it harder to find the real game or new and interesting games by overcrowding the marketplaces. As such, overall happiness is not increased – much the opposite, and therefore Utilitarianism would find the actions of Namesake Imitation game developers to be morally reprehensible.

Likewise, Kant would also condemn their actions. Recall that Kant's Categorical Imperative states, among other things, that humans must always be considered as ends, never purely as a means. It is fairly obvious that the creators of Namesake Imitations are *not* treating humans as an end. The only end these developers have in mind is making more money for their wallets. Not only that, but they are directly using other people in order to accomplish their goal, by actively trying to trick them into either buying or downloading the game! As such, it is exceedingly clear that Namesake Imitation developers are using humans purely as means, with no consideration for them as ends at all, which is the exact opposite of what Kant says should be the case! Therefore, it is beyond reasonable to conclude that, were Kant alive today, he would undoubtedly condemn the actions of the developers of this category of game.

Given that both Utilitarianism and Kant's Categorical Imperative heavily condemn Namesake Imitations, it is basically a forgone conclusion that Garret's Proportionality Framework will do so as well. Since both the means (actively attempting to dupe people in order to make money) and the ends (a marketplace flooded with games which are harmful to the community both individually and as a whole) are morally impermissible, Namesake Imitations are morally impermissible regardless of their intentions. However, just for the sake of completeness, let us touch upon their intentions. The intentions of these developers are purely to make money without any regards to anyone else. As such, not only do the three factors together not add up to moral permissibility, but in fact each individual factor is morally impermissible under this framework!

We now move on to the "TV Test", which is an interesting problem in this case. Personally, we would be more than ashamed to stand in front of a national TV audience and explain that we make games which exploit people in order to make money and we have a feeling that you, as the reader, would feel similarly. However, we would also never make these types of games that exploit people in the first place. If game developers are so bold as to actively try and fool people into spending money, that already shows a certain lack of shame in their actions. Therefore, would they really feel shame to stand in front of a national TV audience and state what they do? They certainly wouldn't want to do so, but that could just be because it could potentially damage their business and harm their money making, not through any sense of actual shame. We would argue that whether or not they would feel comfortable explaining their actions on national TV basically depends on the individual developer. It is reasonable to say that some would feel ashamed and not comfortable, because most people would, but it is also reasonable to say that some would not feel ashamed and be totally comfortable, given their line of work. As such, since the TV test asks whether the person performing the action would be comfortable, whether or not it would find the actions of Namesake Imitation developers depends on the individual developers. For the "Professional Ethic" test, it is first important to understand that Namesake Imitation game developers are very much in the minority. The vast majority of game developers do not create games that are purely intended to dupe people into making them money. We have said before that these games can crowd out other games in the marketplace, but that is mostly because these games are relatively low effort, so the developers who do make them are able to make a relatively high number of them. Namesake Imitations and Pure Imitations are around the same amount of work, and developer Digital Homicide, a two-man indie game development team, released 21 games that were either asset flips or Pure Imitations of asset flips in 2.5 years [24; 25]. This is a rate of less than 1.5 months per game, which any respectable game developer will tell you raises huge red flags for the potential quality of the game as well as the intentions of the developers. If that weren't bad enough, an asset flipper known as Silicon Echo Studios developed and placed 86 games on steam in just 2 months, accounting for at least 10% of all the games published to Steam in that time period [26]. For reference that's more than one game being developed *PER DAY*. On the other hand, take the very popular Call of Duty series. This series is made by a AAA company with a ton of manpower, regarded as a mechanically sound game but nothing unique or special, and they only average about one new game a *YEAR* [27]. As previously discussed, most game developers actively enjoy making games, and would be unhappy with the idea of just churning out games for money. As such, a panel of random game developers would almost certainly contain a majority of people who would look down on Namesake Imitations and deem them to be morally impermissible.

How about the Prima Facie Duties framework? Well, we have already discussed that Namesake Imitations do not add value to the community, and actually detract from it by trying to trick people into spending money by imitating the name or images of popular games. As such, Namesake Imitations definitely violate the duties of beneficence and non-maleficence. Furthermore, it can also be said that they violate the duty of self-improvement. Developers who create these games do not try to do anything unique or different. If a game is good, then it wouldn't need to rely on the cheap tricks that Namesake Imitations do. The fact that developers have to (and are willing to!) rely on these tricks in order to garner revenue clearly shows that they have no desire to improve upon themselves. Making Namesake Imitations is all about making money, rather than making good games or improving upon themselves. Gratitude and reparation are failed, because Namesake Imitations do not give any value back to those who purchase them, and they do not in any way atone for the harm they do to the community. Namesake Imitations violate the duty of Fidelity because they actively try to deceive gamers into buying them, and they violate the duty of justice by "leeching" off the community (they suck up benefits while not giving them out). As such, Namesake Imitations fail to uphold all seven duties, and we do not even need to look at which duty takes priority to see that they would be condemned.

The final lens through which we will examine these games, as with the previous types of games, is that of hedonism. Hedonism is all about maximizing one's own pleasure, with the ultimate pleasure being happiness. As we have well established, making Namesake Imitations is all about trying to make money for yourself, and the developers who do so almost certainly do not enjoy the process. We have also previously discussed that making money does not in itself increase happiness. In fact, toiling away doing something you do not like just for the sake of making money is something that we intrinsically see as bad (think of how often TV shows or movies end with a character moving to a new job that makes less money but they like more, and makes them happier as a result). This is something that one of the authors can personally relate to, as there could be considerably more money made with a different type of job – but the level of happiness in life would certainly not be as high. In fact, it's likely you or someone you know has had similar experiences! As such, although they are certainly making money, and may even believe that this money is bringing them happiness, we can conclude that they are probably not truly happy. If they found something that they liked to do more, and made a living off of that, they would certainly be happier. As such, even hedonism would find the actions of Namesake Imitation developers to be morally impermissible.

8 Objections

8.1 Flooding the Market as a Boon

One of our main negatives about Pure Imitation games is that they flood the market. However, flooding the market can also

be good in some cases. A flooded market can give customers options and force more competitive prices. This only applies, however, when the market is being flooded with products of approximately equal quality. As Pure Imitations are typically lower than average quality, flooding the market with these brings the average quality of the market down, in addition to making high-quality products more difficult to find.

8.2 Tiers of Theft

In this work, we have stated that Namesake Imitations are the worst type of imitation game. But wouldn't the theft of gameplay mechanics be more greivious than simply stealing a name? After all, it seems like a much bigger part of the game that is being imitated. Although there is definitely logic to this in the general case, we assert that in this specific situation stealing the name is much worse. Why is this the case? Well those who are stealing the mechanics have a variety of reasons: To make something cool, to make money, to try to create their personal vision, and so on. However, those who are simply stealing the name or image of games are doing so with one purpose in mind: to deceive customers. The only reason that Namesake Imitations are created is to deceive people into spending their hard earned money on the game. When looking at it this way, Namesake Imitations are definitely a worse offense, especially since their goal is to use humans purely as an ends to lining their own pockets.

9 Conclusions

In this paper we discussed four different types of imitation games:

- Inspired Games – Games which clearly draw inspiration from a source, but are still unique, having their own premise and gameplay mechanics (such as Borderlands).
- Inspired Imitations – Games which imitate their base inspiration, sharing a premise and or mechanics with it, but also add some new or unique features (such as Five Night's as Flumpty's).
- Pure Imitations – Games which heavily imitate their base inspiration. They do not do anything new or interesting and have the same premise and mechanics as the original. Oftentimes these are just "reskinned" versions of the original (for example, taking Sonic the Hedgehog 2 and leaving it the exact same game except swapping Sonic's sprite for Mario's).
- Namesake Imitations – Games which imitate their base material in name or image only. These games attempt to make potential buyers believe they are the original game or associated with the original game by using either the name of their source game or images made to look like their source game (or in the case of Fort Troops, actual images from the source game!).

These 4 types of games are, not coincidentally, listed both in order of their value to gamers and within the gaming community as a whole. For example, Inspired Games add a lot of value to the community as they have interesting features and mechanics. These games, together with Inspired Imitations, comprise many games, due to the fact that game developers almost always draw some inspiration from somewhere. Inspired Imitations also add value to the community, albeit it not quite as much, because although they imitate the source game they still add new or interesting features. Pure Imitations add very little to the community. They offer some value for those who have obscure desires (such as being able to play as Mario in a Sonic game), but not for the community overall. Furthermore, having a fair number of these in gaming marketplaces such as Steam can reduce the visibility of the first two types of games. Finally, Namesake Imitations are quite detrimental to the community. In addition to the aforementioned blocking of games which do add value, these games can be considered predatory games. They are purposely trying to fool gamers into thinking these games are something that they are not, and therefore trick them into either buying the game, in-app purchases, or just bombarding them with ads in order to make money.

In what is also perhaps not surprising, these games also turn out to be listed in order of overall morality, according to the frameworks and theories that we have examined. This seems to indicate that, generally speaking, the more unique a game is, the more moral it is to create!. The first two types of games, Inspired Games and Inspired Imitations, were found morally permissible through each of the lenses examined here. Pure imitations were more conditional. Some theories rejected them as definitely morally impermissible but many theories could possibly declare them either way, depending on the intentions of the individual game developers on a case-by-case basis (whether they are making these games out of genuine enjoyment, or just because they like money). Finally, Namesake Imitations were soundly rejected by all lenses are morally reprehensible. As such, we can conclude that the sale of the first two categories is fine, the third is sometimes okay, but the sale of Namesake Imitations is morally impermissible and should be condemned. Charles Caleb Colton said that imitation is the sincerest form of flattery, but game developers who employ too much imitation fall morally flat.

References

[1] H. Melville, *Hawthorne and his Mosses* (Charles River Editors via PublishDrive, 2018).

[2] A. Hermann, Beyond 'blurred lines': How forensic musicology is altering pop's future, https://www.rollingstone.com/music/music-features/beyond-blurred-lines-how-forensic-musicology-is-altering-pops-future-204986/, 2018, accessed: 2018-11-06.

[3] S. Cawthon, Five nights at freddy's, https://store.steampowered.com/app/319510/Five_Nights_at_Freddys/,

2014, accessed: 2018-10-11.

[4] P. Klepek, One night at flumpty's, https://tvtropes.org/pmwiki/pmwiki.php/Videogame/OneNightAtFlumptys, 2016, accessed: 2018-09-20.

[5] B. D, Case animatronics, five nights at freddy's clone gets hit with dmca claim, https://blogjob.com/oneangrygamer/2015/11/case-animatronics-five-nights-at-freddys-clone-gets-hit-with-dmca-claim/, 2015, accessed: 2018-09-20.

[6] I. Binyamina, Cuphead plane, https://play.google.com/store/apps/details?id=com.plane.run, 2017, accessed: 2018-09-20.

[7] Moonrise, Fort troops, https://apkpure.com/fort-troops/com.forttroops, 2018, accessed: 2018-09-11.

[8] D. Tach, How destiny builds on inspiration from borderlands, https://www.polygon.com/2013/8/1/4578700/destiny-inspiration-borderlands, 2013, accessed: 2018-09-20.

[9] E. Kain, From 'codehunters' to 'aliens: Colonial marines': Gearbox is one of the most controversial studios in the business, https://www.forbes.com/sites/erikkain/2013/03/15/from-codehunters-to-aliens-colonial-marines-how-gearbox-has-become-one-of-the-most-controversial-studios-in-the-industry/#444f837f58ca, 2013, accessed: 2018-09-20.

[10] T. CGSociety, Codehunters, https://web.archive.org/web/20160312025227/https://cgsociety.org/index.php/CGSFeatures/CGSFeatureSpecial/mtv_codehunters, 2012, accessed: 2018-09-20.

[11] J. S. Mill, Utilitarianism., *Fraser's magazine*, *64*(384):(1861), 659–673.

[12] I. Kant, *Grounding for the metaphysics of morals: With on a supposed right to lie because of philanthropic concerns* (Hackett Publishing, 1993).

[13] T. M. Garrett, Business ethics.

[14] G. R. Laczniak, Framework for analyzing marketing ethics, *Journal of Macromarketing*, *3*(1):(1983), 7–18.

[15] H. Kantilaftis, Drawing inspiration from elsewhere to create new video games, https://www.nyfa.edu/student-resources/drawing-inspiration-from-elsewhere-to-create-new-video-games/, 2015, accessed: 2018-09-20.

[16] J. Garrett, A simple and usable (although incomplete) ethical theory based on the ethics of wd ross, *Online at http://www. wku. edu/˜ jan. garrett/ethics/rossethc. htm. Accessed*, *10*:(2004), 2011.

[17] D. Weijers, Hedonism, https://www.iep.utm.edu/hedonism/, 2010, accessed: 2018-08-28.

[18] Hedonism, http://philosophyterms.com/hedonism/, accessed: 2018-08-28.

[19] L. Plunkett, How warcraft was almost a warhammer game (and how that saved wow), https://kotaku.com/5929161/how-warcraft-was-almost-a-warhammer-game-and-how-that-saved-wow, 2012, accessed: 2018-09-1.

[20] S. Totilo, The games that inspired the year's best game developers, https://kotaku.com/the-games-that-inspired-the-worlds-best-game-developers-1765500505, 2016, accessed: 2018-09-11.

[21] Asset flipping, http://crappy-games.wikia.com/wiki/Asset_Flipping, 2017, accessed: 2018-08-29.

[22] C. Graham, Does more money make you happier? why so much debate?, *Applied Research in Quality of Life*, *6*(3):(2011), 219.

[23] CM30, The top 20 worst video game ripoffs, https://gamingreinvented.com/nintendoarticles/the-top-20-worst-video-game-ripoffs/, 2016, accessed: 2018-01-20.

[24] Wretth, [steam] (game) digital homicide mixed pack - 21 games, https://www.reddit.com/r/FreeGameFindings/comments/5rzxt0/steam_game_digital_homicide_mixed_pack_21_games/, 2018, accessed: 2018-01-20.

[25] D. Parsons, [updated] digital homicide's games removed from steam, https://techraptor.net/content/digital-homicides-games-removed-steam, 2016, accessed: 2018-01-20.

[26] A. Frank, Valve removes nearly 200 cheap, 'fake' games from steam (update), https://www.polygon.com/2017/9/26/16368178/steam-shovelware-removed-asset-flipping, 2017, accessed: 2018-11-07.

[27] thenerdofsuperstuff, All call of duty games, https://www.imdb.com/list/ls068935654/, 2017, accessed: 2018-09-20.

AN EXAMINATION OF THE ETHICS OF ASSET FLIPS

Brandon Packard , Jamie Phillips
Clarion University of Pennsylvania
bpackard@clarion.edu, jphillips@clarion.edu

ABSTRACT

Since their inception, video games have exploded in popularity, leading to categories of games that transcend genres. One such category is the asset flip, also known as shovelware. These games are known for being hastily cobbled together using premade assets, being full of bugs and other issues, and being sold for real money. Is this process of asset flipping morally acceptable, or is there something inherently wrong about churning out low quality games for money? In this work, we examine asset flips (and asset flippers) from various angles to determine their value in the gaming community, and then use various theories of ethics to comment on their morality.

1 Introduction

In the last couple of decades, video games have gained momentum as a preferred pastime of many people, and there is no indication that this momentum will stop anytime soon. Furthermore, the field has become large enough and dynamic enough that categories of games which transcend even genres have emerged. One notorious example of this is the Asset Flip (A phrase coined by the famous Youtuber Jim Sterling). Asset Flips Games are those games which have been built almost entirely out of premade assets that are either purchased from a store or ripped from online, with very little original work put into them [1]. Note we are not implying in this work, nor would we ever, that using premade assets is necessarily a bad thing. It is simply not feasible for very small game development teams to be able to make everything from scratch in many cases, so using premade assets is a valuable tool at their disposal. By combining these with their own assets and using the premade assets in creative ways, they can achieve great results without having to model, texture, animate, script, etc. every little thing in the game. With asset flips, however, there is very little in terms of original assets (and none, in many cases). The assets also do not tend to be used to create a cohesive game but are rather just slapped together into something that hopefully looks good enough to get people to buy it [1]. These games are viewed very negatively among gamers, and the influx of asset flips on the popular gaming platform Steam has even been referred to as the "Steampocalypse" [2].

In this paper, we will be addressing what we believe to be a crucial question in the ethics of game design: are asset flip games morally wrong? That is to say, is there some moral obligation to game designers to put effort into their games and make them a good experience for players, or is it perfectly acceptable to haphazardly mash together a bunch of premade assets in order to make money? Note that although we are focusing on asset flips in this work, our argument applies to any purely derivative work of art, such as TV show rip-offs, movie rip-offs, or early Lego sets [3].

In order to answer this question, we will examine asset flips from different ethical theories to show whether the process of asset flipping is morally impermissible. We will then see if there is anyway that we can overwrite any impermissibility and still condone their sale, and finally end by considering and refuting some objections to our arguments.

2 Ethics and Aesthetics

It seems quite intuitive that if something is not ethically permissible to sell, for some given definition of unethical, then it is wrong to sell that item. However, are there any exceptions to this? Take for example the Ford Pinto, a car produced by Ford in the 1970's. The car was marketed as a 2,000 pound car for less than $2,000, even being marketed as "Carefree" – a descriptor that became very ironic when it was found that the Pinto had a tendency to explode from even low-speed read-end collisions. Certainly, we would (and many did) condemn Ford's selling of this car knowing the dangers to be unacceptable. Can anything override this unacceptability? The only thing, if anything, that seems to have this potential is the idea of aesthetic value. Aesthetic value is the value that an object/event/etc. possesses if it is able to elicit pleasure or displeasure when appreciated or experienced aesthetically [4]. If a Ford Pinto has a high aesthetic value, that may be able to outweigh the concern about its safety. Similarly, we might override our moral concerns if a food being sold with the potential of E. coli is particularly delicious or rare.

Perhaps the most stark example of this is Disney. Take for example the Disney movie "The Lion King". This movie was billed as Disney's "first big animated feature that wasn't a retelling of a fairy-tale or previous story [5]". However, this is very widely regarded to not be true. Not only do

some of the scenes in The Lion King mirror a Japanese cartoon known as "Kimba the White Lion" (sound familiar, anyone?), but the plotline was very heavily inspired by Shakespeare's Hamlet. However, when was the last time you heard someone condemning The Lion King for any reason (other than just now, of course)? It doesn't happen often at all, but why? The simplest answer is because of its aesthetic prowess. The Lion King is a timeless movie, having spawned sequels, video games, a massive amount of merchandise, and so on. We as a society are so caught up with our fond memories of The Lion King and its positive values that we tend to forgive the almost-certainly morally impermissible conditions under which it was created and sold. As such, if there *can* be anything that could redeem the unethical selling of a product for profit, it would have to be aesthetic value. Therefore, if the creation of asset flips does turn out to be morally impermissible, for some given definition of morally impermissible, we then need to check them for aesthetic value. If there is also no aesthetic value to be found, then we will have no reason left to approve of their sale, and will be required to condemn it.

3 Ethical Lenses

Now that we have determined whether or not asset flips have value via a set of lenses, let us use this same strategy to determine if a business creating a product that has no value except to themselves is morally acceptable, by examining asset flips through a series of ethical lenses.

3.1 The Proportionality Framework

Utilitarianism

One view is that seeking personal gain at the expense of the common good is an unethical business practice to have [6]. The lens of utilitarianism provides a similar viewpoint, stating that actions are considered moral if they have the highest overall utility of the possible actions at a given time [7]. Who benefits from asset flips? Clearly the game development company does, and the gamers who buy their game and are disappointed do not, but there are two other factors to consider. First is the gaming platform, such as Steam. In the case of asset flips, Steam actually does make some money from the sale of trading cards [1] but is also losing non-asset flipping game developers due to the influx of asset flips [1]. This leads into the other factor to consider, the developers who aren't creating asset flips. These developers see asset flips as "the tapeworms of steam" and "a serious problem", and it is clearly affecting how they have to market and release their own games [2]. So, asset flips offer the developers a lot of utility, more or less come up even on game platforms, but distinctively hurt the players and the community as a whole. As such, we can be confident that utilitarianism would find the process of asset flipping morally reprehensible, especially compared to the process of creating and selling polished games (a practice in which all parties prosper).

Kantianism

Now let us look at the process of asset flipping from a Kantian lens. One of the key takeaways from Kant's first and second formulations of the Categorical Imperative is that people have a perfect duty to not use their humanity or anyone else's humanity merely as a means to an end – that is to say, the human should always be an end, not purely a means [8]. This is a quality we see in many other areas – people deserve to be informed about what ingredients are in food, what materials clothes are made of, or what issues a house or car might have before purchasing any of those items. Doesn't this mean that if players are knowingly purchasing asset flips in order to pay them, they are showing informed consent and therefore being used as ends, which is acceptable under the Categorical Imperative? There is certainly merit to this line of thinking, but there is also one crucial flaw in this viewpoint: players typically *don't* know they are purchasing asset flips! Recall that asset flips are viewed very negatively in the gaming community; if players know that a game is an asset flip, they are very unlikely to want to spend their money on it. Therefore, asset flips are never modeled as asset flips, but are instead presented as any other game. In fact, usually the only clues that you have that a game might be an asset flip are (1) if you can recognize the assets in the screenshots, (2) the developer is a well known asset flipper, or (3) via the reviews from those who have already purchased the game. Remember that asset flipping entails that the developers are creating the games specifically for the sake of profit. If people knew games were asset flips, it would cut seriously into the games purchases, and by extension significantly into the developers profits. Therefore, they actively try to hide this quality (or lack of quality) from players – and what could possibly be using a human as a means more than trying to actively fool them in order to make money? Asset flippers are using gaming platforms, gamers, and the gaming community as a whole as purely a means for the sole purpose of making money, a behavior which Kant would certainly condemn.

Proportionality FrameWork

Garrett provides a lens that attempts to combine these two lenses, a multidimensional model of business ethics called the Proportionality Framework [9]. This framework contends that ethical business decisions can be split into three components: intention, means, and end. These refer to the motivation behind a person's actions, the process or method used to bring about ends, and the outcomes of an action, respectively. Although the details are a bit vague, the idea is that if two of the components are strong enough, they can outweigh a negative third component to make a decision be considered moral overall (for example, if a doctor wants to help a patient and performs the correct surgery in the correct way, but the patient ends up passing away, we wouldn't consider this morally impermissible because the intentions and the means make up for the less-than-desirable ends). However, since we have already shown that asset flip game companies do not have good intentions (making money for themselves with little concern for much else), or good means through Kantian-

ism (using other humans and the community as their means), or good ends via Utilitarianism (they make money but cause issues for gamers and the gaming community), their actions would also be considered immoral under this framework.

3.2 Teammates and Televisions

Laczniak [10] briefly explains two ethical frameworks that are specifically aimed at ethics in businesses. The first is the "professional ethic", which states that actions are moral only if a disinterested panel of professional colleagues says that they are. Given that game developers are leaving Steam over asset flips and the community generally disapproves of this type of game, we believe it is safe to say that a randomly selected panel from the gaming community would absolutely condemn the actions of asset flippers. The second framework is the "TV test", which simply states that for person in a business to know whether to take an action, they should ask themselves if they would feel comfortable explaining to a national TV audience why that action is the one that was chosen. This is a tough lens to use, as different developers would have different levels of "shame". There are almost certainly developers who would be ashamed to stand on national TV and explain what they do, but there are others that would have no problem doing so. In fact, one of the most notorious asset flippers is a game studio called Digital Homicide. In addition to their many asset flips, they are known for trying to sue both a Youtuber and 100 anonymous users (for whom Digital Homicide tried to subpoena Valve for real identities) for slander (and for millions of dollars!), just for saying that their games are bad [11; 12; 13]. This being said, we do think it is safe to state that the average person would feel uncomfortable standing in front of a national TV audience and telling everyone that they make very low-effort games and sell them on gaming platforms for the sole purpose of making money. Therefore, this framework would also frown upon the actions of asset flippers.

3.3 Prima Facie Duties

Laczniak [10] then goes on to describe what he believes to be more comprehensive frameworks for business ethics, such as the Prima Facie Duties framework. This framework, based on the work of W.D. Ross, is based on the notion that there are several "at first sight" duties which are self-evident and constitute moral obligations under most circumstances. There are eight duties Ross puts forth, although they can be condensed into seven [14]: Beneficence, harm-prevention (which can be seen as part of beneficence), non-maleficence, self-improvement, gratitude, reparation, fidelity, and justice. The first states that one should do good, if given the opportunity, and the second states that one should not harm others. Duties of self-improvement state that actions should be taken which promote one's own self interest. The duty of gratitude says that, when possible, you should show benefactions to those who show benefactions to you, and the duty of reparation says that you need to make up for the injuries you have done for others. The duty of fidelity says you need to keep your promises and contracts and not engage in deception. Finally, the virtue of justice requires you to act so as distribute benefits and burdens fairly (note that this seems to dis-include being a parasite, or soaking up benefits for yourself while lauding burdens upon others). These duties all stem from moral intuition – that is to say, we can just naturally *see* that non-maleficence is a good rule to follow. Similarly, when duties conflict (as they often will), moral intuition tells us what the priorities are. For example, we can just *see* that non-maleficence takes precedence over beneficence in the general case.

This framework is typically used for medical or business decisions. For example, say you were going to bring a new medicine to market. Using the first three duties for simplicity (in reality, all seven duties would have to be considered), the questions you should ask yourself are (1) is this medicine actually going to help anyone, (2) is this medicine going to hurt anyone, and (3) is this medicine a step in improving your own medicine-creating skills? If all three duties are met, then bringing the medicine to market is a moral action. If not all of the duties are met, then you need to look at the priorities of the duties to determine if the action is moral (for example, other factors being equal, non-maleficence normally overrides other duties). This can be applied to business actions as well. If you are the CEO of a large company and are going to roll out a new policy, you have to consider (1) if it is improving yourself and the company, (2) if it is going to benefit others, and (3) if it is going to harm others.

As asset flipping is a business, we can easily examine it via these duties as well: Asset flips do not really provide any benefit to the community, and in fact can even be said to harm it (as the influx of asset flips can drown out other games and developers and make it harder for players to find new and interesting games). Therefore, asset flips certainly violate the duties of beneficence and non-maleficence. They do seem to follow the duty of self-improvement – asset flipping over and over would improve one's skills at asset flipping, and the money gained from the process gives them some security if nothing else. We would also argue that their blatant disregard for the players who are making them money (as previously discussed) violates the duty of gratitude, and their refusal to in any way make up for harming the community violates the duty of reparation. Interestingly enough, the duty of fidelity specifically states that you are not to engage in deception. As we previously discussed, asset flip developers actively deceive players in order to make money, making this one of the most clear duty violations. Finally, asset flippers violate the duty of justice due to their parasitic tendencies – they soak up benefits (i.e. money) by distributing burdens to others. As such, asset flips violate six of the seven duties described by Ross. Since this framework is based on moral intuition, we are sure that you are already aware that it would find asset flipping morally impermissible due to this high number of violations.

In conclusion, we have used seven different ethical lenses to

analyze the morality of creating asset flips. Every single one of these ethical theories/frameworks would deem the process of asset flipping to be morally impermissible. Therefore, it is tempting to state that the creation and selling of these games is morally wrong. However, there is one more factor we need to check first. If asset flips are of sufficient aesthetic value, then perhaps we can overrule this moral impermissibility and still approve of their sales.

4 Asset Flips and Aesthetic Value

In order to determine whether asset flips really do have aesthetic value, we are going to use an approach loosely based on "The Art of Game Design: A Deck of Lenses" [15]. This tool meant for aiding game design is presented as 100 individual cards. Each card presents a lens from which to look at the game you are creating, as well as an illustration and some related questions you should ask yourself while creating the game. The idea here is to conduct an aesthetic evaluation of asset flips by using this Deck of Lenses as a proxy method. Specifically, what we have done is go through the deck and pick out all of the cards which seem inherently relative to asset flips. We will now go through these cards one by one, using each lens to analyze the value of asset flips, and then draw an overall conclusion on what sort of value asset flips have from combining all these viewpoints. To try and avoid bias, we picked these cards strictly on their relevance to the process of asset flipping, not for which side they would provide evidence, and we will be discussing them in the numerical order in which they appear in the deck.

4.1 The Lens of Fun

The first lens from which we will be examining asset flips is the lens of fun. Are asset flips fun for gamers? This is a bit of a tough question, as fun can be quite subjective from gamer to gamer. One metric for evaluating fun comes from Mark Rosewater [16], who states that you should have someone you don't know play the game, and ask them if they would play it again. If their response is anything other than an enthusiastic "yes", the game isn't fun enough. A simple glance at the reviews and comments on the Steam pages for asset flips will tell you that most asset flips get not an enthusiastic "yes", but rather an enthusiastic "no".

One criteria that there does seem to be a consensus on is that of solid mechanics/gameplay [17; 18]. Games which have effective, intuitive mechanics or solid gameplay offer the potential for great fun, but games which are unfair or have controls gamers find to be undesirable are very commonly deemed to not be fun. Where do asset flips fit in on this spectrum? As so little effort is put into them, asset flips often have very choppy mechanics, or mechanics that just don't quite seem to "mesh" together (note that basic functionality scripts such as movement and attack scripts are also assets, and are usually also bought/ripped for asset flips). Between the clunky-if-functional controls and the overwhelming consensus on the Steam store reviews that many asset flips are not worth buying, we believe is it reasonable to state that asset flips are not generally conceived to be fun, and therefore do not have any aesthetic value according to this lens.

4.2 The Lens of The Elemental Tetrad

This lens examines games by whether four crucial elements of games interact in harmony within a game. These four elements are Aesthetics (graphics and visuals), Technology (what is used to interact with the game), Mechanics (How the game plays and controls/feels), and Story (Any storyline behind the game). If a game has a strong showing in all four categories, that is a strong piece of evidence that the game is "good". Do asset flips have a strong showing in all four categories? Almost certainly not. The aesthetics in asset flips tend to be underwhelming. It is perhaps intuitive to think that buying assets with better graphics will lead to overall better aesthetics in a game, but this is a little misleading. Even if the highest quality assets are bought, the way in which asset flip games tend to cobble them together leads to a game that just doesn't look quite right, because the game looks like a mishmash of assets instead of being formed into one coherent whole. The technology in asset flip games is nothing special, but it's sufficient. You aren't likely to see anything like biofeedback or voice capture, but the simple keyboard and mouse technology has been used in games for a long time and has proven tried and true. Mechanics are often lacking in asset flips as well, due to them often being pre-made scripts that are combined with minimal effort, as discussed in the previous lens. Finally, asset flips almost never have a good story (a good story is something that usually takes a large amount of time and effort to create, both things that asset flippers tend to not employ). Take as an example asset flip horror games (a genre in which story is typically crucial). An incredible amount of indie asset flip horror games start the exact same way: some relative has left you a house, you go to check out the house, fall asleep or otherwise unconscious, and start gameplay by waking up in a horror environment. Past this or another brief intro, many asset flips have either an incomprehensible story, or no story at all (such as Despair [19; 20] or basically any Jeff the Killer based game – the reader is referred to Jon Wolfe for commentary on these games [21]). Therefore, it can safely be concluded that not only are all four elements not in harmony, but in fact three of the four are typically very weak in asset flips, showing that they have no aesthetic value when examined through this lens either.

4.3 The Lens of the Player

A quite appropriate lens, the lens of the player seeks to examine games based not on the game, but on how cognizant of the player the game is – what do people like and not like, what do they expect, if we were the player what would we want, etc. The view through this lens is quite clear – asset flips are not created with the player in mind, at all. Recall that asset flips are games which feature minimal effort and

are primarily intended to make money for the developer. You will notice that nowhere in that definition is the player mentioned or considered. It could be argued that asset flippers have to have the player in mind, as they need people to be willing to buy their games in order to make money, but this is thinking about money – NOT thinking about the player for their own sake. As with the previous two lenses, no value can be found in asset flips from this lens.

4.4 The Lens of Pleasure

This lens is similar to the lens of fun, but in this paper, we will use "fun" to refer to the games overall enjoyability and "pleasures" to refer to the individual aspects that a play might find enjoyable. For example, a player may find killing enemies to be very satisfying, but overall not care for the game. One such game is Dead Island, widely lauded for its fun gameplay and criticized for most other aspects of the game [22]. Do asset flips have value when viewed through this lens? Just as it's a rare and impressive feat for a game to do everything right, it's also incredibly hard to do everything wrong (imagine, for example, how hard it would be to get the minimum possible score on the SATs – surely you would get SOME things right even if by pure luck!). Therefore, many asset flips offer some minor pleasures to the player. It can be said then that asset flips do have a little value in this category, although the number of pleasures provided is so low that they are far outweighed by frustrations, and therefore any value gained by them is trivial.

4.5 The Lens of Expected Value

The lens of expected value is an interesting one. This lens basically asks if the game does what the player expects – if the player walks through a door, does the space they are in make context with the rest of the game, or if the player uses an item what happens is a reasonable used of that item (such as using a key to open a lock). Now, there are some games that go for more of a surreal feeling [23], and there's nothing wrong with that. However, when a game is not specifically created to be surreal, consistently going against the players expectations tends to lead to a jarring experience of which not many players are fond (This has a backing in psychology, as Rogers and Monsell found that in general, people do not perform as well on tasks involving a lot of context switching [24]).

Do asset flips meet the player's expectations? This is not an easy question, as it seems to depend on whether or not the player knows in advance that it is an asset flip! If the player does not know it is an asset flip, they will likely find much of the gameplay to not meet their expectations. Remember that asset flips are a hodgepodge of assets without any real effort put into unifying them into one coherent whole. Therefore, in addition to the inherent problems this causes (going through a door and loading a new scene which is far too large to be the room you just entered in the previous scene), there are also the more blatant issues. Let's say that the player picks up a hammer. If a puzzle then involved using the hammer to pound in nails or smash something, this would go well with the players preconceived notions of what a hammer does. However, if the hammer were used to stir a pot of soup, that certainly isn't what most people expect (another great example is the splash attack in Pokémon. The first time you use it, you are expecting it to at least do something, but surprise, it does literally nothing [25]). As asset flips do not put a lot of effort into unifying the game, or in many cases even into selecting the right assets, weird encounters like these are quite commonplace. But what if the player knows in advance that a game is an asset flip? Well in that case, the player is expecting the game to be hastily constructed or not unified/coherent – an expectation which, although usually met, we are uncomfortable lauding as a merit in any sense (as this is not the lens of *expectations*, but rather the lens of *expected value*).

4.6 The Lens of Beauty

The lens of beauty seeks to examine games based on whether or not they are "beautiful". This can either mean if each element is beautiful, or if they are beautiful in combination. As discussed in many of the previous lens, the combination or unification of assets is one of asset flips' key weaknesses. As such, let us examine the elements of the game on their own. As these elements are usually bought on an online store where someone else has put in the hard work to create them, they can be very beautiful depending on the assets employed. There is, overall, a large consensus that video games can in fact be considered art [26; 27; 28; 29]. That being said, although games are art and have the potential to be beautiful, you would be hard pressed to find a gamer that would refer to a series of models and textures haphazardly piled together as beautiful. Asset flips may use some beautiful materials in their games, but in the case the value is certainly lesser than the sum of its parts.

4.7 The Lens of Nameless Quality

This lens focuses on determining whether a game feels special or wonderful due to having a natural, organic design. We liken this to immersion – if the game world comes together as a cohesive whole and pulls you into it, it has this unique, nameless quality that is hard to describe. Having the in-game world come together and appear to be a consistent world with its own rules, logic, and so on allows gamers to truly immerse themselves in the game world and therefore enjoy it more. Take horror games as an extreme example of this. To scare seasoned gamers, horror games have to have a couple of qualities, arguably the most important of these being immersion. By immersing the player in the game world, game developers are able to slowly increase the tension and use the player's own imagination against them, drastically increasing the effect of any scares used in the game [30]. Immersion in general is always a tough sell, the smallest inconsistency can remind the user that they are playing a game and break

their immersion in the game world (such as if an in game story piece has many typos). What is important to get out of this is that immersion is hard to pull off. Games that do not unify their various elements into one cohesive world therefore are not able to create immersion – they do not feel natural or organic, but rather like a bunch of disparate items forced together (in fact many games that *do* unify their elements still don't manage to create immersion). Considering that that is literally what an asset flip is, we will find no value in them from this lens either.

4.8 The Lens of Love

This lens asks one vital question about designing a game: do the creators of the game love the game? This lens asks whether the game developer loves the game for the game itself, not for fame or anything else that the game may bring. When you love a game, you pour your time and effort into it to really make something fantastic. Rapidly churning out games in order to make money clearly shows a lack of love for your creations, and therefore a lack of value when examined through this lens.

4.9 The Lens of Playtesting

Playtesting is when you give your game to players (preferably people you don't know) and have them play the game and give you feedback on what they liked and didn't like. This lens asks if game designers have playtested properly and incorporated feedback from those playtesting sessions to make the game the best it can be. As one of the authors has worked for multiple years with a course which requires user testing and playtesting as part of the course, we can tell you two things. One, playtesting takes time. You have to set up the game or part of the game you want them to play, set up appointments, carry out the testing, analyze the results, and then incorporate any feedback into the next version of the game. Two, playtesting almost always brings up issues that you didn't know exist and almost always improves the game when feedback is incorporated. Game developers tend to get so locked into their own viewpoints that they miss things that playtesters pick up on right away. The first point is evidence enough of asset flips failure to properly playtest: asset flips are so low effort that there is no time or resources allocated to playtesting. If you need any more evidence that playtesting is not incorporated into these games, their low scores on Steam and other platforms clearly indicate that players had many complaints about these games that effective playtesting would have fixed. Since asset flippers clearly do not playtest their games properly, the gameplay suffers and prevents asset flips from having any value based on this lens.

4.10 The Lens of the Client

Similar to the lens of the player, this lens states that when making a game for someone else, you need to know what they want (and provide it for them). In most cases, gamers ARE the client, so the value through this lens is the same as the value in the Lens of the Player (aka none). Let us then examine the other possibility, that a client is buying the game in order to have someone else play it. In this case, the premade assets and fancy screenshots of the game may lure the client into buying the game. However, this is still the game developers considering the client's money, not the client themselves. When considering the client, a game developer needs to consider what the client says they want and what they really want, not just what will make them spend money on the game. Therefore, we will also find no value in asset flips when viewed from this perspective.

4.11 The Lens of Responsibility

This lens looks at whether or not a game meets its responsibilities as a game. This is a hard lens to use, as it is hard to define exactly what responsibilities a game has to the community. However, as we have established that asset flips hurt the community and actively try to deceive gamers, and as it is safe to say that harming the community is *not* one of the responsibilities a game has to the community and most likely violates at least one responsibility, we will find no value from this lens either.

4.12 The Lens of Secret Purpose

This lens simply asks game developers: by making this game, are you working towards your one true purpose? Without commenting on the moral correctness of having money be your only true purpose, it can conclusively be said that asset flips are created purely to make money, and that they do in fact make money. As such, when viewed through this lens, asset flips are given some value, but once again only in the sense of making money for the developers.

5 Discussion

When viewing asset flips from all different angles and from all different lenses, it becomes clear that the only real value these games offer is to make money for the developers, which is not aesthetic value. Players (generally) do not enjoy them or think highly of them, they don't do anything new or unique, and overall do not offer any benefit to the gaming community whatsoever, only to the developers.

We already know that asset flips are morally impermissible according to seven ethical theories and frameworks. As such, the only way that we may still be able to approve of the development and sale of these games is if they provide sufficient aesthetic value to offset that moral impermissability. Fortunately, the exact conversion ratio of morality to aesthetic value is irrelevant here, since there is *no* aesthetic value whatsoever to these games! If there isn't any aesthetic value to asset flips, then there certainly isn't enough to overturn

their moral impermissability. As such, our last rationale with which to approve the sale of asset flips does not apply, and therefore we are left with no other choice than to condemn the sale of these games.

6 Possible Objections

We will now examine three possible objections to our argument that the development and sale of asset flips is condemnable, and refute each one.

6.1 The Lens of Profit

One of the lenses in the deck that does not relate to aesthetic value, the Lens of Profit, specifically looks at whether or not a game makes money as an indication of its value. One thing that can be said for asset flips: they are profitable. First of all, if they weren't, there wouldn't be so many game development companies that have adopted the strategy of asset flipping. Second, asset flips (specifically those on Steam) are actually profitable in two ways. The obvious one is the revenue from the sale of the games. Even though Steam has a pretty nice return policy (owned for less than 2 weeks with less than 2 hours of playtime), many people have bought these games and simply never bothered to return them (think about the last time you got a mail-in rebate and didn't send it in for one reason or another). The second, much less obvious form of revenue is the Steam trading cards. Steam trading cards are virtual objects players get from playing games which have some value in the Steam Marketplace (dictated by supply, demand, etc.). Players who purchase asset flips will receive some of these trading cards, which they can then sell on Steam for real money. In these real-money transactions, both Steam and the developer of the game receive a share of the profits. Therefore, when viewed through the lens of profit, asset flips are actually quite valuable. Therefore, this lens seems to assert that the need to make a profit can override both moral and aesthetic values. However, this seems to be a weak objection – in what other area of life do we count the need to make a profit as most important? In any other area of business, we would condemn companies which only care about profit. In our social lives, we would distance ourselves from those who only performs actions that benefit themselves. In education we would condemn schools that exist purely for profit and not to actually teach their students. In fact, many movies conclude with characters taking jobs that make them less wealthy but more happy! There is nothing special or significant about asset flips that should make us treat them any differently from these other areas, and therefore this objection does not have any true merit.

6.2 Project vs Product

Allow us to play devil's advocate for a moment. Perhaps the asset flip was a game developer's first game, or perhaps they were just trying out something new and it didn't work out. Is learning and trying new things really something we want to fault game developers for?

The key refutation for this objection is that game development can be either a hobby or a business. The difference is basically the difference between a project and a product. If one releases their games for free, then it is a project. Once games are released for money, however, they stop being a project and start being a product. It is generally accepted that once something has become a product, it is held to a different standard than a free project. Since asset flips are released for money, it is clear they fall into the product category, and that the process of making asset flips and selling them for money constitutes a business.

Let us now split up that categorization further. Are asset flips simply a business (that is to say, to be treated as any other business), or is there more to it, some higher level of responsibility than a typical business? For the latter to be true, we would have to assert that game developers have more responsibility to gamers than Expo© has to the customers who buy its whiteboard markers, or more responsibility to the gaming community than Taco Bell© has to the restaurant community. Based on our time spent playing with games, working with games, and perusing game development literature, there is simply no evidence that this is the case. Therefore, game development should be treated as any other business, and our argument still applies.

6.3 Some Value is Better than no Value

Although many asset flips are sold at higher prices, there are also many that are sold at very low prices. As such, one could argue that these games, due to their inexpensive nature, allows players to have access to a whole host of games they may not otherwise be able to afford. This would then maximize the overall happiness because those who can afford the "better" games will buy the better versions, and those who cannot will still be able to garner some happiness from the rip-off asset flips.

This is an interesting objection, as it is true that some people will afford some asset flips, and if you cannot afford the "good" games you may still be able to get some pleasure out of those that you can afford. However, there are at least 2 key counterpoints to this objection. The first counterpoint is that, generally speaking, asset flips are *not* fun to play. Not "they aren't as fun to play as a $50 game" but rather "they aren't fun to play *at all*". One bit of evidence for this is that Valve, the owners of Steam, publicly refer to asset flips as "fake games" and their creators as "fake developers" [31]. Furthermore, crafting a fun game takes time. Take as an example the very popular Call of Duty series. This series is made by a AAA company with a ton of manpower, regarded as a mechanically sound game but nothing unique or special, and they only average about one new game a *YEAR* [32]. On the other hand, an asset flipper known as Silicon Echo Studios developed and placed 86 games on steam in just 2 months, accounting for at

least 10% of all the games published to Steam in that time period [33]. For reference that's more than one game being developed *PER DAY*. Anyone in the game development business will tell you that this is much too short for most games to be any fun at all, let alone to actually be sold for money.

Let us look at this another way. Let's say, for sake of argument, that asset flips are sold for $1. I'm picking this amount because it's the lowest price that games are typically sold for (and games are usually sold for more, somewhere around $3 to $15, depending on how bold the developer is). To get a high quality, "good" game, it's typically not more than $50 at most. That means that, in the best case, asset flips only cost 1/50th of high quality games. Therefore, if it can be said that they provide at least 1/50th of the happiness of high quality games, there may be some merit to this objection. One way to (loosely) capture fun is average playtime, as people tend to play games longer when they are more fun. Many of the games that would be considered high quality on Steam have over 50 hours of gameplay on average [34] while many asset flips have less than 10 minutes of gameplay (only around 1/300th as much time). This is due to people opening the game, realizing how bad it plays or how poorly it's put together, and then refunding it immediately to get their money back. This is evidence that asset flips do not in fact provide the correct proportion of happiness to match their price tag. This perhaps is not surprising – would you rather read 10 really badly written 25-page books or one really well written 250 page book? Not only would reading 10 really badly written and short books not provide as much pleasure as one really well written book, but it's likely you would get frustrated with how bad the books are and enjoy each one less and less! When you consider that these are conservative values (for example, Borderlands is a fantastic game with many hours of gameplay for only $20), and there are great games which cost much less and asset flips which cost much more, it becomes clear that pound-for-pound asset flips still can't hold their weight. Furthermore, the median price of a game on Steam (as of April 2018) is $5.99, with the median price of indie games (i.e., not made by a huge development studio such as EA) being $3.99 [35]. Therefore, on top of the other problems with this objection is the fact that asset flips are about $2 cheaper when considering the median prices, and even at a measly $1 would only be 1/6 of the price, nowhere near the 1/50th that was previously discussed. You would increase overall happiness more by scrimping and saving $20 over time to buy one really good game, instead of buying and playing 20 $1 games one at a time.

6.4 Hedonism

Hedonism is a component of Utilitarianism which states that "good" simply means having pleasure and the absence of pain. As such, it states that things which cause pleasure are good [36; 37] – for example, helping others is good because you get true pleasure from doing so. As hedonism is all about increasing your own pleasure, surely creating Asset Flips would be moral under hedonism, right? Well, not exactly. Making money (the main goal of asset flippers) would not be considered a moral action unless it maximizes their pleasure. Now, you could argue that if money can buy true pleasure, maximizing profits would be maximizing pleasure, and therefore be morally acceptable. However, life in the real world isn't quite so simple. There has been a lot of debate over whether money buys happiness. Graham [38] attempts to examine the complexities underlying this debate, and has found that the issue is a lot more complicated than most studies will say. Her take-home point is that there are many factors in the issue and although making more money helps, more money does not necessarily mean more pleasure. And if you need any more evidence from the game development point of view, simply remember back to the now-defunct studio of Digitial Homicide, or to any other number of asset flippers and how they respond to the gaming community, and you will find a running theme that mass producing these games for money has not given them any true pleasure [39; 40]. As such, not even a theory that heavily involves maximizing one's own pleasure can justify the business actions of asset flippers.

6.5 Legal Moralism

What about legal moralism, or the idea that it is moral to do something if it is legal to do so [41]? Legal moralism would say that since creating Asset Flips is legal, it is a moral action. However, this is a moral theory that simply doesn't hold up. Slavery was once legal, same-sex marriage was once illegal, and there is still a law in PA stating "Any motorist who sights a team of horses coming toward him must pull well off the road, cover his car with a blanket or canvas that blends with the countryside, and let the horses pass." [42]. Odds are that most people would be uncomfortable with basing their morals on any of these laws, but it also illustrates a bigger point. The law *changes*. The law is constantly changing, many times to reflect what the people want. Can we really base our morals on something that is so fluid? Furthermore, basing our morals purely on laws means basing our morals purely on what the people making the laws believe, and not what we believe in our hearts. As such, legal moralism does not hold up as an objection to our claim that creating and selling asset flips is immoral.

7 Conclusions

In this paper, we have examined asset flips to see if the process of creating these hastily cobbled together games and selling them for real money is morally wrong. To this end, we first analyzed asset flips through a series of seven different ethical theories/frameworks, and found that all seven would deem it to be morally impermissible, mainly based on the levels of deception which the creators employ.. We then stated that sometimes we can still accept the sales of items even when it is morally impermissible for them to be sold just for a profit, but only when the item has sufficient aesthetic value. We then determined through a set of through a series of 12

different lenses meant to analyze games from various angles that asset flips do not in fact have any aesthetic value, and therefore we must reject their sale as morally wrong. We then considered three objections to these arguments and refuted each. The conclusion is that when it comes to being morally permissible, asset flips are nothing but a flop.

References

[1] Asset flipping, http://crappy-games.wikia.com/wiki/Asset_Flipping, accessed: 2018-08-29.

[2] In defense of asset flips on steam, https://venturebeat.com/2018/07/12/in-defense-of-asset-flips-on-steam/, 2018, accessed: 2018-08-29.

[3] D. Faraci, Kiddicraft, the company lego ripped off to make plastic bricks, https://birthmoviesdeath.com/2014/02/06/kiddicraft-the-company-lego-ripped-off-to-make-plastic-bricks, 2014, accessed: 2018-11-01.

[4] L. Plato, A. Meskin, Aesthetic value, *Encyclopedia of Quality of Life and Well-Being Research*, pages 76–78.

[5] B. Bradley, as 'the lion king' copied from a japanese cartoon? here's the real story, https://www.huffingtonpost.com/2015/01/27/lion-king-kimba_n_6272316.html?slideshow=true, 2017, accessed: 2018-11-01.

[6] W. J. Byron, The meaning of ethics in business, *Business Horizons*, *20*(6):(1977), 31–34.

[7] J. S. Mill, Utilitarianism., *Fraser's magazine*, *64*(384):(1861), 659–673.

[8] I. Kant, *Grounding for the metaphysics of morals: With on a supposed right to lie because of philanthropic concerns* (Hackett Publishing, 1993).

[9] T. M. Garrett, Business ethics.

[10] G. R. Laczniak, Framework for analyzing marketing ethics, *Journal of Macromarketing*, *3*(1):(1983), 7–18.

[11] M. Cosimano, Indie developer digital homicide sues jim sterling, https://www.destructoid.com/indie-developer-digital-homicide-sues-jim-sterling-349283.phtml, 2016, accessed: 2018-08-28.

[12] P. Klepek, Angered game developer sues critic jim sterling for $10 million, https://kotaku.com/angered-game-developer-sues-game-critic-jim-sterling-fo-1765484317, 2016, accessed: 2018-08-28.

[13] R. Grosso, Digital homicide suing 100 steam users for 18 million, https://techraptor.net/content/digital-homicide-suing-100-steam-users-18-million, 2016, accessed: 2018-08-28.

[14] J. Garrett, A simple and usable (although incomplete) ethical theory based on the ethics of wd ross, *Online at http://www. wku. edu/˜ jan. garrett/ethics/rossethc. htm. Accessed*, *10*:(2004), 2011.

[15] S. Jesse, *The art of game design: A book of lenses* (Morgan Kaufmann Publishers, Boston, MA, 2008).

[16] M. Rosewater, Ten things every game needs, https://magic.wizards.com/en/articles/archive/making-magic/ten-things-every-game-needs-part-1-part-2-2011-12-19, 2011, accessed: 2018-08-29.

[17] Pluralsight, What makes a great game? the key elements of successful games, https://www.pluralsight.com/blog/film-games/what-makes-a-great-game-the-key-elements-of-successful-games, 2014, accessed: 2018-08-29.

[18] A. Kadle, 9 essential elements for fun in games, https://www.upsidelearning.com/blog/index.php/2011/01/31/9-essential-elements-for-fun-in-games/, 2011, accessed: 2018-08-29.

[19] A. Hodkinson, The worst game on steam, https://boardgamegeek.com/geeklist/194616/worst-game-steam, 2015, accessed: 2018-08-29.

[20] Steam store – despair, https://store.steampowered.com/app/368990/Despair/, 2015–2018, accessed: 2018-08-29.

[21] J. Wolfe, Jeff the killer games, https://www.youtube.com/playlist?list=PLqfNLQec1ZRlMzPdi-GFOuwg9XaVVI5zM, 2013–2018, accessed: 2018-08-29.

[22] G. Miller, Dead island review, http://www.ign.com/articles/2011/09/04/dead-island-review, 2011, accessed: 2018-08-29.

[23] R. Briean, Surrealist games that you must play, https://www.argyllfreepress.com/2015/05/04/surrealist-games-that-you-must-play/, 2016, accessed: 2018-08-29.

[24] R. D. Rogers, S. Monsell, Costs of a predictible switch between simple cognitive tasks., *Journal of experimental psychology: General*, *124*(2):(1995), 207.

[25] ShortList, 20 weirdest video game weapons, https://www.shortlist.com/tech/gaming/20-weirdest-video-game-weapons/3192, 2013, accessed: 2018-08-29.

[26] A. Smuts, Are video games art?, *Contemporary Aesthetics*, *3*(1):(2005), 6.

[27] A. Cruz, Are video games art?, https://angelcruz3.wordpress.com/2015/03/29/are-video-games-art/, 2015, accessed: 2018-08-29.

[28] J. P. Gee, Why game studies now? video games: A new art form, *Games and culture*, *1*(1):(2006), 58–61.

[29] C. Melissinos, P. O'Rourke, *The art of video games: From Pac-Man to mass effect* (Welcome Books New York, NY, 2012).

[30] D. Heaven, How the scariest video games use our own minds to terrify us, https://www.newscientist.com/article/dn28375-how-the-scariest-video-games-use-our-own-minds-to-terrify-us/, 2015, accessed: 2018-08-29.

[31] Steam, Changes to trading cards, https://steamcommunity.com/games/593110/announcements/detail/1954971077935370845, 2017, accessed: 2018-11-01.

[32] thenerdofsuperstuff, All call of duty games, https://www.imdb.com/list/ls068935654/, 2017, accessed: 2018-09-20.

[33] A. Frank, Valve removes nearly 200 cheap, 'fake' games from steam (update), https://www.polygon.com/2017/9/26/16368178/steam-shovelware-removed-asset-flipping, 2017, accessed: 2018-11-07.

[34] Astats, Steam games overview: Ordered by average playtime, https://astats.astats.nl/astats/Steam_Games.php?Sort=8, 2019, accessed: 2019-03-22.

[35] A. Wawro, Steamspy: $20and60$ were the top-earning steam game prices last year, https://www.gamasutra.com/view/news/316146/SteamSpy_20_and_60_were_the_topearning_Steam_game_prices_last_year.php, 2018, accessed: 2019-03-22.

[36] D. Weijers, Hedonism, https://www.iep.utm.edu/hedonism/, accessed: 2018-08-28.

[37] Hedonism, http://philosophyterms.com/hedonism/, accessed: 2018-08-28.

[38] C. Graham, Does more money make you happier? why so much debate?, *Applied Research in Quality of Life*, *6*(3):(2011), 219.

[39] Ata berdiyev, http://crappy-games.wikia.com/wiki/Ata_Berdiyev, accessed: 2018-08-28.

[40] Silicon echo studios, http://crappy-games.wikia.com/wiki/Silicon_Echo_Studios, accessed: 2018-08-28.

[41] K. Himma, Philosophy of law, https://www.iep.utm.edu/law-phil/#SSH2a.i, 2010, accessed: 2019-01-20.

[42] K. Bayliss, Wow! these laws are stupid, https://www.nbcphiladelphia.com/the-scene/archive/Wow-These-Laws-are-STUPID.html, 2009, accessed: 2019-01-20.

THE USE OF HEURISTICS FOR ROBOTIC GUIDANCE

Krish Pillai, Ph.D. and Caroline Lee Rublein
Lock Haven University
kpillai@lockhaven.edu, clr2027@lockhaven.edu

ABSTRACT
This paper presents an inexpensive implementation a closed loop guidance algorithm that is driven by heuristics to achieve improved performance for line/boundary following. Advanced Robotics labs call for considerable capital investment (CAPEX) and operating expenses (OPEX). In this paper, the authors describe a semester-long project that was inexpensive to build, but facilitated extensive enquiry into advanced areas of robot motion control. The study described herein in restricted to mobile robots with three degrees of freedom (3DOF); capable of moving along X and Y coordinates, with the added capability to steer and rotate in-place, or reverse its directional vectors. The Lego EV3 platform, an off-the-shelf product was repurposed for this study. The software was implemented in embedded Java. The problem domain was limited to line-following, since a robust line and boundary following algorithm is an essential component of most spatial reasoning challenges in robot navigation. The algorithm presented here can be easily adapted to a multitude of problem domains through the choice of appropriate sensors and end-effectors. The implementation presented herein was extensively tested, and robotic sensor data acquisition were done through remote monitoring using TCP service.

KEY WORDS
Robot kinematics; control systems; heuristics, guidance algorithms;

1. Introduction

Robots are often created to do tasks generally qualified by the "Three D's": dirty, dull, and dangerous. Most of these tasks involve the robot having to navigate a workspace without colliding with objects, following a map that is constructed either dynamically, or by detecting a pathway that is summarily presented to it. For certain jobs such as automated truck driving, the robot would need to follow the general curve defined by the road ahead of it. The roadmap before such a vehicle may be autonomously derived through advanced image processing techniques or by sensing markings on the road or along the shoulders. Once a map of the environment has been generated and the robot has localized itself within the generated map, the task at hand is to follow the pathway in a precise and efficient manner. In this paper, the authors present a fairly sophisticated, yet inexpensive experimental setup that facilitates the investigation of important and advanced concepts of robotics control and guidance. The paper deals specifically with the setup and programming associated with implementing and analyzing path-following functionality in robots, combining established control systems theory with direct iterative computer control. This work was done as part of ongoing undergraduate research work on robotics and control systems at the authors' institution.

2. The Scope of the Guidance Problem

Edge following, in its simplest form, may be implemented using a bang-bang controller - a controller that switches abruptly between two states. Thermostats and simple electric instruments such as water boilers employ such "all" or "nothing" mode of control. This mode of control is frequently used in complex machines that operate on minimum-time tasks [1]. As in all manner of controls, the objective is to minimize some error signal detected by sensors, the error being described as the difference between a measured value and a desired value. But since bang-bang controllers can only be in one of two states, the error signal will tend to oscillate about the desired value, leading to wear and tear effects on the robot. The resulting oscillations will furthermore impede the onward progress of the robot since it would be cycling back and forth about the optimal pathway.

A more sophisticated approach is to use negative feedback with some form of proportional control [1]. In this case, the larger the error, the stronger the feedback signal would try and correct the source of the error. When tracking a pathway, the dynamics of the robot is again affected not just by its instance attributes such as the velocity and acceleration of the platform, but also by the nature of the traversed pathway itself. Under such conditions, navigation can be enhanced if the control loop retains memory of previous responses to stimuli, and in addition, is able to predictively respond to changing conditions. The Proportional-Integral-Derivative (PID) Controller [1] achieves this by using three stimulus terms to generate its control signal.

The primary corrective term is proportional to the error, which makes the corrective action depend linearly on the error. The second term, which is the integral of the detected error, is used to make older error readings affect the current corrective signal. The third term is the differential of the error, and it is used to make the corrective signal dependent on the rate at which the detected error changes over previous readings. The Superposition property [1] states that, "for all linear systems, the net response caused by two or more stimuli is the sum of the responses that would have been caused by each stimulus individually". By summing the individual PID terms, the control system can achieve near optimal control in guiding the robot with minimal deviations from its intended path.

2.1 The Structure of the 3DOF Robot

The robot chassis has two independently powered wheels in the front, and a free rotating wheel in the rear to maintain its balance. The independent drives allow the 3-wheeled robot to be differentially steered by varying the relative rotational velocity of each wheel. The robot is equipped with a color sensor, which is mounted close to the front axle of the robot. The color sensor is facing downwards, measuring the reflected light intensity (RLI) of the surface the robot moves on.

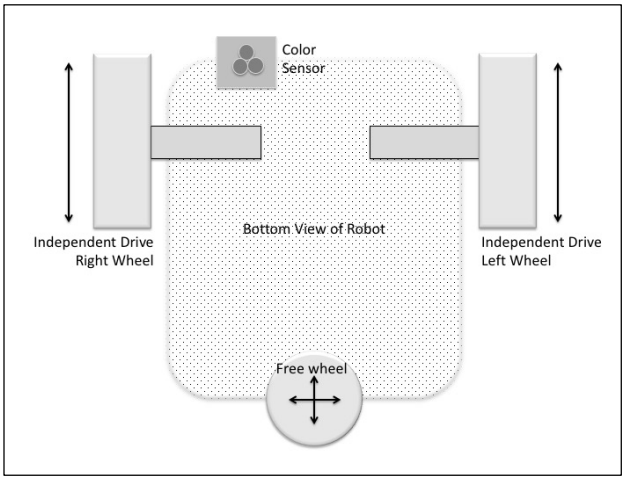


Figure 1: Schematic diagram of the 3DOF Robot

The robot was built using the Lego's Mindstorms EV3 robot kit, Education version. This kit provides sensors and motors, along with interlocking plastic bricks, axles of various dimensions and wheels and gears. The kit also comes with a controller, which runs a version of the Linux operating system. The controller, referred to as the EV3 brick, can also be booted from a micro SDHC card with a Debian based operating system that supports the ARM9 processor, designed for microcontroller use.

The algorithm was implemented using embedded Oracle Java SE 8. The pathway to be followed by the robot was marked out on a white board with 1" wide black tape, and the boundary to be tracked was arbitrarily chosen as the left edge of the tape. The choice of either edge has no significance on the structure of the algorithm, other than the fact that the control logic, and therefore the path followed would be reversed. The specifications of the controller EV3 brick are shown in the table below.

Display	178x128 pixel Monochrome LCD
Main Processor	TI Sitara AM1808 @300MHz
Main Memory	64 MB RAM and 16 MB Flash (micro SDHC)
USB Port	Yes
Wi-Fi driver	Yes
Bluetooth driver	Yes

Table 1: EV3 Brick Specifications

The Texas Instruments TI Sitara chip makes use of a single 32-bit RISC ARM9 processor core [4]. This core is no longer recommended by the manufacturer for new systems design. Taking this into consideration, data types were largely restricted to floats instead of the Java default of double, and multithreading was avoided wherever possible through synchronous software polling.

The EV3 Robot is a product from Lego® with a worldwide user-base. The Education Core Set is not expensive, costs about $400, and includes an impressive array of components to build various end-effectors suited for particular tasks. The programmable brick (Controller) runs Linux and can be booted from a Micro SD, making it highly configurable. SunFounder and BrickPi3 are other kits that have cheaper controllers, but those brands still require Lego blocks and parts designed for the EV3 to be fully functional, thereby having very little impact on the overall investment for a multi-purpose fully functional unit. Moreover, the customer support and knowledge base for the Lego® product is well established and the product will, in all likelihood, survive in the market much longer than the others the authors tested out. In addition to this, the EV3 provides its native OS, which allows users to use Graphical programming to rapidly prototype their algorithms. For all the aforementioned reasons, the Lego EV3 proved to be a better cost-effective solution that other similar robots.

2.2 Bang-Bang Controller

The basic negative feedback control loop is shown in Figure **2**. The light sensor detects the reflected light intensity (RLI), which is compared to a reference value. The objective of the control loop is to make the measured value match the reference setting. The comparator generates an error value, which is fed to the steering logic. A positive error will cause the steering logic to drive the left wheel forward causing the chassis to bear right. Similarly, a negative error will cause the right wheel to rotate, causing the chassis to steer left.

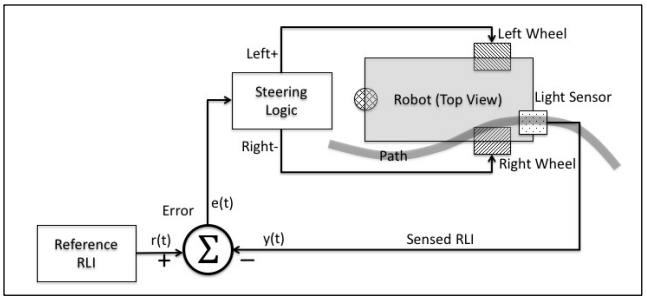


Figure 2: Bang-Bang Control implementation of a Left-edge follower

The control loop will continuously shuttle between these two states so that the average value of the error is effectively zero. The figure depicts a left-edge following robot. Implementing a right-edge following algorithm is just a matter of changing the logic of the steering logic to steer the opposite way. This simple algorithm is inefficient in that the robot tends to saw tooth its way along the path, effectively generating a non-optimal velocity vector in the direction of the path the robot is following. Considerable improvement can be achieved by in-lining a proportional control block between the comparator and the steering logic, such that the drive signal fed to each wheel is made proportional to the magnitude of the error detected by the feedback loop.

2.3 Proportional-Integral-Derivative (PID) Controller

Optimal control can be achieved by using three control terms to drive the feedback loop. The block diagram in Figure **3** shows a Proportional-Integral-Derivate controller that makes use of three concurrent stages to generate a feedback signal $u(t)$. The error value $e(t)$ is computed as the difference between a reference value $r(t)$ referred to as a setpoint, and the measured reflected light intensity $y(t)$. The control variable *u(t)* is described by the equation:

$$u(t) = K_p \times e(t) + K_i \times \int_0^t e(\tau)\, d\tau + K_d \times \frac{d\, e(t)}{dt}$$

As the error increases or decreases based on the movement of the robot, the value of u(t) tracks it within tight bounds.

The multipliers K_p, K_i, and K_d are non-negative coefficients of the proportional, integral and derivative terms respectively. The overall effect of the PID control loop is the sum of the individual terms, by *Superposition* principle. The proportional term simply feeds an amplified form of the error to the steering logic. The value of the coefficient K_p will affect the responsive of the system to error deviations. As K_p assumes a larger value, the loop tends to behave more like a simple bang-bang controller.

The integral term adds a memory component to the loop by helping it "remember" previous control signals. By maintain a running average, the loop keeps count of the residual error from previous iterations by generating a value dependent on the historic cumulative value of the error. A larger value of K_i has the effect of making the loop more sluggish to react. The integral component will become zero when the running average drops to zero. The derivative control is predictive, in that it tracks the rate of change over time of the error value. This term will grow as the difference between consecutive error readings widens. By choosing an appropriate value of K_d, the control loop can be made to track changes rather aggressively. The derivative component of the loop is also referred to as the "anticipatory control" term since it generates a control signal proportional to the rate of change of error.

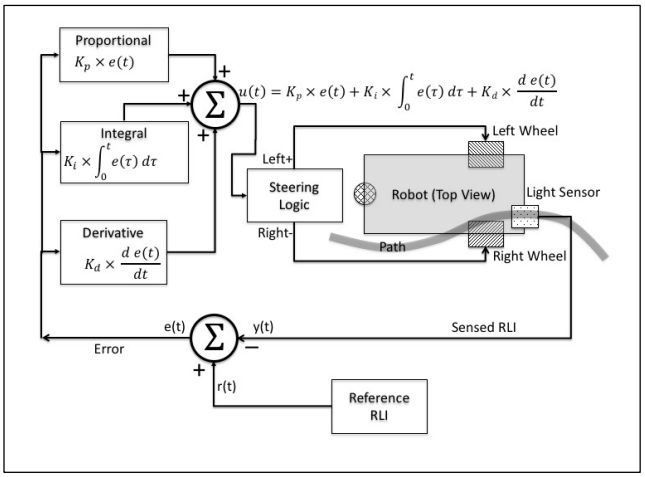


Figure 3: Proportional-Integral-Derivative controller

The values of K_p, K_i, and K_d are dependent on the electro-mechanical characteristics of the robot. The process of ascertaining these coefficients is termed *Tuning*, and this step is critical to the performance of the feedback control loop. Incorrectly chosen coefficients can render the loop unstable, sending it into oscillations limited only by the non-linearities of the system, and that could result in the degradation of the structural integrity of the robot. Manual tuning involves making tiny adjustments to the values of K_p, K_i, and K_d, and observing the robot as it tracks the path presented to it. A step input to the RLI can be used to study the manner in which the loop responds to changes in these coefficients. In this case, a step input involves making an abrupt change to the RLI by exposing the sensor to the brightest surface after the control loop has settled. Recording the *transient response* of the system is a way in which the effectiveness of the control loop can be quantified. The response of an electromechanical system to an abrupt change from an equilibrium or steady is called its transient response, and it can be measured in terms of the following:

- Rise time – time taken for the signal to go from 10% to 90% of its steady state value
- Overshoot – the amount by which the signal rises above its steady state value as a result of abrupt change

- Settling time – time for the output to settle within specified tolerance limits, measured from the time an abrupt change was applied
- Steady state error – the difference between the desired steady state output and the measured one

The objective of tuning is to make the control signal follow the variations in the measured value with fidelity. A system is said to be underdamped if the loop has a tendency to oscillate significantly for small changes in the controlled variable. On the other hand, a system is deemed overdamped if it takes considerable time to reach its steady state value. A highly responsive loop is said to be critically damped, allowing it to respond to variations in the controlled value is minimal lead or lag time.

The values of K_p, K_i, and K_d affect the aforementioned characteristics in their own unique ways. The effect of changes made to each of the PID coefficients on the response of the robot is summarized in the table below:

Coeff.	Rise time	Overshoot	Settling time	Steady State Error
K_p	Decreases	Increases	Negligible	Decreases
K_i	Decreases	Increases	Increases	Decreases
K_d	Negligible	Decreases	Increases	No change

Table 2: Influence of coefficients on the control loop

The process of manually tuning the PID controller is a tedious process and typically involves the following steps:

1. Design a bang-bang controller to study system characteristics
2. Introduce proportional control to improve the rise time
3. Add derivative control to reduce the overshoot
4. Add integral control to reduce the steady-state error
5. Iteratively adjust each of the gains K_p, K_i, and K_d until the overall response improves to being critically damped

3. Loop-Tuning using the Ziegler-Nichols Method

Several methods have been proposed for tuning the PID controller. A heuristic approach to tuning the loop was proposed by John Ziegler and Nathaniel Nichols [2]. The Ziegler-Nichols method involves setting the values of both the integral and derivative coefficients K_i and K_d to zero initially. The tuning process starts off by increasing the value of K_p until the system becomes underdamped, that is the point at which the loop starts to oscillate. The value of K_p at which this occurs is recorded as K_u, the ultimate proportional coefficient for this setup. The periodicity of the oscillations is now measured and recorded as T_u. Once these two values are available, the K_p, K_i, and K_d values can be computed for various combinations of control components as shown in the table below:

Control Type	K_p	K_i	K_d
P	$0.50\ K_u$	-	-
PI	$0.45\ K_u$	$0.54K_u/T_u$	-
PID	$0.60\ K_u$	$1.20K_u/T_u$	$3K_uT_u/40$

Table 3: Ziegler-Nichols method

One of the drawbacks of the Ziegler-Nichols method is that a slight variation of system parameters will render the coefficients invalid, pushing the steady state error beyond tolerance limits. It is seen that drift in motor and sensor characteristics forces frequent recalibration of the control coefficients as operating conditions change. Sensor measurements are affected by ambient light conditions as well as by power-drain from a continuously operating power source. Motor torque and power characteristics are also dependent on the power supply. This is where a computer program can serve to automate the tuning process as soon as a loss of performance is observed by the system. The implementation details of this self-tuning algorithm in explained in the following section. Advanced techniques for using Artificial Intelligence have been proposed for self-tuning[3]. The mechanism described here is based on a heuristic approach, which was first proposed in 1945 and used as a basis for manually tuning controllers.

3.1 Self-tuning using the Ziegler-Nichols Method

The self-tuning algorithm was implemented in embedded Java on a Lego EV3 robot running Linux installed with JRE 8. The classes used for this implementation were taken from the open source LeJOS (a Java based framework) library and modified to suit this experiment. The basic flow chart for the algorithm is shown in the figure below.

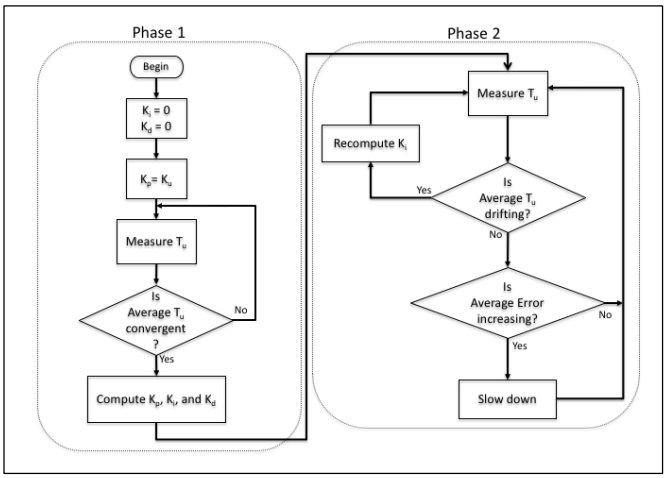


Figure 4: Tuning flow for Ziegler-Nichols method

The methodology for tuning a PID control loop based on Ziegler-Nichols method starts off with certain preliminary investigations. The first step is to gage the "*Ultimate period*" of oscillations for an underdamped control loop.

For the loop to be severely underdamped, the value of K_p is chosen such that the loop displays short rise times, high overshoots, and poor settling times. An important challenge was to peg data on the robot without overloading its processor. A framework for acquiring sensor and other data from the robot was implemented in Java. The on-board processor and storage space being limited on the robot, a client-server Data Acquisition System (DAS) was designed so that data collection was least intrusive.

3.2 Remote Data Acquisition System (DAS)

The DAS consists of a TCP service running on a laptop, listening on port 7777 (configurable). The robot, which is equipped with a WiFi dongle, runs the client side of the DAS system. The DAS framework is implemented on the client side as an object (type DASClient), which can be instantiated and used to send raw packets of time stamped information to the server on a need basis. The server makes use of a utility interpreter to write the contents of the packet to a file in comma separated variable format (CSV). The DAS server, which executes on the remote laptop, makes use of the built-in formatter provided by Java (java.util.Formatter) to generate CSV content. The raw packets can then be processed offline using an off-the-shelf spreadsheet/graphing software. An independent data acquisition system that off-loads storage and processing to a remote computer is least taxing on the limited CPU and storage capabilities of the robot controller. The objective of the control loop is to sense and track the left-edge of the black line. The reflected light intensity when the sensor beacon is directly on the edge was recorded as 50 normalized units. The loop would have to drive the steering logic in such a manner that the robot seeks to measure 50 units with its color sensor. The *setpoint* for the loop was therefore set at 50.

3.3 Phase 1 of the tuning process

The first phase of the tuning procedure involves finding the ultimate gain K_u of the loop. The ultimate gain of the loop is defined as the proportional coefficient of loop which pushes it into oscillation. To momentarily nullify the effect of the other control terms, the coefficients K_i and K_d are maintained at zero during this preliminary stage. K_p is then increased until the control loop oscillates consistently with stable periodicity. For the EV3 robot, this phenomenon was observed at a K_p value of 0.8, and the periodicity of the oscillation was measured to have an average value of 240 milliseconds (T_u), or a frequency of 4.166 Hz. Once Ku and Tu have been ascertained for the loop, the empirical values of K_p, K_i and K_d can be computed based on Ziegler-Nichols method as shown in Table **3**: Ziegler-Nichols method.

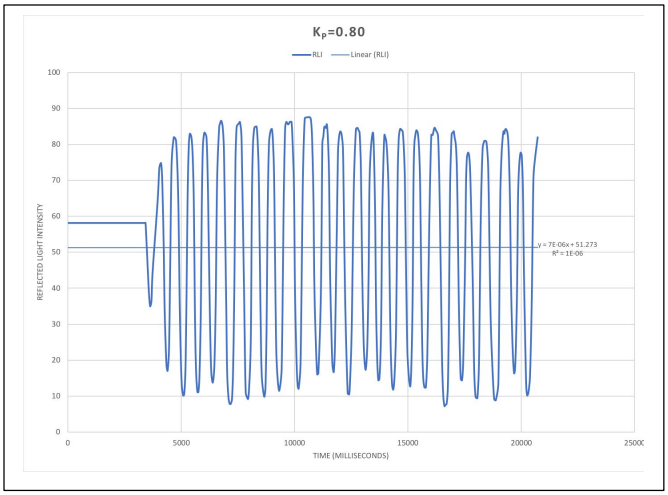


Figure 5: Ultimate gain data for the PID loop

The tuning rule is expected to make the PID loop less prone to external disturbances. The following value of the coefficients were selected using Ziegler-Nichols guidelines. The sensor data was remotely acquired from the robot using a TCP client-server.

Coefficient	**K_p**	**K_i**	**K_d**
Value	0.48	4.0	0.014

It was observed that the loop was considerably stabilized as a result of the aforementioned settings. The ultimate setting drove the loop to oscillate between extreme values of the detected error, which indicates that the mobile robot is oscillating considerably about its intended pathway. The error measurements gathered for the same track with the Ziegler-Nichols recommended coefficients is shown below.

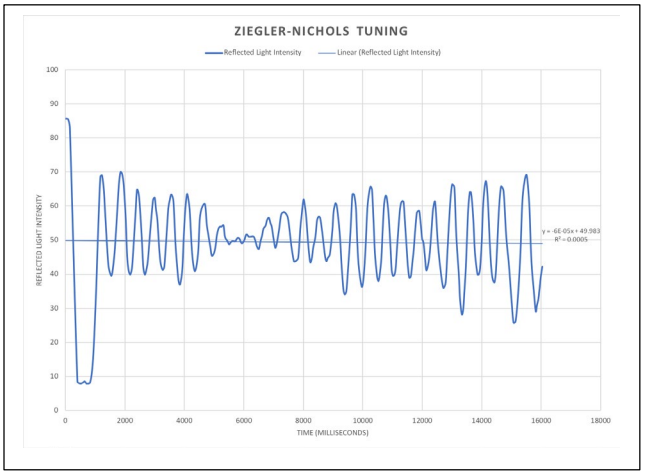


Figure 6: Error readings with Ziegler-Nichols settings

The loop initially works to acquire the path edge, causing the error to sharply saturate in the negative direction. But once the robot starts moving, it is observed that the RLI error is fairly limited within ± 20 units. A trendline based on linear regression is superimposed on the error readings and it is very much in agreement with the RLI *setvalue* of

50 units for the loop. The control loop maintains a queue of 256 error readings in a buffer, sampled at a rate of 20 milliseconds. The computed average of this buffer, multiplied by the coefficient K_i, forms the integral term of the control loop. This computed average error also serves as a reliable indicator of the goodness with which the robot tracks the path presented to it.

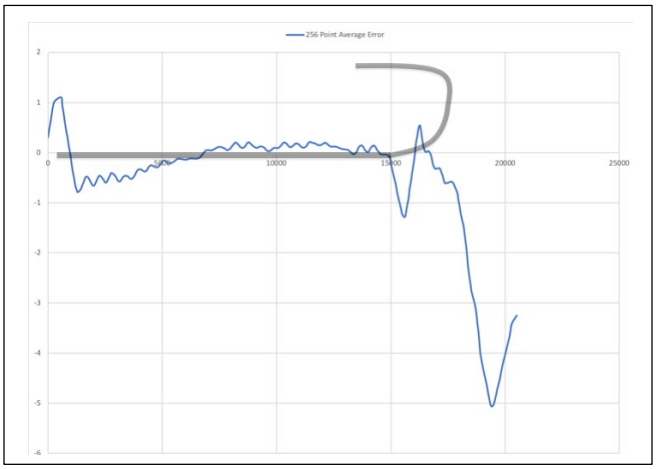


Figure 7: Average error on a curved path

The absolute value of the average increases as the robot loses track of the edge that it is meant to follow. The average error is shown in **Figure** 7, superimposed on a path that ends in a curve. The control system tries to keep the robot on track, but the curvature of the path is too sharp for the robot to accomplish the turn successfully at its operating velocity.

3.4 Phase 2 of the tuning process

Maintaining a precisely tuned PID control loop is not an easy task. As the power source of the robot depletes and the structural integrity of the snap-on pieces degrades, the loop characteristics change with it. Sensor readouts and motor torque characteristics change over time, causing the loop to react differently to external disturbances. The Phase 2 step of the tuning process is intended to keep the system continually tuned by varying K_i and K_d to control overshoot and settling times. By monitoring the peak error continually, the software can take a decision to Ki. The coefficient of the integral component loop can be increased or decreased to gain aggressive response to steering. An important observation is that a momentary but sustained increase in the half cycle time of the error signal is an indication that the robot is not succeeding in minimizing its error, in spite of its steering action. When this happens, it is also observable that the integral component, or the summation of successive errors build up. Assuming the structural integrity of the robot is still intact, this could only mean that the robot is in the process of negotiating a sharp turn in the path as shown in Figure 7. If the loop is not aggressive enough (short rise time), the robot may easily lose track of the pathway. The robot will also be unable to stay on track unless it has the capability to slow down as it drives into sharp turns. Adjusting Ki to follow the pathway more aggressively under these circumstances can achieve improved line following around sharp turns.

Continuous real-time tuning of the control loop was implemented in two ways (Figure 4). Speed control was used to negotiate sharp turns in the path, and coefficient tuning was used to compensate for structural changes:

- Speed control based on accumulated error average
- K_i tuning based on Tu variations

The first proposition trades off speed for control on a need basis. The closed loop control system will guide the robot along straight paths and paths with gentle curves in them. But negotiating sharp turns involves slowing down the robot enough to bring response times within manageable limits. Slowing down the robot has the effect of reducing the arrival rate of disturbances to the sensor system, giving the control loop more time to settle and track changes in the path. By slowing down the robot on a need-basis, triggered by accumulated error, the robot can traverse a given path at optimal speed.
The second proposition is based on the fact that a change in the structural integrity of the robot will result in its transfer function, and thereby its frequency response being altered. Such changes can be monitored by tracking the frequency of oscillations since it is reflected in the cycle time of those oscillations. Changes in overshoot (overcontrol) can be corrected by tuning K_i values under program control.

3.5 Improvements to the implementation

A simple line-following robot offers a plethora of data that can be acquired and analyzed to study principles of direct digital control. The long-term plan is to have robots cooperate to complete a task, under support from an overhead drone, capable of capturing images of the terrain. Through remote data acquisition, the image processing and the control loops can be driven by a powerful fixed computer that can communicate with the robots and the drone to get a task done. As mentioned at the onset, motion planning [6] in robots is generally a complex task involving the robot transitioning through multiple states. Typically, line following is just one of the states that the robot needs to assume as part of an array of steps needed to solve a specific problem. A software exit point implemented via an Action Listener from the line-following algorithm is a desirable feature to have. The average error can be used as a threshold to determine if the executing line-following algorithm needs to be terminated and have the robot transition to a different state. Another project that is being currently investigated is the use of Remote Method Invocation (RMI) to control the robot, so that the robot is basically reduced to a platform with sensors and motors being remotely driven by a controller that resides on a powerful desktop machine.

4. Conclusion

The project described in this paper was conducted as part of a semester long independent study into Cybernetics. It was observed that an empirical approach to advanced concepts in control theory based on experimentation, observation, and related inference was considerably more effective than the conventional approach of building a foundation in differential equations and Laplace transforms before approaching topics in Control theory [5]. The author is of the opinion that by restricting this introductory investigation entirely to discrete measurements, and by studying the effects of various modes of control, the curiosity to inquire into the theory of this exciting topic is considerably whetted among students. The implementation in Java is open-ended and presents opportunities for several extensions to the basic control framework. Control systems is an advanced topic typically offered as part of an engineering curriculum. An introductory course such as the one presented here gives non-engineering students an opportunity to delve into this interesting and useful topic. This basic theory is becoming indispensable to all intelligent mobile robots active today.

5. Acknowledgements

The implementation discussed in this paper was tested out exhaustively by the University Robotics team. Considerable effort was put into testing the set up and refining the code by Mr. Shaun Donohue, member of the Robotics team.

References:

[1] F. Golnaraghi, and B. C. Kuo, *Automatic Control Systems*, 10th ed., McGraw Hill, 2017.

[2] J. G. Ziegler, and N. B. Nichols, "*Optimal Settings for Automatic Controllers*," Transactions of the ASME, November 1942.

[3] F. Lin, R. D. Brandt, and G. Saikalis, "*Self-Tuning of PID Controllers by Adaptive Interaction*," in Proceedings of the Americal Control Conference, Chicago, Illinios, June 2000.

[4] Texas Instruments, "*AM 1808 ARM® Microprocessor*," SPRS653E –February 2010 – Revised March 2014.

[5] H. J. Sussmann, and J. C. Willems, "*300 Years of Optimal Control: From the Brachystochrone to the Maximum Principle*," IEEE Control Systems, vol 17, issue 3, pp. 32-44, June 1997.

[6] H. Choset, K. M. Lynch, S. Hutchinson, et al, "*Principles of Robot Motion: Theory, Algorithms and Implementaion*," The MIT Press, Cambridge Massachusetts, 2005.

MANY SERIAL PROGRAMS CAN BE EASILY PARALLELIZED TO RUN HUNDREDS OR THOUSANDS OF TIMES FASTER

Erik Wynters
Bloomsburg University of Pennsylvania
ewynters@bloomu.edu

ABSTRACT
For many problems, a parallel implementation run on a powerful graphics card's multi-core processor (GPU) runs hundreds or thousands of times faster than a serial implementation run on the computer's main central processing unit (CPU). This is demonstrated with an image processing example and a discrete facility location example. Using C++ AMP (Accelerated Massive Parallelism), the serial version is easily parallelized with small changes to the original program.

KEY WORDS
parallel speedup GPU C++ AMP

1. Introduction

Many serial programs can be easily parallelized with small code changes to run hundreds or thousands of times faster. This paper demonstrates that with two examples and shows that the method used in this paper is an easy way to introduce massive parallelism into a college course or research project.

The relatively new method used in this paper is C++ AMP, which stands for Accelerated Massive Parallelism. It is a set of language extensions to standard C++ that makes it easy to parallelize an algorithm that performs a task on each element of an array ([1]). The parallelized version will execute in parallel on a graphics card's processor (GPU) and C++ AMP handles the details of memory allocation and deallocation on the GPU and creation and execution of thousands or millions of threads that run in parallel on the GPU by creating a thread for each element of the array. Some older approaches to parallel programming on GPUs are CUDA ([2], [3]) and Thrust ([4]) that only worked with NVidia graphics cards. In contrast, C++ AMP can be used with different brands (e.g., NVidia GTX cards and AMD Radeon cards). C++ AMP was originally part of Microsoft's Visual Studio IDE to create programs on computers that use Microsoft's Windows operating system. It is an open standard however, and implementations for other operating systems are beginning to become available.

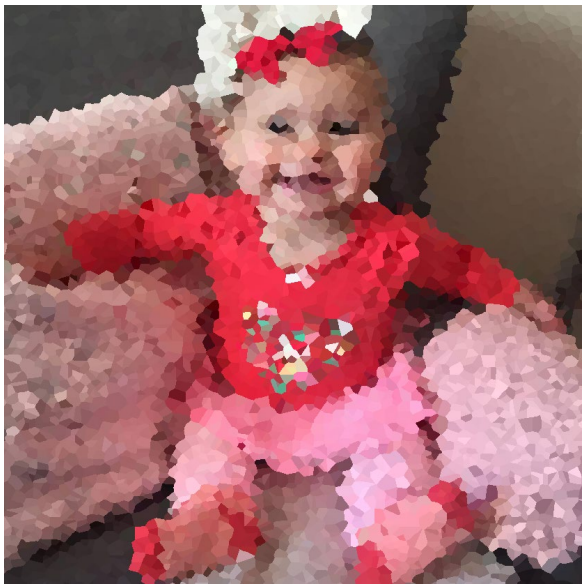
The C++ Standard Template Library (STL) ([5]) contains a generic/template container class called *vector* based on a dynamically-allocated array. AMP has a template class called *array* that is similar but is stored in graphics card memory. Its constructor can use a standard vector object to initialize the array, and after performing some task in parallel on the GPU, a simple assignment statement will wait for it to complete the task and then copy its contents back into a standard vector. The function *parallel_for_each* is used to specify what to do in parallel for each element of the array object. It uses a function passed as a parameter that is usually defined as an inline anonymous function using a lambda expression which was added to C++ in the C++11 standard (created in 2011). Lambda expressions automatically capture any data that's needed from the enclosing scope. The other popular high-level GPU programming library, Thrust, doesn't fully support lambda expressions, so complex tasks still require creating a function object (a.k.a. functor), which requires creating a class with data fields, a constructor and a method that overloads *operator( )*. The lambda expressions supported by C++ AMP make the code shorter and more explicit as to what task is being performed within the body of the parallel_for_each call.

2. Image Processing Example

This example processes all the pixels in a digital image serially and in parallel. Given a set of randomly generated sites (pixels chosen from the image), it uses them to create a "stained glass" mosaic effect as shown in Figure 1. For each pixel in the image it calculates the distance to each site and stores which one it is closest to. After computing the closest site for each pixel, each pixel uses the color of the closest site as its new color, creating a polygon with a single color around each site.

This is a pixelated version of the Voronoi diagram, which is a subdivision of the plane into regions where each region is associated with a specific site and consists of all points in the plane closer to that site than any other site. It has a long history in many areas of natural science, mathematics, and computer science ([6], [7], [8]). The discrete pixel-based approach used here has been created with more complex algorithms on GPUs ([9], [10], [11]), but are too complex to be covered in a class that's just introducing parallel processing on GPUs. The simple brute-force algorithm discussed here is easy to explain to students and can be easily parallelized using C++ AMP.

Figure 1. A Mosaic Image With 6000 Polygons



The specified number of sites were randomly generated based on the image size and stored in an STL vector called *sites*. The index of the closest site to each pixel is stored in a linearized (1D) representation of the (2D) array of pixels called *pixels*. The serial code for doing this is shown below in Figure 2. It uses the *for_each* template function from the STL's algorithm library along with a lambda expression (inline nameless function) to specify what to do for each index of the pixels vector in which the index of the closest site to each pixel will be stored. Both the serial and parallel version use the *[&, numSites]* syntax that indicates that by default items captured from the enclosing scope by the lambda expression will be captured by reference but *numSites* will be captured by value. A simple *point* struct with x and y fields and a helper function (*euclidean_distance*) to calculate the distance between two points is used in the code shown below in Figure 2. The image is square with both the number of pixel rows and the number of pixel columns stored in a named constant called RESOLUTION. INFINITY is also a named constant defined in C++.

Figure 2. Serial Code For Computing Closest Site

```
// serial version of computing closest site to each pixel
for_each(pixels.begin(), pixels.end(), [&, numSites] (int i) { // for each pixel
        int y = i / RESOLUTION;
        int x = i - y*RESOLUTION;
        point p(x, y);  // point whose coordinates depend on index in pixels vector

        float min_dist = INFINITY;  // initialize minimum distance
        int min_nbr = 0;            // and index of site that achieves it
        for (int i = 0; i < numSites; i++) { // for each site
                point s = sites[i];
                float d = euclidean_distance(p, s);
                if (d < min_dist) {  // if current site is closer to current pixel
                        min_dist = d;     // than best found so far,
                        min_nbr = i;      // update distance and site index
                }
        }
        pixels[i] = min_nbr;  // save index of closest site to current pixel
});
```

The code above requires only minor changes to make it execute in parallel on a GPU as shown in Figure 3. To do something for each pixel in parallel, the STL vector *pixels* used in Figure 2 is used to create an AMP *array* object called *d_pixels* on the graphics device (d_ for device). Similarly, an AMP *array* object called *d_sites* is created from the vector *sites* used above on the CPU. After creating those two array objects on the graphics device, STL's *for_each* function is changed to AMP's *parallel_for_each* function to create a thread for each pixel and perform the task specified by the lambda expression in parallel for each element of the array *d_pixels*. Other minor changes made to the STL for_each function call are that the extent property of the d_pixels array is used instead of the begin and end iterators of the pixels vector to specify how many threads to create and instead of having an integer index into the vector as a parameter it has a similar parameter of the index class, where <1> means it's 1-dimensional. As shown below in Figure 3, it can be used as is as an "index" into an array object but it also can be converted to an integer by adding [0] to its name, which means get its first component (which is the only one it has a 1D index object). The only other change made to convert from for_each to parallel_for_each was the *restrict(amp)* directive before the body of the lambda expression, which is necessary for it to compile for execution on the GPU. As the directive suggests, there are some restrictions on code that's executed on GPUs, e.g., not all primitive types and mathematical functions can be used. But as shown below

in Figure 3, common types like int and float can and AMP has its own version of most mathematical functions that can be used with the restrict(amp) directive. An example of that is that the helper function *euclidean_distance* uses the standard *sqrtf* function in the serial version to calculate a single-precision floating point square root, while the corresponding helper function in the parallel version uses the restrict(amp) directive and uses the function *__dp_d3d_sqrtf* from AMP's fast math library.

Figure 3. Parallel Code For Computing Closest Site

```
// create GPU array object copies of CPU vectors
array<point, 1> d_sites(sites.size(), sites.begin(), sites.end());
array<int, 1> d_pixels(pixels.size(), pixels.begin(), pixels.end());

// parallel version of computing closest site to each pixel
parallel_for_each(d_pixels.extent, // for each pixel in parallel
  [&, numSites](index<1> idx) restrict(amp) { // compile to run on GPU
        int y = idx[0] / RESOLUTION;
        int x = idx[0] - y * RESOLUTION;
        point p(x, y); // point whose coordinates depend on index in d_pixels array

        float min_dist = INFINITY; // initialize minimum distance
        int min_nbr = 0;           // and index of site that achieves it
        for (int j = 0; j < numSites; j++) { // for each site
                point s = d_sites[j];
                float d = euclidean_distance(p, s);
                if (d < min_dist) {  // if current site is closer to current pixel
                        min_dist = d;    // than best found so far,
                        min_nbr = j;     // update distance and site index
                }
        }
        d_pixels[idx] = min_nbr; // save index of closest site to current pixel
});
pixels = d_pixels; // wait for GPU code to complete and copy results back to CPU
```

3. Image Processing Results

Different numbers of sites were randomly generated and the closest site to each pixel in an image of resolution 1024 x 1024 was calculated using a 7th-generation 2.71 GHz Intel Core i5 CPU and an NVIDIA GTX 1080 Ti GPU with 3584 cores. Microsoft's Visual Studio 2017 C++ compiler was used to create both executables with the O2 flag set to optimize code for maximum speed and with the fast math library used for single-precision floating point calculations on both the CPU and GPU.

As shown in Table 1, this algorithm ran about five hundred times faster in parallel on the platform used for testing (run times are in seconds). Both versions were compiled and run in release mode without debugging for accurate run times.

Table 1. Image Processing Run Times

Sites	CPU	GPU	Speedup
16000	25.956	0.055	471.9
32000	51.497	0.107	481.3
64000	105.737	0.205	515.8
128000	211.651	0.390	542.7

4. Facility Location Example

Facility location has been a topic discussed in business, operations research, and computational geometry for many years ([12], [13], [14], [15]). As the name suggests, it involves choosing a location for a new facility of some kind with a particular objective. If it's a factory, the objective might be to place it so the sum of the distances to all the warehouses or distribution sites is minimized. Another objective could be to minimize the maximum distance to any destination or customer, i.e., find the center of the smallest circle that contains all the destination sites.

The facility location example used in this paper is concerned with where to place a communications tower, e.g., a radio station transmitter or a cell phone tower. A major factor to be considered in this situation is called free space loss ([16]) (the loss in signal strength through free space). The signal loss is more important for devices further away from the tower. Minimizing the total of all distances would give closer devices the same importance as ones further away, but that's not appropriate because the closer ones have stronger signals and don't have much signal loss. Minimizing the maximum distance might be better, but doesn't take into account where all the devices far away from the tower are. It usually just minimizes the distance to the two devices farthest away from the tower

by finding a location halfway between them (the average of their positions). But there could be thousands of other devices far away from those two, which could have much better signal strength if the transmitter was closer to them.

Given a set of existing sites that should be effectively communicated with, the goal in this example is to choose a location for a new facility that emphasizes all sites that are further away and have more signal loss. That was implemented by scaling the distances to fairly low numbers and using them as exponents (with base 2) as the contribution to a weighted sum over all sites for each potential facility location. In this paper, a brute-force approach for a discrete version of the problem was used. Using a fixed number of rows and columns for potential locations for the transmitter, the best position for the transmitter was calculated.

The serial and parallel programs are similar to the previous example, but instead of finding the minimum distance to an existing site for each row and column, the total weighted sum of distances to all sites was calculated for each row and column.

5. Facility Location Results

The same CPU, GPU, compiler, and compiler settings mentioned in the previous results section were used again. The locations of the existing sites and potential location for the new communication facility were represented by a discrete 1000 x 1000 2D array. Different numbers of existing sites to communicate with were randomly generated, and the sum of the exponential scaled distances to all the sites was calculated for each potential facility location in the 2D array.

The code for both the serial and parallel versions are very similar to the previous example except for adding division and the powf function to the distance calculations and the sum being calculated over all the sites for each potential location index rather than the minimum.

As shown in Table 2, the parallel version had bigger speedups with this problem than the previous example. Instead of being around 500 times faster, the parallel version was around 5000 - 7000 times faster for this example. For the last row in Table 2, the CPU time was about 116 minutes while the GPU time was less than half a second.

Table 2. Facility Location Run Times

Sites	CPU	GPU	Speedup
16000	317.644	0.064	4963.2
32000	635.154	0.114	5571.5
64000	1269.038	0.206	6160.4
128000	2539.996	0.365	6958.9

6. Conclusion

This paper shows that small code changes can make serial CPU programs run hundreds or thousands of times faster in parallel on a powerful GPU using C++ AMP. This is an easy way to introduce parallel processing on GPUs into a college class or research project.

It might seem unrealistic that the speedups in Table 2 exceed the number of cores in the GPU, but GPU cores are not equivalent to CPU cores. They have different strengths and weaknesses. Many factors affect speedups that can be achieved with parallel processing on GPUs. They include characteristics like inherent parallelism, computational intensity, data transfer needs, and memory access patterns. The experiments described in this paper demonstrate well the potential of parallel processing on GPUs using the hardware and software described in this paper. The second example had more parallelism since there wasn't a conditional statement (GPU threads are not independent) and it had more single-precision floating point calculations by adding the scaling and the powf function to each distance calculation used in the first example. The greater parallelism and those additional single-precision floating point calculations are what GPUs are best at and why the speedups were about 10 - 14 times what they were in the first example.

Although both of the examples in this paper took less than half a second for the largest instances on the GPU used, bigger instances or other problems that take more time could be much slower. If a larger instance or a harder problem took a whole day (24 hours) to solve it in parallel with the biggest speedup achieved in Table 2, it would take over 19 years to solve it serially. That shows how parallel speedups can make it practical to solve some problem instances that wouldn't be practical to solve serially.

References:

[1] K. Gregory & A. Miller, *C++ Amp: Accelerated Massive Parallelism With Microsoft Visual C++* (Redmond, WA: Microsoft Press, 2012).

[2] J. Nickolls, I. Buck, M. Garland, & K. Skadron, Scalable parallel programming with CUDA, *ACM Queue, 6(*2), 2008, 40-53.

[3] R. Farber, *CUDA application design and development* (Waltham, MA: Morgan Kaufmann, 2011).

[4] J. Hoberock & N. Bell, *Thrust: a productivity-oriented library for CUDA: GPU Computing Gems Jade Edition* (Boston, MA: Morgan Kaufmann, 2011).

[5] N. M. Josuttis, *The C++ standard library: a tutorial and reference, 2nd ed.* (Upper Saddle River, NJ: Addison-Wesley, 2012).

[6] F. Aurenhammer, Voronoi diagrams: a survey of a fundamental data structure, *ACM Computing Surveys*, *23* (3), 1991, 345-405.
[7] J. O'Rourke, *Computational Geometry in C* (Cambridge, MA: Cambridge University Press, 1994).
[8] M. de Berg, M. van Kreveld, M. Overmars, & O. Schwarzkopf, *Computational Geometry: Algorithms and Applications, 2nd ed.* (Berlin: Springer-Verlag, 2000).
[9] K. Hoff, J. Keyser, M. Lin, D. Manocha, & T. Culver, Fast computation of generalized Voronoi diagrams using graphics hardware, *Proc. ACM SIGGRAPH*, New York, NY, 1999, 277-286.
[10] G. Rong, & T. Tan, Jump flooding in GPU with applications to Voronoi diagram and distance transform, *Proc. ACM Symp. Interactive 3D Graphics and Games*, New York, NY, 2006, 109-116.
[11] T. Cao, T. Tang, A. Mohammed, & T. Tan, Parallel banding algorithm to compute exact distance transform with the GPU, *Proc. ACM Symp. Interactive 3D Graphics and Games*, New York, NY, 2010, 83-90.
[12] W. V. Gehrlein, & M. Pasic, Centroid – a widely misunderstood concept in facility location problems, *International Journal of Industrial Engineering Theory, Applications and Practice*, *16*(2), 2009, 99-107.
[13] P. Bose, A. Maheshwari, & P. Morin, Fast approximation for sums of distances, clustering and the Fermat-Weber problem, *Computational Geometry: Theory and Applications*, *24*(3), 2003, 135-146.
[14] N. Megiddo, The weighted Euclidean 1-center problem, *Mathematics of Operations Research*, *8*(4), 1983, 498-504.
[15] G. T. Toussaint, Computing largest empty circles with location constraints, *International Journal of Computer and Information Sciences*, *12*(5), 1983, 347-358.
[16] C. A. Balanis, *Antenna Theory: Analysis and Design, Third Edition (*Hoboken, NJ: Wiley, 2005).

REFEREED GRADUATE PAPERS

A COMPARATIVE STUDY OF DATA MINING TECHNIQUES USED TO TEST PREDICTIVE ACCURACY OF AUTISM SPECTRUM DISORDER SCREENING PROCESS

Jaime Hutchinson, Ian Schauer and Raed Seetan
Computer Science department
Slippery Rock University
{jxh1106, ids8249, raed.seetan}@sru.edu

ABSTRACT
Introduction: Autism Spectrum Disorder (ASD) is a developmental disorder characterized by impediments in ability to interact with and relate to others, typically noticed in early childhood. It is recommended that all children be screened for ASD. There is a high economic impact of autism, and the increasing number of documented ASD cases calls for the development of a more effective method of screening. It is important to diagnose at an early age so that affected children can get adequate treatment. Methodology: The WEKA tool was utilized to test the performance of the Naïve Bayes and J48 Decision Tree algorithms. Results: The Naïve Bayes analysis on the AQ-10 Child dataset yielded the highest precision (0.963), sensitivity (0.972), and specificity (0.954). Conclusion: The Naïve Bayes algorithm was the better option for predicting a patient's need for a referral to a specialist.

KEYWORDS
Decision Tree Classification, Bayesian Classification, Autism-Spectrum Quotient

1. Introduction

Autism Spectrum Disorder (ASD) is estimated by the Centers for Disease Control and Prevention to affect nearly 1 in every 59 children across all racial, ethnic and socioeconomic groups, and is four times more common in boys than in girls [1]. Children as young as one-year old exhibit signs of autism, typically in the form of social orienting impairments. It has been found that at 14 months old, those with ASD underperformed on all variables (social effect, social cognition: joint attention, communication: regulatory intentionality, communication form, and play) compared to those without ASD. [3]. Unfortunately, the average age of diagnosis is not until four years old. This is after an average of two years of parents seeking help and taking their child to see three different medical professionals [4].

Currently, the use of existing screening methods remains questionable due to a lack of data and evidence supporting their use. It is important to still work to collect and analyze data for screening instruments because there is a clear benefit of early behavioral intervention for children diagnosed with ASD. There is evidence that outcomes for children with autism can be significantly enhanced by early intensive intervention [4]. This makes early diagnosis crucial to improving not only the social skills of affected children but their quality of life and that of their families. The time between parents noticing their child having social difficulties and being diagnosed and beginning appropriate treatment is a time that comes with high levels of frustration.

The Autism-Spectrum Quotient (AQ) is a self-report measure of autistic traits. It is one of many screening tools that can be used to help detect ASD. Multiple studies, including systematic review, have confirmed that the 10 item AQ (AQ-10) has adequate validity to be used in health surveys as a measure of autistic traits [5] [6]. In this study, we will be reviewing data from the AQ-10 used on children and adolescents. The types of Data Mining Techniques that can be used to help with diagnosing ASD include machine learning, supervised learning, classification algorithms. The two algorithms to be investigated are the Naïve Bayes and J48 classification algorithms.

The performances of the Naïve Bayes and J48 Decision Tree algorithms will be investigated. The precision, specificity, and sensitivity will be measures used to determine which algorithm is the most dependable and accurate in predicting the presence of ASD in individuals who take part in answering the questions in the AQ-10 using datasets published on the UCI Machine Learning Repository. For this experiment, the data will be filtered so that only the data from the answers to the AQ-10 questionnaire administered to participants are analyzed.

This paper includes multiple sections including related work, methodologies, and the conclusion. In the relates work section you will find information about the related studies we explored to gain an understanding of previous findings on this topic. In the methodology section, we describe the tools and resources used for our experiment. Finally, the conclusion includes the results of our experiment.

2. Related Work

Aside from the AQ-10, there are many other screening tools that are regularly used for ASD. The EDUTEA Questionnaire is a tool that can be used to screen for ASD in a school setting. One study aimed to determine the sensitivity, specificity and positive predictive value of this survey in order to conclude about its usefulness for ASD screening in schools. Using the ADI-R diagnostic algorithm as well as exploratory factor analysis, the study found that this survey had high sensitivity (87%), specificity (91.2%) and positive predictive value (0.87), justifying its use in schools [7].

Another study examined the detection of autism using machine learning on home videos. Several models (including decision trees, support vector machine [SVM], logistic regression (LR), radial kernel, and linear SVM) assessed 30 behavioral features of the young children with autism in 116 short home videos. A sparse 5-feature LR classifier (LR5) yielded the highest accuracy in classifying autism across all the age groups tested [8].

A third study measured the diagnostic accuracy of International Epidemiology Network (INCLEN) Diagnostic Tool for Autism Spectrum Disorder (INDT-ASD) in Comparison with Diagnostic and Statistical Manual of Mental Disorders-5 (DSM-5). By using the INDT-ASD, 118 children aged 2-9 years who had symptoms suggestive of ASD could be compared to the child's DSM-V diagnosis of ASD. This test was found to have a sensitivity of 100% and specificity of 75%. This diagnostic tool was confirmed to be clinically useful in the diagnosis of ASD [9].

In a recent study in 2018, Thabtah used Naïve Bayes and Logistic Regression algorithms to show the significance of the features collected relating to the screening of ASD. Classifiers generated by the machine learning algorithms showed high sensitivity, specificity and accuracy rates. These results strongly indicate that we should replace existing scoring functions and handcrafted rules within screening tools with more intelligent machine learning models [10].

The analysis of these two algorithms, Naïve Bayes and J48 Decision Tree, have not been studied against each other on results of the AQ-10 in a prior experiment.

3. Methodology

3.1 Datasets

The datasets utilized for this study are two similar classification datasets from Fadi Fayez Thabtah at the Manukau Institute of Technology in Auckland, New Zealand, in the Department of Digital Technology [11] [12]. These particular datasets were utilized because they are both classification data sets with 21 attributes which represent results from the Autism Spectrum Quotient survey (AQ-10). The similarities in the dataset structures allow for the algorithms to be properly compared rather than the datasets or surveys they represent. The "Child" dataset has 292 instances and the "Adolescent" dataset has 104 instances. The AQ-10 records ten behavioral features and ten individual characteristics that have proven to be effective in detecting the ASD cases from controls in behavior science. The children surveyed are either ages 4-11 (child) or 12-15 (adolescent), and the questions are tailored to either age group. In the surveys, answers are classified as 0 or 1 (binary attributes), indicating the presence or absence of behavioral features. A score of 6-10 on this assessment justifies referral to a behavioral specialist [13][14]. Below are the statements posed on the questionnaire. Possible answers to these statements include "Definitely Agree", "Slightly Agree", "Slightly Disagree", and "Definitely Disagree". Scoring is based on agreeing or disagreeing; the degree is negligible. The statements are shown in Figures 1 and 2.

1	S/he often notices small sounds when others do not
2	S/he usually concentrates more on the whole picture, rather than the small details
3	In a social group, s/he can easily keep track of several different people's conversations
4	S/he finds it easy to go back and forth between different activities
5	S/he doesn't know how to keep a conversation going with his/her peers
6	S/he is good at social chit-chat
7	When s/he is read a story, s/he finds it difficult to work out the character's intentions or feelings
8	When s/he was in preschool, s/he used to enjoy playing games involving pretending with other children
9	S/he finds it easy to work out what someone is thinking or feeling just by looking at their face
10	S/he finds it hard to make new friends

Figure 1: AQ-10 (Child) Statements

1	S/he notices patterns in things all the time
2	S/he usually concentrates more on the whole picture, rather than the small details
3	In a social group, s/he can easily keep track of several different people's conversations
4	If there is an interruption, s/he can switch back to what s/he was doing very quickly
5	S/he frequently finds that s/he doesn't know how to keep a conversation going
6	S/he is good at social chit-chat
7	When s/he was younger, s/he used to enjoy playing games involving pretending with other children
8	S/he finds it difficult to imagine what it would be like to be someone else
9	S/he finds social situations easy
10	S/he finds it hard to make new friends

Figure 2: AQ-10 (Adolescent) Statements

3.2 Tools

The WEKA tool was utilized to experiment with classification algorithms. These algorithms include the

Naïve Bayes algorithm the J48 decision tree algorithm. The algorithms were applied to the data sets separately so that the individual outcomes could be compared. Bayes' classification method was chosen because it agrees with the objective of predicting the probability that a tuple belongs to either of the classes in question, allowing for the best predictors of the AQ-10 survey to be determined.

The Naïve Bayes algorithm comes from the Bayesian Network which predicts outcomes based on probabilities. This specific algorithm is based on two assumptions. The first assumption is that attributes are equally important. The second assumption is that attributes are statistically independent. This means that knowing the value of one attribute says nothing about the value of another. It will be helpful in determining if an individual possesses characteristics that are predictive of having ASD [15].

A decision tree is like a flowchart, where the internal nodes are the tests, the branches are the outcomes of the tests and the leaves are the class labels which are predicted by the outcomes of the tests. Decision trees select the attributes that best split tuples into the different classes. This may be helpful in finding the attributes that best predict ASD diagnosis [15].

The reasoning behind choosing these two types of techniques as possible ways to analyze this data is because they are both well-known techniques used for classification. The learning and classification steps of decision tree induction are fast and straightforward [15]. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases [15].

3.3 Preprocessing

Using the WEKA tool, preliminary exploration of the datasets being studied could be accomplished. First, using the OneR algorithm provided a baseline for which the other results could be compared. Below are examples of early findings that were discovered for each dataset.

Taking a first look at the "Autistic Spectrum Disorder Screening Data for Children" dataset, the distribution of the class attributes was examined. The dataset has 292 instances and there are 151 instances in the class of "No" and 141 instances in the class for "Yes". This means that this dataset is generally balanced. The dataset was analysed with the OneR algorithm which makes a rule that is based on each attribute, then chooses the single attribute with the minimum error for the prediction of the data set. It was found that the A4_Score had the lowest error. Using just the A4_Score predicted 228 of 292 (78%) instances correctly. This tells us the answer a subject has for the fourth question "If there is an interruption, I can switch back to what I was doing very quickly" may be a key determinant on how the subject's results are classified.

Like the previous dataset, the "Autistic Spectrum Disorder Screening Data for Adolescent" dataset's class attributes distribution was examined. The dataset has 104 instances and there are 63 instances in the class of "Yes", and 41 instances in the class of "No". This dataset is also balanced. There is not a "rare" class so there still should not be any significant performance degradation. The dataset was analysed with the OneR algorithm. It was found that the A5_Score had the lowest error. Using just the A5_Score predicted 71 of 104 instances correctly (68%). This tells us the answer a subject has for the fifth question "I find it easy to 'read between the lines' when someone is talking to me" may also be a key determinant on how the subject's results are classified.

3.4 Experiments

3.4.1 J 48 Decision Tree Results

Using the J48 Decision Tree algorithm on the "Autistic Spectrum Disorder Screening Data for Children" dataset, 268 of 292 (92.8%) instances were classified correctly. Sensitivity is the probability that a test will indicate 'Yes' among those with ASD. The sensitivity for this experiment was 0.901. Specificity is the fraction of those without ASD who will have a negative test result. The specificity of this experiment was 0.934. The experiment resulted in a Tree size of 31 with 16 leaves, taking 0.02 seconds to run. The pruned tree for this experiment can be found in Figure 3.

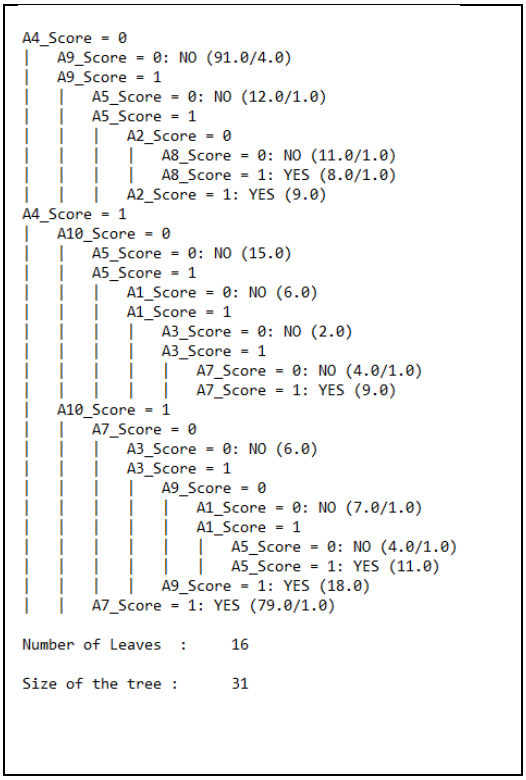


Figure 3: J48 pruned tree for Children ASD Dataset

Table 1: Confusion Matrix for Child Dataset – J48 Decision Tree

	No (Predicted)	Yes (Predicted)	Total
NO (Actual)	141	10	151
YES (Actual)	14	127	141
Total	155	137	292

Using the J48 Decision Tree algorithm on the "Autistic Spectrum Disorder Screening Data for Adolescent" dataset, 85 of 104 (81.7%) instances were classified correctly. Sensitivity is the probability that a test will indicate 'Yes' among those with ASD. The sensitivity for this experiment was 0.873. Specificity is the fraction of those without ASD who will have a negative test result. The specificity of this experiment was 0.731. The experiment resulted in a Tree size of 19 with 10 leaves, taking 0.00 seconds to run. The pruned tree for this experiment can be found in Figure 4.

```
A5_Score = 0
|   A10_Score = 0: NO (14.0)
|   A10_Score = 1
|   |   A7_Score = 0: NO (9.0/1.0)
|   |   A7_Score = 1: YES (3.0)
A5_Score = 1
|   A3_Score = 0
|   |   A10_Score = 0: NO (8.0)
|   |   A10_Score = 1
|   |   |   A4_Score = 0: NO (4.0/1.0)
|   |   |   A4_Score = 1
|   |   |   |   A8_Score = 0: NO (4.0/1.0)
|   |   |   |   A8_Score = 1: YES (8.0)
|   A3_Score = 1
|   |   A6_Score = 0
|   |   |   A8_Score = 0: NO (3.0)
|   |   |   A8_Score = 1: YES (3.0)
|   |   A6_Score = 1: YES (48.0/2.0)

Number of Leaves  :     10

Size of the tree :      19
```

Figure 4: J48 pruned tree for Adolescent ASD Dataset

Table 2: Confusion Matrix for Adolescent Dataset – J48 Decision Tree

	No (Predicted)	Yes (Predicted)	Total
NO (Actual)	30	11	41
YES (Actual)	8	55	63
Total	38	66	104

3.4.2 Naïve Bayes Results

Using the Naïve Bayes algorithm on the "Autistic Spectrum Disorder Screening Data for Children" dataset, 281 of 292 (96.2%) instances were classified correctly. Sensitivity is the probability that a test will indicate 'Yes' among those with ASD. The sensitivity for this experiment was 0.972. Specificity is the fraction of those without ASD who will have a negative test result. The specificity of this experiment was 0.954 The experiment took 0.00 seconds to run.

Table 3: Confusion Matrix for Child Dataset - Naïve Bayes Classifier

	No (Predicted)	Yes (Predicted)	Total
NO (Actual)	144	7	151
YES (Actual)	4	137	141
Total	148	144	292

Using the Naïve Bayes algorithm on the "Autistic Spectrum Disorder Screening Data for Adolescent" dataset, 96 of 104 (92.3%) instances were classified correctly. Sensitivity is the probability that a test will indicate 'Yes' among those with ASD. The sensitivity for this experiment was 0.968. Specificity is the fraction of those without ASD who will have a negative test result. The specificity of this experiment was 0.854. The experiment took 0.00 seconds to run.

Table 4: Confusion Matrix for Adolescent Dataset - Naïve Bayes Classifier

	No (Predicted)	Yes (Predicted)	Total
NO (Actual)	35	6	41
YES (Actual)	2	61	63
Total	37	67	104

Both techniques were analyzed for both the child and adolescent datasets. The run times for all experiments were virtually zero seconds, rendering them negligible for comparison. For each data set, the J48 Decision Tree algorithm had a higher precision for predicting "Yes" than it did "No". In contrast, for each dataset, the Naïve Bayes algorithm had a higher precision for predicting "No" than it did "Yes". The performance of the algorithms for each dataset is displayed in Figures 5-7. Figure 5 displays the Sensitivity of each experiment. Figure 6 displays the Specificity of each experiment. Figure 7 displays the precision of each experiment.

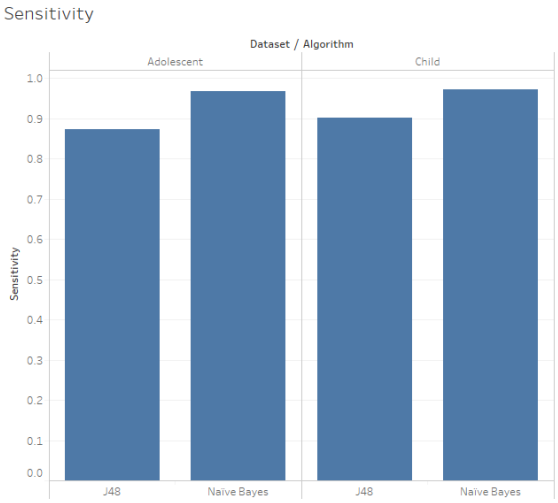


Figure 5: Accuracy of Naïve Bayes and J48 algorithms (Sensitivity)

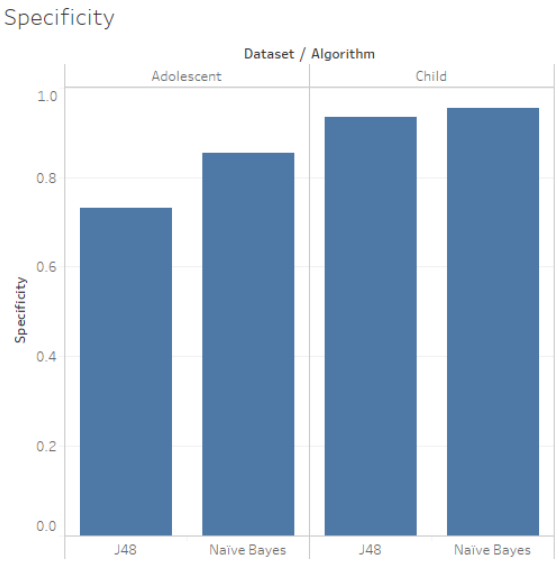


Figure 6: Accuracy of Naïve Bayes and J48 algorithms (Specificity)

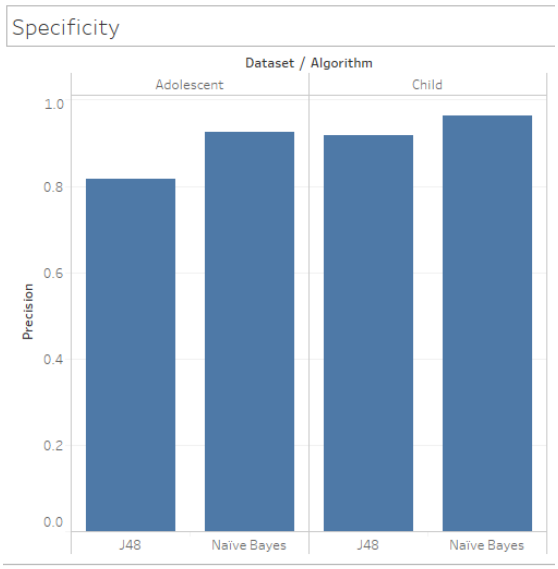


Figure 7: Accuracy of Naïve Bayes and J48 algorithms (Precision)

For both datasets, the answer data to the AQ-10 was used to classify each instance as a Yes or No for the presence of ASD. For both the Naïve Bayes and J48 decision tree algorithms, 10-fold cross validation was utilized for the test mode. This means the original sample is randomly split into 10 equal subsamples. Of the 10 subsamples, 9 are used for training data and 1 is used as validation data. The process is repeated 10 times so each sample is used once for validation. The 10 results are then averaged to produce a single estimation.

4. Conclusion

The objective of the study was to use data mining techniques to determine which classification algorithm, the Naïve Bayes or the J48 Decision Tree algorithm, can best be used to help with diagnosing ASD. We used the WEKA tool throughout the experiment to compare these two algorithms.

The highest precision (0.963), sensitivity (0.972), and specificity (0.954) were observed during the Naïve Bayes analysis on the AQ-10 Child dataset. Of the 292 instances in the data set, 281 instances were correctly classified and 11 incorrectly classified instances, accounting for its high accuracy of 96.3%. Naïve Bayes testing of the AQ-10 Adolescent dataset also yielded high precision (0.924), sensitivity (0.968), and specificity (0.854). With 96 correctly classified instances and 8 incorrectly classified instances out of 104, the Naïve Bayes algorithm had a 92.3% accuracy for the Adolescent dataset. The results of our experiment suggest that the Naïve Bayes algorithm is a more reliable technique for the task of predicting ASD using an AQ-10 screening questionnaire, as it had higher values of all measures for both datasets.

Ultimately, the Naïve Bayes algorithm was determined to be the best option for predicting a patient's need for a referral to see a specialist based on the AQ-10 questionnaire for both adolescents and children. The Naïve Bayes technique had better precision, sensitivity, and specificity than the J48 tree algorithm. The AQ-10's validity would be worth investigating in future research. When the Naïve Bayes algorithm was used to classify the data set using on the demographics of the participants instead of the answers to the AQ-10, Similar results were obtained in terms of precision. Comparing these results to the EDUTEA study, it appears using the AQ-10 questionnaire could be useful in ASD screening protocols in schools.

References:

[1] “Autism Spectrum Disorder.” Centers for Disease Control and Prevention. 2015. Accessed October 3, 2018. https://www.cdc.gov/ncbddd/autism/screening.html

[2] “Autism Spectrum Disorder.” National Institute of Mental Health. 2018. Accessed September 29, 2018. https://www.nimh.nih.gov/health/topics/autism-spectrum-disorders-asd/index.shtml

[3] Rebecca J. Landa, PhD; Katherine C. Holman, PhD; Elizabeth Garrett-Mayer, PhD, Social and Communication Development in Toddlers With Early and Later Diagnosis of Autism Spectrum Disorders, JAMA Psychiatry 2007 Retrieved from https://jamanetwork.com/journals/jamapsychiatry/fullarticle/482365

[4] Susan E Bryson, PhD, Sally J Rogers, PhD, Eric Fombonne, MD, Autism Spectrum Disorders: Early Detection, Intervention, Education, and Psychopharmacological Management, The Canadian Journal of Psychiatry 2003 Retrieved from: https://journals.sagepub.com/doi/abs/10.1177/070674370304800802

[5] Emily Ruzich, Carrie Allison, Paula Smith, Peter Watson, Bonnie Auyeung, Howard Ring, SimonBaron-Cohen, Measuring autistic traits in the general population: a systematic review of the Autism-Spectrum Quotient (AQ) in a nonclinical population sample of 6,900 typical adult males and females, Molecular Autism, 2015, Volume 6, Number 1, Page 1, Retrieved from: https://molecularautism.biomedcentral.com/articles/10.1186/2040-2392-6-2

[6] Lundin, A., Kosidou, K. & Dalman, C. J Measuring Autism Traits in the Adult General Population with the Brief Autism-Spectrum Quotient, AQ-10: Findings from the Stockholm Public Health Cohort (2018). Retrieved from: https://doi.org/10.1007/s10803-018-3749-9

[7] Morales-Hidalgo, P., Hernández-Martínez, C., Voltas, N. and Canals, J. (2017) "EDUTEA: A DSM-5 teacher screening questionnaire for autism spectrum disorder and social pragmatic communication disorder." International Journal of Clinical and Health Psychology 17(3) 269-297. Accessed November 4, 2018. Doi: 10.1016/j.ijchp.2017.05.002

[8] Tariq, Q., Daniels, J., Schwartz, J.N., Washington, P., Kalantarian, H. and Wall, D.P. "Mobile detection of autism through machine learning on home video: A development and prospective validation study." PLoS Medicine, 15 (11): e1002705. Accessed November 11, 2018. Doi: 10.1371/journal.pmed.1002705

[9] Vats, P., Juneja, M. and Mishra, D. "Diagnostic Accuracy of International Epidemiology Network (INCLEN) Diagnostic Tool for Autism Spectrum Disorder (INDT-ASD) in Comparison with Diagnostic and Statistical Manual of Mental Disorders-5 (DSM-5)." Indian Pediatrics, 55(6): 482-484. Accessed November 11, 2018. Retrieved from https://www.ncbi.nlm.nih.gov/pubmed/29978815

[10] Thabtah, F. (2018) An accessible and efficient autism screening method for behavioural data and predictive analyses. Health Informatics Journal. 1-17.

[11] Thabtah, F. F. "Autistic Spectrum Disorder Screening Data for Children." 2017. Retrieved from https://archive.ics.uci.edu/ml/machine-learning-databases/00419/

[12] Thabtah, F. F. "Autistic Spectrum Disorder Screening Data for Adolescents." 2017. Retrieved from https://archive.ics.uci.edu/ml/machine-learning-databases/00420/

[13] "AQ-10 (Child Version). National Institute for Health Research. 2012. Retrieved from https://micmrc.org/system/files/webinars/AQ10-Child.pdf

[14] "AQ-10 (Adolescent Version). National Institute for Health Research. 2012. Retrieved from https://micmrc.org/system/files/webinars/AQ10-Adolescent.pdf

[15] Jiawei Han , Micheline Kamber , Jian Pei, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2011

A COMPARISON OF PREDICTION METHODS FOR CUSTOMER CHURN USING SAS ENTERPRISE MINER

Joshua Rhoads and Dr. Jay Annadatha
Department of Computer Information Science, Clarion University of Pennsylvania
J.A.Rhoads@eagle.clarion.edu, jannadatha@clarion.edu

ABSTRACT
Customer Churn is one of the biggest and most critical challenges for any business. By predicting churn, companies can proactively engage and retain their customer base. With machine learning advances, it is possible to train a predictive model using prior churn history to make better churn predictions. This data-driven approach to customer churn applies machine learning to cut down on costs and lost profits due to churning customers.

This paper will evaluate supervised machine learning models to classify customer records based on a binary target variable indicating *customer churn.* Simple classification models will be created to predict the outcome of each record including logistic regression, a neural network, and a decision tree. These models will also be combined into an ensemble and compared to a gradient boosting model. The models will be created and evaluated using SAS Enterprise Miner with comparison by overall accuracy, cumulative percent capture rate, and cumulative lift. Priority for model selection will be given to evaluation based on rank-ordered methods used in business applications due to resource constraints. The evaluation results in a logistic regression model with the best overall accuracy and a neural network with the best cumulative percent capture rate through the second decile.

KEY WORDS
Customer Churn, Data Analysis, Predictive Model, Logistic Regression, Neural Network, Decision Tree, CART, Ensemble, Gradient Boosting, Supervised Model, Classification, Machine Learning, RFM, Clustering, Attrition, SAS, SAS Enterprise Miner

1. Introduction

One of the most important metrics for business performance is the number of customers that stop subscribing or no longer do business with a company, known as churn or attrition [1]. This makes customer churn a critical challenge and affects the business bottom line. A small amount of monthly churn can build up over time and lead to losing half of all customers over a year [2]. The rate of customer churn over time can also be used to measure customer satisfaction and determine how quickly customers are becoming dissatisfied enough to leave.

Analysis of customer churn is useful for contractual industries such as insurance, banking and telecommunications that lose profits when customers close accounts. For most industries, the cost of acquiring customers is five to twenty-five times more than keeping current customers [3]. Continuously finding new customers can increase these costs and reduce profits. For instance, acquiring customers in banking is much more difficult and costlier than keeping current customers [4]. For this reason, predictive models can be developed to assist business decision makers with finding customers that are predicted to leave.

The application of machine learning methods to customer churn can improve the ability for brands and companies to proactively engage customers and reduce churn. With machine learning advances, it is possible to train a predictive model using prior churn history to make better predictions. Those data-driven predictions can be made using supervised learning with labeled customer data collected over time. These predicted customers can also be targeted for engagement and specialized marketing based on predictive models indicating selected customers.

Customer churn can be generally divided into the categories of contractual, non-contractual, voluntary, and involuntary [5]. The following is a summary of each form of customer churn:

- **Contractual:** customers make payments on a schedule as part of a contract, cancellation is recorded.
- **Non-Contractual:** customers can make purchases at any time, cancellation is not recorded.
- **Voluntary:** customers make a voluntary choice to leave
- **Involuntary:** customers are forced to leave due to expiration or out of their control

This analysis will focus on contractual and voluntary churn based on the assumption that the dataset is updated during customer cancellation.

1.1 Customer Churn Analysis – RFM

Traditionally, customer churn has been analyzed based on measures of customer activity that include recency, frequency, and monetary value (RFM) [6]. RFM is based on the assumption that customers that have engaged with the business recently and at high frequency are not likely to churn. This model can work well for non-contractual churn where the assumption is correct, and can be developed into a rule-based model to indicate churning customers.

1.2 Customer Churn Analysis – Machine Learning

Using machine learning methods, predictive models for customer churn can be built. Both supervised and unsupervised techniques are used depending on whether data features are based on RFM or customer demographics and engagement. When RFM features are available for customers, they can be clustered into groups based on variable similarities. These groups can then be given probabilities for churn, and a customer's propensity for churn is determined by which cluster they are added to. This allows for a better understanding of each customer in their relation to each other with large groups of customers to focus attention.

Common machine learning methods for churn prediction use supervised classification to identify customer records [7]. These models are trained using a labeled dataset with a variable in each record indicating if the customer has left (churned) during the data collection period. Supervised classification models such as *logistic regression, decision trees, gradient boosting, and ensembles* can be used and evaluated based on rank-ordered results as well as overall accuracy. Overall accuracy can be used to verify that the models are not overfitting the data while giving priority to rank-ordered evaluation for model selection.

Supervised methods can use transactional data as well as demographics and behavioral data to develop complex rules for predicting customer churn. The most important variables can be identified by the models and used to identify customers. Once the predictive model is trained on labeled data, it can be used to score new customers and identify those likely to churn. The advanced methods used by classification make it more accurate for predicting individual customers than clustering or traditional RFM. Machine learning methods can provide an increase in 10-12% accuracy when compared based on AUC-ROC (Area Under Curve for Receiver Operating Characteristic) and log-loss [7].

1.3 Model Evaluation

Predictive models can be evaluated differently depending on the analysis objectives. These methods are based on classification error (batch), rank order, and profit assessment [8] as listed below.

Batch Evaluation:

- Traditional evaluation methods using results from a confusion matrix.
- Looks at the results of the dataset as a whole.
- Metrics include: overall accuracy, precision, recall, and specificity.

Rank-Ordered Evaluation:

- Records are sorted by propensity (probability of being in the target class)
- Records are evaluated for accumulated correct classification at each percentage depth by propensity.
- Metrics include: lift charts, gains charts, cumulative lift, and percent of captured responses

Profit Assessment Evaluation:

- Provides weights for classifying each record dependent on classification result.
- Some results are more costly or profitable than others
- For instance, false negative may cost more in lost profits than a false positive for customer churn.
- Metrics include: expected profit/loss, total profit/loss

1.4 Rank-Ordered, Lift Based Evaluation

While overall accuracy is a useful measure to evaluate models, in a business application the rank-ordered lift (or rank-matrix) can be more effective. For large datasets used for classification in a business setting, it is often more cost effective to only look at a small percentage of the most likely customers to leave.

- Cumulative lift is based on sorting the records by the probability of the record being in the target class (known as *propensity*) to find the top percentage of records classified. The metric is a ratio between the accumulated records correctly classified within each decile and records found by random selection. This results in a smaller portion of the data being used for targeting after sorting by propensity, and can be less costly than using the entire dataset for evaluation.
- Cumulative percentage captured response is a way to accumulate lift into a percentage of total records predicted by the model (*captured*) in each decile [10]. This can be used to evaluate how much of the churn records are being found by the model up to the decile of choice. The best model can be chosen by using cumulative lift and percentage of captured responses to identify the most likely customers to churn and take action to reduce churn.

The rank-ordered approach to evaluation has the following advantages:

- Provides a small group to focus on with a percentage determined by business leaders.
- The smaller group is easier to engage with when resources are not available for all customers.
- Provides more accuracy than traditional RFM or clustering when customer data is available [7].

This analysis uses ***rank-ordered*** evaluation.

1.5 Analysis Assumptions

The following assumptions are made to simplify the analysis for consistency in results:

- Analysis is focusing on ***contractual*** and ***voluntary*** churn – where data has been labeled, and will not consider customers that have left but have not been labeled.
- The telecom company has ***limited*** **resources** to take action on churning customers.
- Due to limited resources, a ***rank-ordered*** approach will be used to find the top 20% of records most likely to indicate churn (cancel their accounts).
- ***Percent captured response*** through the second decile are used as the primary evaluation value.
- Models should still have relatively high lift and overall accuracy to ensure the models are not overfitting training data.
- ***Test data*** is used for comparing models and avoid overfitting the validation data.

2. Analysis Methodology

Analysis is done using SAS Enterprise Miner which implements the SEMMA data-mining analytical framework. To build the predictive models, data is partitioned into training (40%), validation (30%), and test (30%) datasets. Models are trained by learning from the training data, evaluated and fine-tuned using validation data, and compared for predictive accuracy using test data. Predictive Models developed include logistic regression, decision trees, and neural networks. These models will be combined to create an ensemble, and compared to the single models as well as a gradient boosting model. Performance measures for comparing the models include cumulative lift, cumulative percentage of captured responses, and overall accuracy results from the test data.

2.1 Dataset

The dataset is a CSV file (Telco_Customer_Churn) that contains customer information for a telecommunications company. Data includes information related to accounts, services the customer uses, demographics, and a churn variable to indicate that the customer has left. Data consists of 20 variables and 7043 records. Fig 1 lists the variables in the dataset and Table 1 shows the data dictionary.

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat
21	Churn	Char	3	$3.	$3.
16	Contract	Char	14	$14.	$14.
5	Dependents	Char	3	$3.	$3.
12	DeviceProtection	Char	19	$19.	$19.
9	InternetService	Char	11	$11.	$11.
19	MonthlyCharges	Num	8	BEST12.	BEST32.
8	MultipleLines	Char	16	$16.	$16.
11	OnlineBackup	Char	19	$19.	$19.
10	OnlineSecurity	Char	19	$19.	$19.
17	PaperlessBilling	Char	3	$3.	$3.
4	Partner	Char	3	$3.	$3.
18	PaymentMethod	Char	25	$25.	$25.
7	PhoneService	Char	3	$3.	$3.
3	SeniorCitizen	Num	8	BEST12.	BEST32.
15	StreamingMovies	Char	19	$19.	$19.
14	StreamingTV	Char	19	$19.	$19.
13	TechSupport	Char	19	$19.	$19.
20	TotalCharges	Num	8	BEST12.	BEST32.
1	customerID	Char	10	$10.	$10.
2	gender	Char	6	$6.	$6.
6	tenure	Num	8	BEST12.	BEST32.

Figure 1: Variables in the telecom dataset

Data Dictionary

customerID	Record ID
gender	Male or Female
SeniorCitizen	1 = Yes, 0 = No
Partner	Yes or No, Indicates if the customer has a partner
Dependents	Yes or No, Indicates if the customer has dependents
tenure	Number of months as a customer
PhoneService	Yes or No, Indicates if the customer has phone service
MultipleLines	Yes, No, or No Phone service, Indicates if the customer has multiple lines
InternetService	DSL, Fiber Optic, or No, Indicates if customer has internet service
OnlineSecurity	Yes or No, Indicates if the customer has online security
OnlineBackup	Yes or No, Indicates if the customer has online backup
DeviceProtection	Yes, No, or No Internet Service, Indicates if the customer has device protection
TechSupport	Yes, No, or No Internet Service, Indicates if the customer has tech support
StreamingTV	Yes, No, or No Internet Service, Indicates if the customer has streaming
StreamingMovies	Yes, No, or No Internet Service, Indicates if the customer has streaming movies
Contract	Month-to-Month, One Year, Two Year, Indicates customer contract type
PaperlessBilling	Yes or No, Indicates if the customer has paperless billing
PaymentMethod	Electronic Check, Mailed Check, Bank Transfer, Credit Card, Indicates the customer payment method
MonthlyCharges	Monthly Charge to customer
TotalCharges	Customer's total amount of charges
Churn	Yes or No, Indicates if the customer has left

Table 1: Data Dictionary

2.2. Exploring the Data

Data Exploration is the first analysis step to understand the data and observe the interactions among the variables. By exploring data, analyst can come up with strategies to improve model performances. Fig 2 shows a processing diagram with three nodes: StatExplore, MultiPlot, and Graph Explore.

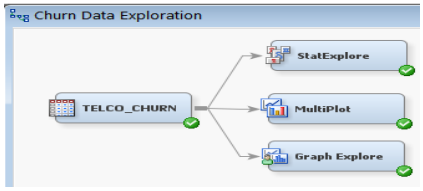


Figure 2: Churn Data Exploration

Both the StatExplore node (performing chi-squared analysis) and multiplot (which plots variable distributions based on the target (churn) variable) show that the variables 'Contract, 'tenure', 'OnlineSecurity', and 'TechSupport' have an important relation to the target (Churn). The graph explore node shows the frequency of Churn as in fig 3. Result: Churn Frequency: 526, Non-Churn Frequency: 2474, Churn Percentage: 17.53%

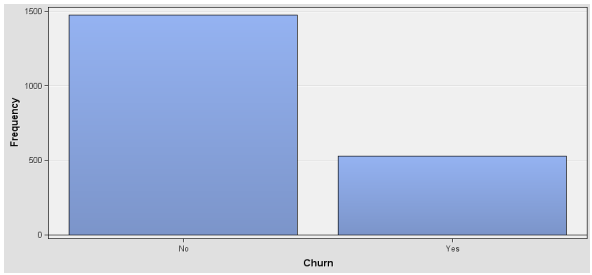


Figure 3: Churn Frequency Chart

2.2.1 Data Preparation – Numeric Variables

Numeric Variable Analysis output is shown in Fig 4.

- MonthlyCharges and TotalCharges show signs of positive skew with TotalCharges showing the most skew in the positive direction.
- Tenure has two peaks at the beginning and end of the distribution, and can be binned for improved model performance.

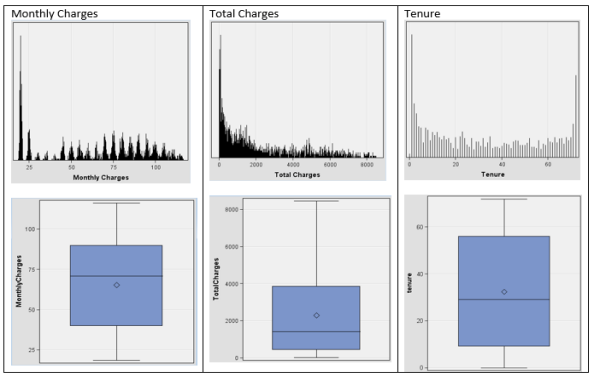


Fig 4: Numeric Variable Distribution

2.2.2 Data Preparation – Categorical Variables

Fig 5 displays a distribution chart for the categorical variables.

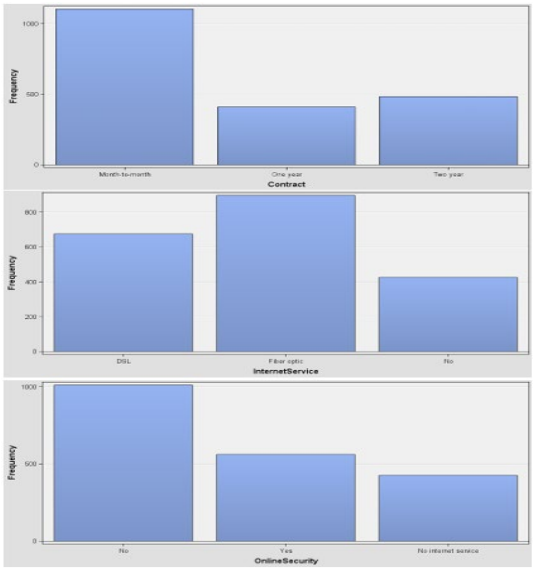


Fig 5: Categorical Variable Distribution

Some categorical variables have categories with same meaning (as shown in Fig 6) such as OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, and StreamingMovies have values for "No" and "No Internet Service".

Fig 6: Online Security Categories with Similar Meaning

Variables showing the biggest differences in proportion are 'PhoneService' (82.4% difference), and 'SeniorCitizen' (68.2% difference).

2.2.3 Data Preparation – Replace/Modify Records

Some of the variables have values with the same meaning as in Fig 6. The record values of "No Internet Service" is ***replaced*** with the value "No" using the 'Replacement' node shown in Fig 7. Similarly, the values of "No phone service" for MultipleLines will be changed to "No".

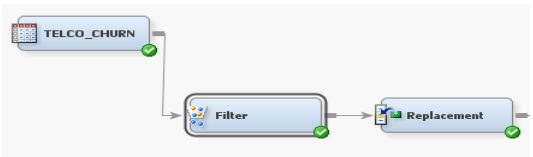


Figure 7: Replacement Node

2.2.4 Data Preparation – Outliers

A boxplot of tenure by churn as in Fig 8 shows possible outliers for values of tenure above 60 months. Analysis can be improved for outlier sensitive models like logistic regression by binning these outlier variables together as shown in Fig 9, with resulting variables shown in Fig 10.

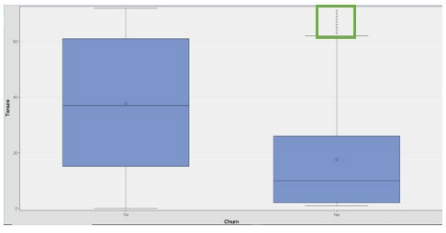


Figure 8: Tenure Boxplot with Outliers

Figure 9: Binning Node to reduce Outliers influence

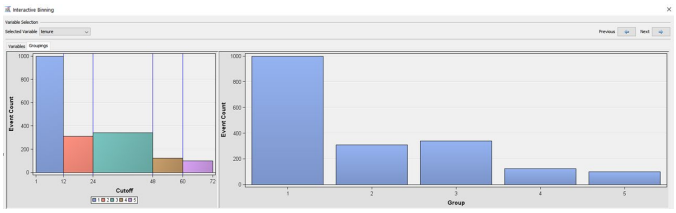


Figure 10: Interactive Binning in SAS Enterprise Miner

2.2.5 Data Preparation – TotalCharges

There are multiple reasons for removing the TotalCharges variable including missing data and multicollinearity with MonthlyCharges.

Analysis of the missing Data shows TotalCharges has missing data. Eleven records are missing out of 7043 records, which is a small portion. Total Charges is an interval variable, and does not have a large influence on churn. Therefore, the missing data can be removed from the dataset without affecting other records.

Including independent variables with a high degree of correlation in the model can cause errors. To avoid multicollinearity, the final variables used in the predictive analysis are selected using a "Variable Selection" node as shown in Fig 11.

Figure 11: Variable Selection Node

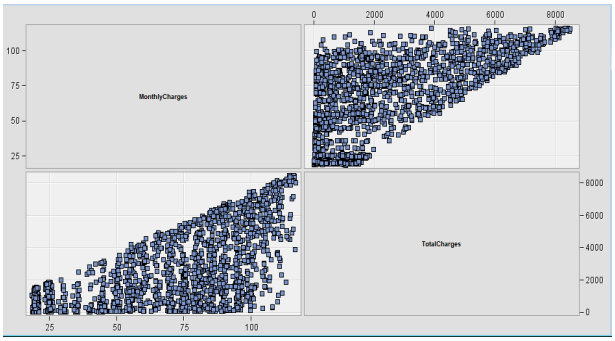


Figure 12: Scatterplot Matrix with Monthly Charges and Total Charges

A scatterplot matrix as in Fig 12 shows a correlation between variables *MonthlyCharges and TotalCharges.* The total customer charges will increase when there are more monthly charges. Accordingly, the TotalCharges variable is ***dropped*** to reduce multicollinearity.

3. Model Development

Churn prediction models are developed using SAS Enterprise Miner by connecting nodes in a processing diagram as shown in Fig 13. Predictive models are implemented using logistic regression, decision trees, neural networks, and ensemble methods as in Fig 13.

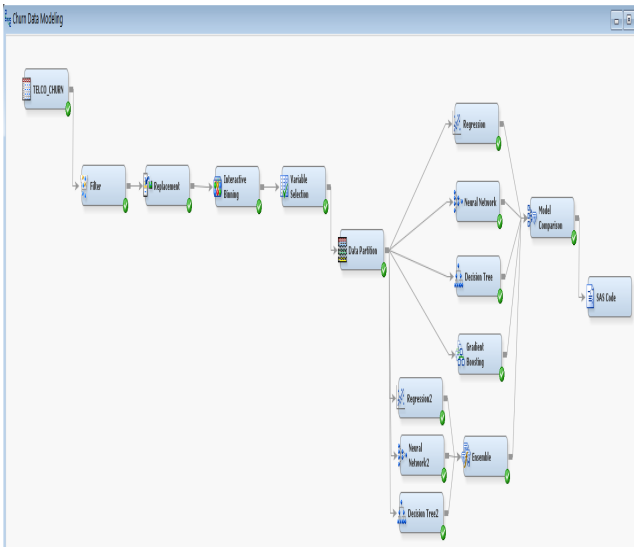


Figure 13: Analytical flow Processing Diagram

3.1 Data Partition

The resulting variables that have been transformed for analysis will be split into training (40%), validation (30%), and test (30%) data as shown in Fig 14.

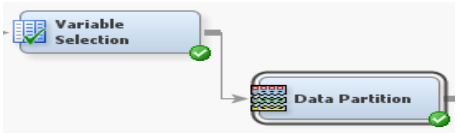


Figure 14: Data Partition Node

The partitioned data is now ready for building the churn models.

Model Results and Evaluation

- Results from each model are assessed for overall accuracy using training and validation data.
- Crossvalidation is used to avoid overfitting the training data.
- Overall accuracy is defined as the ratio of correct classifications to total number of records.
- The rank-ordered evaluation metrics of cumulative lift and percent captured response are evaluated for validation and training data.

3.2 Model 1: Logistic Regression

Logistic regression is used to classify records based on a linear function of the log-odds of a dependent, categorical variable. A selection method is chosen to find coefficients

with the best fit for the independent variables. In this case, churn is the target variable and stepwise selection is used. Table 2 shows the output of the logistic regression analysis.

Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept		1	-2.1906	0.1024	457.98	<.0001
Contract	Month-to-month	1	1.1339	0.1166	94.57	<.0001
Contract	One year	1	-0.0736	0.1238	0.35	0.5520
GRP_tenure	1	1	1.0198	0.0972	110.07	<.0001
GRP_tenure	2	1	0.0243	0.0893	0.07	0.7860
GRP_tenure	3	1	-0.1551	0.0964	2.59	0.1076
InternetService	DSL	1	0.00769	0.0855	0.01	0.9283
InternetService	Fiber optic	1	1.0179	0.0813	156.64	<.0001
StreamingMovies	No	1	-0.2809	0.0573	24.07	<.0001

Table 2: Logistic Regression Model output

Coefficient Analysis:
Pr > ChiSq is <.0001 for Intercept, Contract=Month-to-Month, GRP_tenure = 1, InternetService=Fiber optic, and StreamingMovies=No. These variables are most likely the significant indicators of churn and included in the model.

Accuracy Evaluation of Logistic Regression Model:
Table 3 gives the classification matrix. Overall Accuracy is 79.98% (training) and 81.41% (validation).

	False Negative	True Negative	False Positive	True Positive
Training	349	1850	214	399
Validation	257	1414	135	303

Table 3: Regression Classification Table

Rank-Ordered Accuracy:
Rank-ordered accuracy is evaluated using cumulative lift and cumulative percentage captured response through the second decile as in Table 4.

Bin	Percentile	Cumulative Lift Training	Cumulative Lift Validation	Cumulative % Captured Response Training	Cumulative % Captured Response Validation
1	10	2.79952	2.99858	28.075	30
2	20	2.54274	2.61305	50.909	52.286

Table 4: Regression Lift Table

Results from Lift Table 4 (Logistic Regression):
- Overall accuracy indicates that the model is not overfitting the training data.
- Cumulative lift through the second decile for validation data is 2.61305 with a cumulative % response of 52.286. This indicates that for validation data in the second decile, the model will capture 2.6 times as many customers as no model, and will capture 52.286% of churned customers.

3.3 Model 2: Neural Network

An artificial neural network uses weights between layers of processing nodes for classification or regression. A Neural Network model is created as shown in Fig 15 with a hidden layer having 3 nodes and average error as the selection criterion.

Figure 15: Neural Network Node

Accuracy Evaluation of Neural Network Model:
Table 5 presents the classification matrix for the Neural Network model. Overall Accuracy is 80.37% (training) and 80.75% (validation).

	False Negative	True Negative	False Positive	True Positive
Training	342	1854	210	406
Validation	260	1403	146	300

Table 5: Neural Network Classification Table

Rank-Ordered Accuracy:
Rank-ordered accuracy is evaluated using cumulative lift and cumulative percentage captured response through the second decile as shown in Table 6.

Bin	Percentile	Cumulative Lift Training	Cumulative Lift Validation	Cumulative % Captured Response Training	Cumulative % Captured Response Validation
1	10	2.90617	2.82009	29.144	28.214
2	20	2.52405	2.58806	50.535	51.786

Table 6: Neural Network Lift Table

Results from Lift Table 6 (Neural Network):
- Overall accuracy does not decrease from training to validation indicating the model is not overfitting the training data.
- Cumulative lift through the second decile for validation data is 2.58806 with a cumulative % response of 51.786. Therefore, for validation data in the second decile, the model will capture 2.6 times as many customers as no model, and will capture 51.786% of churned customers.

3.4 Model 3: Decision Tree

Decision tree models apply a selection criteria to split the data into a tree-like structure of smaller groups at each level. The decision tree used in this model uses a chi-squared probability condition, and gives a series of decisions that can be used to classify each record.

Decision Tree Analysis findings:

- The most important variables selected by the decision tree are 'Contract', 'Internet Service', and grouped 'tenure'.
- Records with a contract of one or two years are labeled as Churn=No by the first split.
- Grouped tenure is used twice in the tree split by 'InternetService' under Fiber Optic (or missing) indicating that variable is important.

Accuracy Evaluation of Decision Tree Model:
Overall accuracy is calculated as the ratio of true positives and negatives to the total number of records. As shown in Table 7. Overall Accuracy is 79.8% (training) and 79.75% (validation).

	False Negative	True Negative	False Positive	True Positive
Training	360	1856	208	388
Validation	274	1396	153	286

Table 7: Decision Tree Classification Table

Rank-Ordered Accuracy (Decision Tree):
Table 8 presents the decision tree lift table using which the Rank-ordered accuracy is evaluated.

Bin	Percentile	Cumulative Lift Training	Cumulative Lift Validation	Cumulative % Captured Response Training	Cumulative % Captured Response Validation
1	10	2.67411	2.79431	26.817	27.956
2	20	2.46441	2.47181	49.341	49.46

Table 8: Decision Tree Lift Table

Results of Decision Tree Lift:

- Overall accuracy does not decrease from training to validation indicating that the model is not overfitting the training data.
- Cumulative lift through the second decile for validation data is 2.47181 with a cumulative % response of 49.46. Therefore, for validation data in the second decile, the model will capture 2.6 times as many customers as no model, and will capture 49.46% of churned customers.

3.5 Model 4: Gradient Boosting

A gradient boosting model minimizes the gradient of an error function to sequentially improve the accuracy of classification for multiple generated decision trees. Fig 16 shows the Gradient Boosting node added to the model process flow.

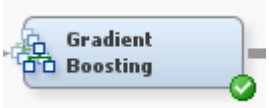


Figure 16: Gradient Boosting Node

Accuracy Evaluation of Gradient Boosting Model:
As shown in Table 9, Overall Accuracy is found to be 80.58% (training) and 81.13% (validation).

	False Negative	True Negative	False Positive	True Positive
Training	384	1866	198	400
Validation	258	1409	140	302

Table 9: Gradient Boosting Classification Table

Rank-Ordered Accuracy (Gradient Boosting):
Rank-ordered accuracy is evaluated using cumulative lift and cumulative percentage captured response through the second decile as shown in Table 10.

Bin	Percentile	Cumulative Lift Training	Cumulative Lift Validation	Cumulative % Captured Response Training	Cumulative % Captured Response Validation
1	10	2.57653	2.83794	25.839	28.393
2	20	2.34376	2.55236	46.925	51.071

Table 10: Gradient Boosting Lift Table

Results of Gradient Boosting Lift:

- Overall accuracy from training to validation indicates that the model is not overfitting the training data.
- Cumulative lift through the second decile for validation data is 2.55236 with a cumulative % response of 51.071. Therefore, this model, for validation data in the second decile, will capture 2.6 times as many customers as no model, and will capture 51.071% of churned customers.

3.6 Model 5: Ensemble

An ensemble uses a collection of models to try to improve accuracy by combining the results. The ensemble is created using the logistic regression, neural network, and decision tree models as in Fig 17. The models are combined by assigning the record to the target class based on averaging the probability of the target variable for each record.

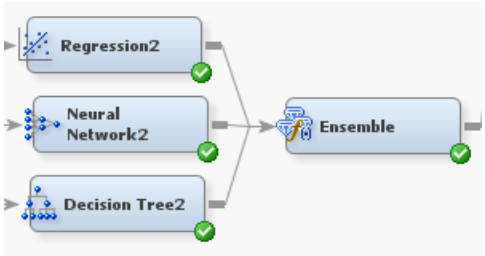


Figure 17: Ensemble Node

Accuracy Evaluation (Ensemble Model):
From the classification table shown in Table 11, Overall Accuracy is 80.33% (training) and 81.18% (validation).

	False Negative	True Negative	False Positive	True Positive
Training	344	1855	209	404
Validation	264	1416	133	296

Table 11: Ensemble Classification Tables

Rank-Ordered Accuracy of Ensemble Model:
Rank-ordered accuracy is evaluated using cumulative lift and cumulative percentage captured response through the second decile as in Table 12.

Bin	Percentile	Cumulative Lift Training	Cumulative Lift Validation	Cumulative % Captured Response Training	Cumulative % Captured Response Validation
1	10	2.86618	2.98073	28.743	29.821
2	20	2.55075	2.60591	51.07	52.143

Table 12: Ensemble Lift Table

Results of Ensemble Lift:

- Overall accuracy from training to validation indicates that the model is not overfitting the training data.
- Cumulative lift through the second decile for validation data is 2.36628 with a cumulative % response of 52.143. Accordingly, for validation data in the second decile, the model will capture 2.4 times as many customers as no model, and will capture 52.143% of churned customers.

3.7 Model Comparison

Model comparison is performed in SAS Enterprise Miner by connecting the output of each model node to the input of a 'Model Comparison' node. A selection criterion is used to decide which of the models is the best by performance. The output from the comparison node is used by a 'SAS Code' node to create a table that includes cumulative lift and percent response rate for training data.

The following options are used during model comparison.

- Ten bins are used to create a lift table by decile.
- ROC index is chosen as the model selection criterion.
- Test dataset will be used for comparison.

Fig 18 shows the comparison of ROC Curves using Training, Validation and Test data.

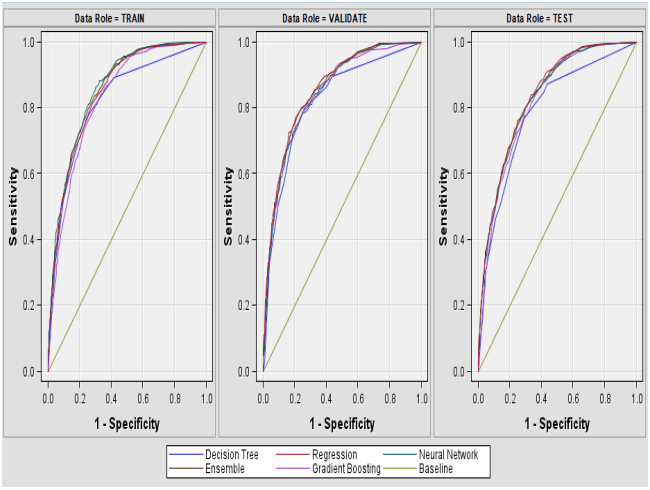


Figure 18: ROC Curves from Model Comparison

ROC Curve Results:

- Most models show relatively good accuracy when used with test data.
- The decision tree model shows a lower accuracy when used with test data and is most likely overfitting the training data

4. Analytical Findings

Five supervised models are compared for predicting the customer churn. This includes an analysis of the variables most likely to indicate attrition, and the predictive model most likely to find *customer churn* sorted by propensity.

4.1 Variables Related to Churn

Based on data exploration, four influencing variables related to churn are identified and included in the models created above. The four variables of interest are:

- **Contract**: this variable gives the type of account the customer has. Customers with Month-to-Month accounts show a higher proportion of churn than other contracts.
- **tenure**: this variable indicates how long the customer has had account (account tenure). Customers with less tenure are more likely to churn.
- **OnlineSecurity** and **TechSupport**: indicate if the customer has online security or tech support. Customers that have online security and tech support are less likely to churn.

4.2 Model Selection

Model selection is based on the cumulative % capture rate, cumulative lift, and overall accuracy.

Accuracy based Model Evaluation:
Table 13 gives the overall accuracy for each model with counts for False Negative (FN), True Negative (TN), True Positive (TP), and False Positive (FP).

	FN	TN	FP	TP	Accuracy
Regression	153	1397	292	269	0.789199
Decision Tree	301	1380	170	260	0.776883
Neural Network	278	1377	173	283	0.786357
Gradient Boosting	299	1395	155	262	0.784936
Ensemble	293	1393	157	268	0.786831

Table 13: Overall Accuracy for Classification Models

Overall accuracy shows that the logistic regression and ensemble methods have the best overall accuracy. All the five models have relatively good accuracy and signs of overfitting is not seen.

Rank-Ordered Accuracy and Comparison:
Table 14 gives the evaluation metrics for each model by cumulative lift and cumulative percent captured response as well as the overall accuracy.

	Cumulative Lift Test - 2nd Decile	Cumulative % Captured Response Test - 2nd Decile	Overall Accuracy Test
Regression	2.39297	47.95	**0.789199**
Decision Tree	2.28306	45.784	0.776883
Neural Network	**2.41966**	**48.485**	0.786357
Gradient Boosting	2.34849	47.059	0.784936
Ensemble	2.36628	47.415	0.786831

Table 14: Cumulative Lift and % Captured Response Comparison

From the results shown in Table 14,
- The logistic regression model has the best overall accuracy for the test data.
- The neural network model has the best cumulative percentage of churn records captured through the second decile.
- The neural network still has relatively good overall accuracy at 78.64%.
- The logistic regression model and neural network are the models that give the best results for the test data set and should be evaluated for improvement and optimization.

From the analytical insights, *Neural Network* or *Logistic Regression* models are *recommended candidates* based on the priority of overall accuracy or rank-ordered lift.

5. Conclusions

5.1 Machine Learning Approach to Customer Churn Analysis

For customer churn prediction, there are advantages to using supervised machine learning models instead of traditional RFM or cluster analysis when customer data is available. Most often, the method chosen depends on the business case as well as available data. When data is available, machine learning methods provide better accuracy in prediction of *customers likely to leave.* This predictive analytics strategy can save money and improve profits by cutting down the cost of new customer acquisition.

5.2 Model Choice for Telecommunications Churn

From this analysis, *neural network* model gives the best cumulative % captured response for the test dataset. This is the model to be chosen for implementation to select the top 20% of customers most likely to close their account. Customer records can be scored and sorted by probability of churn given by the model. Action can taken for the top 20% of customers with the highest propensity for attrition.

The *logistic regression* model also gives a high cumulative % captured response through the second decile. In addition, this model gives the best overall accuracy and is less complex model than the neural network. It is recommended that logistic regression model be used along with a neural network model for performance or combining them into an *ensemble*.

5.3 Recommendations

The following are the recommendations for taking action on customers (predicted) likely to churn:
- Customers with low tenure and a Month-to-Month contract should be given incentives to change to a one-year or two-year contract.
- Customers with low tenure and a longer contract should be given incentives to add online security or tech support.
- Other customers should be given incentives to increase tenure and customer satisfaction.

5.4 Possible Model Improvements and Future Work

The numeric models such as logistic regression and neural network can be improved by transforming numeric variables with skew [9]. The other models can also be optimized and re-evaluated with new data to determine if they have become better at evaluating customers at the second decile. The ensemble and gradient boosting models can also be optimized on new data by varying parameters to improve results. These results can be

compared to the neural network and logistic regression models for better performance.

The optimized neural network and logistic regression models could be used to score new customers and provide groups of the top 20% of customers most likely to churn. This system could use an ETL (Extract, Transform, and Load) step to ingest the customer data into SAS and use the score code from the modes to score the results [10]. This information can be shared with business and marketing leaders for focused engagement with customers.

References

[1] Li, Susan. "Predict Customer Churn – Logistic Regression, Decision Tree And Random Forest". Datascience+, 2019, https://datascienceplus.com/predict-customer-churn-logistic-regression-decision-tree-and-random-forest/.

[2] "How to Leverage AI To Predict (And Prevent) Customer Churn". Towards Data Science, 2019, https://towardsdatascience.com/how-to-leverage-ai-to-predict-and-prevent-customer-churn-f84d653a76fb.

[3] "The Value of Keeping The Right Customers". Harvard Business Review, 2019, https://hbr.org/2014/10/the-value-of-keeping-the-right-customers.

[4] "The 4 D's Of Customer Attrition". The Financial Brand, 2019, https://thefinancialbrand.com/55772/banking-customer-attrition-analysis/.

[5] Dalinina, Ruslana. "Building Customer Churn Models For Business". Datascience.Com, 2019, https://www.datascience.com/blog/what-is-a-churn-analysis-and-why-is-it-valuable-for-business.

[6] "Is RFM Still King? A Data Science Evaluation | Resci". Resci, 2019, https://www.retentionscience.com/blog/rfm-king/.

[7] Abbott, Dean. *Applied Predictive Analytics: Principles and Techniques for The Professional* (John Wiley & Sons, 2014).

[8] Sarma, Kattamuri S. *Predictive Modeling With SAS Enterprise Miner: Practical Solutions For Business Applications, Third Edition* (SAS Institute, 2017).

[9] Shmueli, Galit et al. *Data Mining For Business Analytics. 3rd ed*. (John Wiley & Sons, 2016).

ANALYSIS OF AUTISM SPECTRUM DISORDER DIAGNOSIS IN CHILDREN AND ADULTS

Faizan Syed
Computer Science Department
Slippery Rock University
Slippery Rock, PA, 16057
fxs1006@sru.edu

Hannah Daugherty
Computer Science Department
Slippery Rock University
Slippery Rock, PA, 16057
hld1006@sru.edu

Raed Seetan
Computer Science Department
Slippery Rock University
Slippery Rock, PA, 16057
Raed.seetan@sru.ed

ABSTRACT
Increasing instances of autism necessitate improved methods for screening and diagnosis. The current approach to diagnosis is time consuming and flawed with misdiagnosis being common. Recent research has focused on ways machine learning can be used to shorten and make more precise the diagnostic process for determining autism. This study attempts to determine the algorithm with the greatest precision, sensitivity, and specificity in diagnostic prediction when used on the adult and child autism datasets from the UC Irvine Machine Learning Repository. Using the WEKA tool, we first handled the imbalance between the two sets of data as well as the missing values. Analysis was performed using the Bayes classifier model Naive Bayes and the trees classifying model, J48. The results showed that J48 outperformed Naive Bayes in terms of accuracy, sensitivity, and specificity.

KEY WORDS
Autism Spectrum Disorder; Naive Bayes; J48;

1. Introduction

Autism Spectrum Disorder (ASD) includes several conditions, Autistic disorder, ASD/pervasive developmental disorder (PDD), and Asperger disorder and is characterized by problems with social, emotional and communication skills. It usually results in "significant social, communication and behavioral challenges" and "many people with ASD also have different ways of learning, paying attention, or reacting to things" [1]. Diagnosis of the disorder is difficult with many patients having delayed diagnoses that must be based on behavior and development. While there is no cure for ASD, early intervention can improve a child's development. Treatment can help ASD patients to walk, talk, and socialize with others [1].

The prevalence of autism is currently 1 in 59 children. It "occurs in all racial, ethnic, and socioeconomic groups, but is about 4 times more common among boys than among girls" [1]. People with ASD range from high to low functioning and may require vastly different amounts of care throughout their lives. Unfortunately, many ASD patients experience delays in diagnosis - often due to misdiagnosis with another disorder - and in doing so, are missing out on therapy that can have much more important impact on their lives [3]. Physicians cannot diagnose ASD with a blood test or short office visit. Instead, ASD diagnosis requires a two-step evaluation. First is a developmental screening and then a comprehensive diagnostic evaluation [1].

While research has shown that a diagnosis of ASD is usually reliable by the age of 2, most children are not diagnosed until they have reached 4 years old. Additionally, studies have shown that parents notice the developmental problems associated with ASD by their child's first birthday [1]. Some parents are not educated about the traits they notice in their children and as a result, do not seek necessary diagnostic services [5]. Plus, early misdiagnosis means that autism isn't picked up until the demands of school and social situations increase [3]. The high prevalence of ASD necessitates a faster screening process so children can begin intervention treatment as early as possible.

As noted by other research, "with incidence rates rising, the ability to classify autism effectively and quickly requires careful design of assessment and diagnostic tools [6]. Taking that into account, our study joins others that seek to determine how machine learning algorithms can be used to improve the screening and diagnosis process for autism. Specifically, we will use work with two datasets concerning autism, adult and child, and use them in a comparison. This study will use the WEKA algorithms Naive Bayes and J48 with the goal of finding which can provide the most accurate, sensitive and specific results.

Our paper describes related works that attempted to use machine learning to improve the diagnosis of ASD. Methodology includes the origin of our dataset and the tools used as well as Preprocessing & Analysis Preparation and Analysis Methods. Our results and conclusion follow.

2. Related Work

Due to the high prevalence of autism, there have been numerous studies completed relating to its diagnosis and screening. In their research article, Li et al. describe a potential method of using machine learning to identify adults with autism by measuring motor impairment.

This method would reduce the need to rely on long term observations of the individual. This would be made possible due to "altered motor ability such as poor eye-hand coordination, unstable balance, and unusual gait patterns" observed in autistic patients [4]. If measured accurately, these physical characteristics could give clinicians a reliable tool for diagnosis. Improved autism screening could result from "using non-invasive quantitative measures to objectively quantify differences and classify autism from neurotypical controls" [4].

The study had a small sample size of only sixteen autistic adults and 16 age-sex-IQ matched control volunteers. Researchers collected data on 40 kinematic parameters for comparison as participants were asked to observe and imitate hand movements presented in video clips [4]. Once the results were recorded, machine learning and supervised classification were used and the Naive Bayes classifier determined "which of the 8 different imitation conditions were most suitable for discriminating ASC and neurotypical patients" [4]. Support Vector Machine (SVM) was also tested as a possible classifier using both Linear SVM and Radial Basis Function (RBF) SVM. Linear SVM was found to demonstrate the most sensitivity, specificity, and accuracy compared to other classifiers. Li et al. concluded that "results from this exploratory study may prove valuable in demonstrating the possibility of using quantitative kinematic measure and machine learning for preliminary diagnosis of autism" (2017).
A 2016 study from Bone et al. sought to use machine learning to improve autism screening and diagnosis instruments. Their intention was to "utilize ML to derive autism spectrum disorder (ASD) instrument algorithms in an attempt to improve upon widely used ASD screening and diagnostic tools" [2]. Using data from 1,264 verbal ASD patients and 462 verbal individuals with on ASD developmental or psychiatric disorders, the study created algorithms for classification. This method used support vector machine to target "best-estimate clinical diagnosis of ASD versus non-ASD" [2]. In a promising result, the authors of this study found that their algorithm outperformed others, were tunable, and more efficient. They reached 89.2% sensitivity and 59.0% specificity using only five behavioral codes. The small number of codes needed to achieve a reliable outcome will help improve on the "limitations of current caregiver-report instruments and indicate possible avenues for improving ASD screening and diagnostic tools" [2].

Machine learning was also used in a study aiming to shorten observation-based screening and diagnosis of autism by Wall et al. It is noted that diagnosis of autism via the Autism Diagnostic Observation Schedule-Generic (ADOS) looks to impairment in three developmental areas: language and communication, reciprocal social interactions and restricted or stereotypical interests and activities [6]. Based on this information, researchers "used a series of machine-learning algorithms to study the complete set of scores from Module 1 of the ADOS" [6]. Sixteen classifiers were constructed using Weka. These included ADTree, J48, and Random tree, and others. The algorithms were compared for sensitivity, specificity, and accuracy on the 29 items in the ADOS Module 1. ADTree was found to perform with perfect sensitivity, specificity, and accuracy under these conditions, classifying 612 individuals. Future use of this method could reduce the "overly prohibitive and time consuming" process of ASD diagnosis and screening [6].

3. Methodology

3.1 Dataset

Increase in Autistic Spectrum Disorder cases in Adults and Children demands the development of datasets in behavioral traits. Autism datasets pertaining to clinical and screening are very important to improve efficiency, sensitivity, specificity and predicative accuracy but are very uncommon to find. We preferred to use adult and child autism datasets from the UC Irvine Machine Learning Repository due to its features and characteristics to specifically determine influential autistic traits in adults and children to prove effectiveness in finding ASD cases from controls in behavior science [7].

The data sets studied were extracted from UC Irvine Machine Learning Repository [7]. Adult data set contains 704 instances and Child data set contains 292 patient survey instances. Overall 20 variables were included in the analyses, and those variables were categorized into ten individual and ten behavioral features. Individual attributes contained patient demographic information such as age, gender, ethnicity, jaundice indicator, number of family members with PDD, person taking the test, residence, application if previously used by patient and age group. Remaining 10 unsupervised variables are answers responded for each screening related questionnaire. Predicting Class variable 'Class/ASD' is discrete (nominal), and remaining supporting variables are mixed of discrete, categorical, binary and continuous (numerical).

3.2 Tools

The dataset was analyzed using WEKA (Waikato Environment for Knowledge Analysis) tool, version 3.9.3. Naive Bayes and J48-trees classification models were used to analyze and compare results. Measures of accuracy were used to compare efficiency, sensitivity, specificity and predictive accuracy.

3.3 Preprocessing and Analysis Preparation

Initial step was to handle imbalanced data that existed in child and adult data sets. Data sets contained missing data in age (1% - Child, <1% - Adult), ethnicity (15% - Child, 13% - Adult) and relationship (15% - Child, 13% - Adult) features, compared to adult, child data set contained very less volume of instances and both data sets contained class imbalance (Yes, No).

Missing data was handled using unsupervised attribute filter 'ReplaceMissingValue' in Weka. Addition of sample data to child data set was handled

using Synthetic Minority Oversampling Technique (SMOTE). This supervised instance filter class was processed 3 times on child (866 tuples) and 1 time on adult data set (893 tuples) followed by applying unsupervised instance class Randomize to evenly classify statistics since 10-fold of classifying cross-validation will be used for analyses.

3.4 Analysis Methods and Results

Analysis was performed using two algorithms, Naive Bayes which is a Bayes classifier model and J48 Decision tree classifying model. These two algorithms were applied to classify two data sets using 10 Folds of Cross-validation.

Naïve Bayes algorithm is a commonly-used, machine learning probabilistic classifier to make classifications using Maximum a Posteriori decision rule in a Bayesian setting [11]. Decision Tree algorithm explains the attribute behavior for the given or training instances. Tree classification distributes data into an understandable format [12]. J48 decision tree algorithm accounts for missing values, continuous attribute value ranges and derivation of rules [12].

Results of classification tests are measured by accuracy, sensitivity and specificity.
Accuracy calculates true positive and negative test set tuples, it is calculated using (True Positive + True Negative) / (ALL), Sensitivity measures true positive recognition rate, using (True Positive/ Positive) and Specificity is measures true negative recognition rate using (True Negative/ Negative) [10].

Results obtained from Naive Bayes helps predict probability of successful ASD screening procedures, provides standard optical decision making using 20 variables [8]. J48, a decision tree classifying method uses rule for each path from root to a leaf, each attribute-value pair along a path forms a conjunction and final leaf holds the class prediction [8]. These two methods provide results on accuracy, sensitivity or true positive rate or recognition (positive instances classified), and specificity or true negative rate (negative instances classified) [9]. To perform prediction, Weka is used in gathering information about positive and negative instances including confusion matrix, precision and detailed accuracy for each class. Below Weka filters were used for each schema. Figure 1 shows the results for Nave Bayes and J48 algorithms

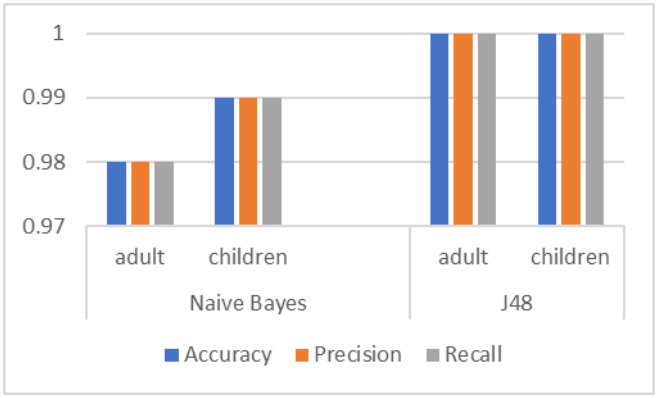


Figure 1: 10-Fold Cross Validation for Nave Bayes and J48 algorithms

4. Conclusion

Two data sets that contained different classifying percentage were analyzed using Naive Bayes and J48 Decision Tree algorithms. After performing analysis, results demonstrate J48 Decision Tree to be better precision (100%) and recall (100%) with fewer or no inaccurate predictions than Naive Bayes.

Overall analysis concludes J48 Decision tree predicted better accuracy and sensitivity and specificity.

References

[1] Autism Spectrum Disorder (ASD). (2018, May 03). Retrieved November 21, 2018, from https://www.cdc.gov/ncbddd/autism/facts.html

[2] Bone, D., Bishop, S. L., Black, M. P., Goodwin, M. S., Lord, C., & Narayanan, S. S. (2016). Use of Machine Learning to improve autism screening and diagnostic instruments: Effectiveness, efficiency, and multi-instrument fusion. Journal of Child Psychology and Psychiatry, 57(8), 927-937. doi:10.1111/jcpp.12559

[3] Ehmke, R. (2018, July 06). Why Autism Diagnoses Are Often Delayed. Retrieved November 21, 2018, from https://childmind.org/article/why-autism-diagnoses-are-often-delayed/

[4] Li, B., Sharma, A., Meng, J., Purushwalkam, S., & Gowen, E. (2017). Applying machine learning to identify autistic adults using imitation: An exploratory study. *Plos One*, 12(8). doi:10.1371/journal.pone.0182652

[5] Thabtah, F., Kamalov, F., & Rajab, K. (2018). A new computational intelligence approach to detect autistic features for autism screening. International Journal of Medical Informatics, 117, 112-124. doi:10.1016/j.ijmedinf.2018.06.009

[6] Wall, D. P., DeLuca, T. F., Harstad, E., & Fusaro, V. A. (2012). Use of machine learning to shorten

observation-based screening and diagnosis of autism. Translational Psychiatry. Retrieved December 3, 2018.

[7] Autistic Spectrum Disorder Screening Data for Adult/Children Data Set (2017). Retrieved from https://archive.ics.uci.edu/ml/datasets/Autistic+Spectrum+Disorder+Screening+Data+for+Children++# https://archive.ics.uci.edu/ml/datasets/Autistic+Spectrum+Disorder+Screening+Data+for+Adolescent+++#

[8] Pei, J., Kamber, M., & Han, J. (2012). Mining: Concepts and techniques. (3rd edition): Morgan Kaufmann

[9] Keyes, M. C., Frank, C. E., Habach, A., & Seetan, R. Artificial Neural Network Predictability: *Patient's Susceptiility to Hospital Acquired Venous Thromboembolism.*

[10] Nikam, S. S, "A Comparative Study of Classification Techniques in Data Mining Algorithms" Oriental Journal of Computer Science & Technology ISSN: 0974-6471, Vol. 8, No. (1): Pgs. 13-19 (2015

[11] Devin S (May 16, 2018), Introduction to Naive Bayes Classification , from https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54

[12] K, Gaganjot., Chhabra, A (2014, July 22). Improved J48 Classification Algorithm for the Prediction of Diabetes: International Journal of Computer Applications (0975 – 8887), Vol. 98, No. (22), from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.678.9273&rep=rep1&type=pdf

REFEREED UNDERGRADUATE PAPERS

PHYSICAL EXERCISE INSTRUCTIONS: UNLOCKING THE KEY TO INJURY-FREE WORKOUTS USING NATURAL LANGUAGE PROCESSING

Matthew Leinhauser | Richard Burns
West Chester University of PA
ML845215@wcupa.edu | rburns@wcupa.edu

ABSTRACT
Instructions should help individuals carry out a task they need help with. In the domain of physical exercise, instructions are very useful in helping an individual execute a biomechanically correct movement and stay injury-free through that movement. If the instructions for that movement are ambiguous and non-descriptive, the risk of suffering injury increases. This research offers a computational approach to improve instructions for bodybuilding exercises that offers correct biomechanical form and minimal chance of injury. We collected "good" and "bad" instructions of thirty different exercises and used natural language processing techniques to analyze the instruction's word count, text complexity, text similarity, lexical diversity, parts of speech, and constituency parse tree. Future research includes analyzing transcripts of physical exercise instruction videos and expanding the corpus used. We also will develop a tool to carry out the best practices for writing instructions in the physical exercise domain.

KEY WORDS
Natural language processing, physical exercise, biomechanics, text annotation

1. Introduction

Physical exercise, specifically bodybuilding, is one of the most common things a person can do to stay healthy. Unfortunately, if not performed biomechanically correct, exercise can also lead to injury. This paper aims to reduce the amount of bodybuilding injuries occurring because of poor execution due to poor text instructions. Using natural language processing (NLP) to model the discourse effectiveness of text in the domain of physical exercise instructions, the safest and most straightforward lexical pattern can be discerned. In this paper, we assess discourse effectiveness by using the Natural Language Toolkit (NLTK) [1], spaCy [2], and textacy [3] to find metrics.

NLTK provides a suite of text processing libraries for classification, tokenization, parsing, tagging, etc. SpaCy is an open-source library for NLP in Python and supports tokenization for over 30 languages. It is designed to help understand large volumes of text and build information extraction and natural language understanding systems. Textacy is a Python library built on the spaCy library and used to perform a variety of NLP tasks such as calculating common text statistics and comparing strings, sets, and documents via similarity metrics.

We chose to use bodybuilding texts versus other fitness texts (cardiovascular exercises, sports plays, etc.) because bodybuilding movements are easily accessible and require minimum equipment. This leads to easy replicability of this research. For example, bodybuilding requires either a gym membership or having access to gym equipment only if the person interested in bodybuilding wants to lift weights. Bodybuilding also includes bodyweight exercises, some of which we used in this research. If we analyzed instruction sets[1] for how to run plays in American football, someone trying to replicate this research would need a football helmet, various pads to protect the body, such as shoulder pads, cleats, a football, and an open field.

2. Exercise Texts

To carry out this study, thirty random exercises, each no more than 185 words, were curated from [4 – 10]. The instruction sets for these exercises were then classified as either good or bad based by the authors expertise in each exercise. A good physical exercise instruction set is one that depicts a clear mental image of how to perform the exercise described while also factoring in safety and the reader's experience level with the exercise. A bad physical exercise instruction set does not take into account experience level or safety and does not paint a mental image because the wording does not flow.

All thirty good physical exercise instruction sets came from one book and 27 of 30 bad physical exercise instruction sets came from three books who share the same author. The remaining three bad instruction sets came from three different books with three different authors. The reason for this setup is because the thirty exercises were determined before determining which books to use. The book used to measure the good exercise instruction sets contained all thirty exercises. The bad instruction sets came from multiple books because all thirty exercises could not be

[1] Instruction set refers to the whole instruction text.

found in one book. A table[2] containing each exercise used and the source of the good and bad exercise instruction sets[3].

Some sample texts are:

1. You hang on a chin-up bar and while trying to stay perfectly vertical raise your legs up until they are 90 degrees.

2. Take hold of a barbell with an underhand grip, hands close together. Straddle a bench with your forearms resting on the bench but with your wrists and hands hanging over the end, elbows and wrists the same distance apart. Lock your knees in against your elbows to stabilize them. Bend your wrists and lower the weight toward the floor. When you can't lower the bar any farther, carefully open your fingers a little bit and let the weight roll down out of the palms of your hands. Roll the weight back up into your hands, contract the forearms, and lift the weight as high as you can without letting your forearms come up off the bench. Forearms, like calves, need a lot of stimulation to grow, so don't be afraid to make them really burn.

3. Stand with your shoulders under the supports of a standing calf raise machine and your toes on the elevated stand provided so that your heels are hanging off the end. Keep your back straight and hands gripping the handlebars. With everything remaining still, stand up on your toes as high as you can. Hold the position for a brief moment to maximize the work on your calves. Slowly lower your heels back down towards the starting position the same way you brought them up.

In the case of Example 1, there is very little text to describe the movement. However, the text may be better described with less words. The phrase, "while trying to stay", could be replaced with "staying". Instead of suggesting the reader stay "perfectly vertical", the instruction now tells the reader to stay perfectly vertical. If the author was not implying the reader should stay perfectly vertical through the entire movement, more words are needed to make the author's point clear. The author cannot be too wordy in writing the instruction or the reader will skim the content. Similarly, too little words may not leave the reader with enough information. For these reasons, word count was measured to try and find the optimal instruction length before the reader becomes disinterested and/or confused.

[2] Table found at http://mattleinhauser.com/chart.html

[3] The instruction sets were categorized by the author's experience with bodybuilding.

[4] Instruction point refers to an individual sentence within an instruction.

Looking at Examples 2 and 3, the instruction points[4] seem difficult to understand. In the case of Example 2, right away the author wants the reader to hold a barbell a certain way in the first instruction point and then also position themselves a certain way in the second point. Based on the verbiage, these instructions seem hard to follow if the reader has no experience with the exercises. This could lead to confusion. Depending on the instructions' complexity, the reader could execute the movement exactly how the author wants or completely miss the mark. However, if the instructions are too simple, they might not be needed at all. By measuring the complexity of the text, we can find how simple or difficult the instruction set is to understand and in turn how well the reader will perform the exercise.

The above examples do not appear to be extremely similar to each other, but other examples are. For instance, the phrase, "with everything remaining still", appears multiple times in the bad physical exercise instruction sets. Having similar instruction points in multiple exercises' instruction sets could lead the reader to think about another exercise when performing a specific exercise. This could lead to injury. On the other hand, similarity helps with simplicity. If the author is trying to convey a point, repetition might help. By measuring text similarity, we will find out whether a higher degree of similarity is used with the good physical exercise instruction sets or the bad ones.

With words such as "straddle", "stabilize", "elevated", and "maximize", examples 2 and 3 seem more lexically diverse than example 1. While the word variation is greater, example 1 is definitely simpler to read than the latter examples. The variety of words an instruction set contains may make the instructions more complex than they actually are. However, example 1 may not contain enough of a variety of words for a reader to accurately understand how to perform the movement. Measuring Lexical diversity (LD), or the distinct number of words in a body of text, takes this into account and determines the variety of text needed for a reader to understand the task at hand. For more information regarding LD, see Section 3.4.

Briefly looking at the three examples, it is easy to tell the parts of speech (POS) of each instruction point are different. The first word in example 1 is a pronoun while the first word in the second and third examples is a verb. The second word in example 1 is a verb, a noun in example 2, and a preposition in example 3. Which exercise is more understandable to the reader could depend on the order of the parts of speech of each instruction point and/or each instruction set as a whole. If that is the case, by looking at the parts of speech involved with each text, we can find the best pattern to convey a clear message to the reader. How the POS for each instruction set were found can be found in Section 3.5.

All three examples are all made up of different phrases and clauses too. Along the same lines as looking at the text's

POS, looking at the phrases and clauses of each instruction point of each good and bad instruction set can determine what sentence structure is better linguistic constituent perspective. For more information on the process of parsing, see Section 3.6.

3. Computing Feature Measures

This section contains information regarding how we believe quantitative metrics can be used to analyze the good and bad physical exercise instructions. Five measures (word count, text complexity, text similarity, lexical diversity, parts of speech, and constituency parse tree) were considered. The reasoning behind the consideration is explained in the below sub-sections.

To process these metrics in python, all instructions were first formatted into a text file (.txt), then a separate text file was created with the names of each of exercise. The exercises were then opened and read by being processed with each exercise name linking to a directory[5] where the physical exercise instruction sets existed.

3.1 Measuring Word Count

Word count is defined as the length of an instruction set from start to finish in terms of the words and punctuation symbols which appear. Word count was found by using the split[6] method in Python, which returned a set of all the words in the instruction set. By using the split method, any punctuation found in an instruction point is appended on to the word before it. For example, applying the split method to the sentence,

I see my dog.

would give the following tokens:

"I", "see", "my", "dog."

Stop words were also included because word count focuses on solely counting the number of words involved in an instruction set and not on a deeper understanding of the instruction set. Stop words are defined as common words – like "a", "the", and "it" – which provide little value to understanding the text.

To find the total number of words in each instruction set, the len() method was applied to the set of words. Putting the word count in an array and dividing by the total number of exercises (30), the average word count for a good physical exercise instruction set and a bad physical exercise instruction set was found.

[5] Corpus found at http://mattleinhauser.com/corpus.html

[6] The split method is defined as returning a list of all the words in a string separated by whitespace if with no parameters.

Three exercises where the bad physical exercise instruction sets were longer than the good physical exercise instruction sets were Dumbbell Shrugs, Lat Machine Pulldowns, and Lunges. Seeing as these three movements are not extremely complex, this was expected.

Shrugs involve hiking the shoulders upward; Lat Machine Pulldowns involve grabbing a bar at a lat machine and pulling it down; and lunges involve stepping into a forward kneel. Due to the simplicity of the exercises, the amount of words used to describe them is not as significant as the number of words used to execute a front squat or a bench press.

Text Category	Average Words per Instruction	Standard Deviation
Good	113	35.042
Bad	75	21.522
Total	94	34.701

Table 1: Average Word Count and Standard Deviation

3.2 Measuring the Text Complexity of each Instruction Set

To find the text complexity of each good and bad physical exercise instruction set, the textacy package was used. This package comes with functions built in for measuring key text complexity metrics such as: Flesch-Kincaid Reading Ease (FKRE) [11], Flesch-Kincaid Grade Level (FKGL) [12], Gunning-Fog Score (GFS) [13], and Automated Readability Index (ARI) [14]. In these equations, ASL is average sentence length, ASW is average number of syllables per word, and PHW is the percentage of hard words.

$$FKRE = 206.385 - (1.015 \times ASL) - (84.6 \times ASW)$$

Equation 1: Flesch-Kincaid Reading Ease

$$FKGL = (0.39 \times ASL) + (11.8 \times ASW) - 15.59$$

Equation 2: Flesch-Kincaid Grade Level

$$GFS = 0.4(ASL + PHW)$$

Equation 3: Gunning-Fog Score

$$ARI = 4.71\left(\frac{characters}{words}\right) + 0.5\left(\frac{words}{sentences}\right) - 21.43$$

Equation 4: Automated Readability Index

All four measurements were used to evaluate the text complexity of each good and bad physical exercise instruction set. These four measurements were chosen over other readability measurements based on how applicable they were to this study as defined by Conzett [15]. FKRE

is commonly used in academics and looks at the average number of syllables per word and the average number of words per sentence. FKGL measures the same variables as the FKRE score, but makes results easier to understand for a broader audience. GFS looks at "complex" words (three or more syllables) and does not take into consideration proper nouns, jargon, and compound words. There is no upper bound, but the score is designed to represent a U.S. grade level. In this research, the GFS has an upper bound of 12, representing 12th grade as the highest level of education. The ARI takes into consideration characters per word, rather than syllables per word, and words per sentence. Due to the uniqueness of each metric, these four metrics were used instead of just one.

Text Category	Avg. FKRE	Avg. FKGL	Avg. GFS	Avg. ARI
Good	77.933	7	10	9
Bad	78.600	6	9	8

Table 2: Average Readability Metrics

Looking at the average FKRE scores, the good physical exercise instruction sets proved to be slightly more difficult to read than the bad physical exercise instruction sets. This result was unexpected as the good instruction sets should be easier to understand than the bad instruction sets. Similarly, this was the case for the FKGL, GFS, and ARI scores.

With consistency across all four readability measures, a certain level of complexity might be needed in order to create a descriptive image of how a bodybuilding movement should be performed. The American College of Sports Medicine recommends a child to begin strength training once they have the emotional maturity to accept and follow directions [16]. The readability measures might be an indicator of when a child/adolescent is emotionally mature enough to perform the movements in our corpus.

3.3 Measuring Text Similarity

To find the text similarity of each good and bad physical exercise instruction set, the Jaccard Similarity equation [17] was used:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A|+|B|-|A \cap B|}$$

Equation 5: Jaccard Similarity Coefficient

In the case of this research, A represents the physical exercise instruction set being compared and B represents a different physical exercise instruction set.

The average similarity for the good physical instruction sets and bad physical exercise instruction sets were found by comparing each exercise instruction set to the 29 other exercises' instruction sets in their respective domain (good and bad). When an instruction set was compared to itself, its result was excluded. A total of 870 comparisons were made for each domain.

Text Category	Avg. Similarity Percentage
Good	15.426%
Bad	17.716%

Table 3: Average Text Similarity

This result confirmed our hypothesis that bad physical exercise instruction sets would be more similar to each other. We hypothesized this because bad instructions do not fully grasp the context of a specific exercise and instead generalize all movements.

3.4 Measuring Lexical Diversity

Lexical diversity (LD), or finding the amount of distinct words, is found by looking at the length of the set of tokens in a physical exercise instruction set and dividing that number by the total length of the instruction set. The equation used was:

$$LD = \frac{\#\ of\ distinct\ words}{total\ \#\ of\ words} \times 100\%$$

Equation 6: Lexical Diversity

This result confirmed our hypothesis that bad physical exercise instruction sets are more lexically diverse than good physical exercise instruction sets. We hypothesized this because a bad physical exercise instruction set will either overcomplicate a particular movement or synthesize all movements into a generalized action.

Text Category	Avg. Lexical Diversity Percentage
Good	5.876%
Bad	8.028%

Table 4: Average Lexical Diversity

3.5 Finding Parts of Speech Patterns

The parts of speech for each instruction set were found using Stanford CoreNLP [18] and the Python wrapper, stanfordcorenlp [19]. The POS tags are from the Penn Treebank Project [20]. The Penn Treebank Project annotates naturally-occurring text for linguistic structure by producing parse trees and annotating text with POS tags. Using pre-defined methods in the wrapper class, the POS for each instruction set were easily found. An example of a tagged sentence is:

[7] The List of changed POS tags list can be found at http://mattleinhauser.com/wrongtags

Lie/NN on/IN a/DT flat/JJ bench/NN with/IN your/PRP$ feet/NNS flat/JJ on/IN the/DT floor/NN ./.

Some POS were incorrectly labeled and were changed using the author's judgement. An example of this is the word "lie". When used in the context of beginning a sentence with, "Lie down", the word "lie" is used exclusively as a verb, however the POS Tagger consistently processed it as a noun. "Lie" was not the only word which was consistently tagged wrong[7].

As of the time of writing, we are still identifying tag patterns between the good and bad exercises. Thus far, these basic patterns were identified:

- 27 of 30 (90.00%) good physical exercise instruction sets start with a verb. 20 of 27 are base form verbs (VB) and the other 7 are gerund/present participle verbs (VBG).
- 26 of 30 bad physical exercise instruction sets start with a verb (VB).
- 22 of 30 (73.33%) good physical exercise instruction sets end with a noun. 17 are singular nouns (NN), 4 are plural nouns (NNS) and one is a proper plural nouns (NNPS).

At the first glance, looking at almost every good and bad physical exercise instruction set shows they look very similar as most start with a verb and end in a noun.

To better understand the makeup of each instruction set we also found the minimum, maximum, and average number of base form verbs (VB), singular nouns (NN), prepositions/subordinating conjuctions (IN), determiners (DT), adjectives (JJ), and adverbs (ADV).

Text Category	Good	Bad
Min. # VB	3	2
Max. # VB	16	10
Avg. # VB	9	5
Min. # NN	7	2
Max. # NN	35	26
Avg. # NN	19	12
Min. # IN	5	2
Max. # IN	29	13
Avg. # IN	14	10
Min. # DT	5	1
Max. # DT	26	16
Avg. # DT	14	7
Min. # JJ	1	0
Max. # JJ	14	11
Avg. # JJ	6	4
Min. # RB	4	1
Max. # RB	16	13
Avg. # RB	10	7

Table 5: Minimum, Maximum, and Average Number of Parts of Speech

The table shows that the minimum, maximum, and average numbers for base form verbs, singular nouns, prepositions/subordinating conjunctions, determiners, adjectives, and adverbs are all higher for the good physical exercise instruction sets versus the bad sets. Since the word count is higher for 90% of good instruction sets against bad instruction sets, this result was expected.

3.6 Finding Patterns in the Constituency Parse Tree

There are two types of parsing: constituency and dependency. A constituency parse tree identifies phrases found in each sentence of a text whereas a dependency parse tree identifies relations between words in each sentence of a text. With instructions, it is much more beneficial to look at how each instruction point is made up phrase-wise rather than trying to identify how the individual words in each instruction point are related. By identifying the phrase makeup of an instruction point, patterns emerge for the instruction set.

Using the Stanford CoreNLP Parser and the stanfordcorenlp wrapper class, the following patterns emerged from looking at the constituency parse trees for each physical exercise instruction set in the good and bad domain:

- 24 of 30 (80.00%) good physical exercise instruction sets start with a simple declarative clause (S). 5 of the remaining 6 have no clause and the one remaining has an inverted declarative sentence (SINV).
- 27 of 30 (90%) bad physical exercise instructions start with a simple declarative clause (S). The remaining three have no clause.

The similar clause structure of both exercise instruction sets confirms why the readability metrics and POS makeup are very similar.

We also found the maximum, minimum, and average number of simple declarative clauses (S), clauses introduced by a subordinating conjunction (SBAR), verb phrases (VP), noun phrases (NP), prepositional phrases (PP), and, adverb phrases (ADVP).

Text Category	Good	Bad
Min. # S	6	4
Max. # S	24	17
Avg. # S	14	9
Min. # SBAR	0	0
Max. # SBAR	9	4
Avg. # SBAR	3	2
Min. # VP	10	6
Max. # VP	35	22
Avg. # VP	24	13
Min. # NP	16	6
Max. # NP	59	41
Avg. # NP	36	25
Min. # PP	5	1
Max. # PP	24	17
Avg. # PP	14	10
Min. # ADVP	1	0
Max. # ADVP	17	9
Avg. # ADVP	6	5

Table 6: Minimum, Maximum, and Average Number of Clauses and Phrases

The table shows the maximum and average numbers for simple declarative clauses, clauses introduced by a subordinating conjunction, verb phrases, noun phrases, prepositional phrases, and adverb phrases are all higher for good physical exercise instruction sets when compared to the bad sets. The minimum number is higher in every case for the good sets except for the minimum number of clauses introduced by a subordinating conjunction. These results were expected because the word count for the good physical exercise instruction sets was higher than the bad sets. A surprising finding is how much higher the average number of verb phrases and noun phrases the good sets contain. This could be the key difference in understanding what makes an instruction set good versus bad in the realm of physical exercise.

4. Conclusion

The good physical exercise instruction sets had a higher word count, a higher average of various text complexity metrics, a lower average similarity, and lower average lexical diversity than the bad physical exercise instruction sets. From this, one can infer that a good physical exercise instruction set uses more complex language to create a mental image of the movement, more words to describe how to execute the exercise safely and efficiently, is not very repetitive and does not use a copious amount of distinct words. The lower average lexical diversity and higher text complexity could be because the good physical exercise instruction sets came from one book, whereas the bad physical exercise instruction sets came from six. This also could have affected all the metrics looked at, but we believe this to be unlikely. We believe all metrics were not affected by coming from one author because three of the six books used for the bad instruction sets have the same author. This author's writing was used for 27 of the 30 bad instruction sets. The patterns created from the POS tagger and constituency parse trees show the best phrasing and lexical composition of how to best communicate a written physical exercise instruction set to an individual.

The accuracy of these results can be improved by expanding the corpus used and involving biomechanics experts, such as physical therapists and athletic trainers, and experienced and novice bodybuilders, in a study asking them to grade the physical exercise instruction sets on how effective they are for creating a safe and easy-to-perform movement.

As mentioned earlier, future work includes identifying more tag patterns and parse patterns between the good and bad exercises. Additionally, we would like to take the transcripts from popular YouTube videos describing bodybuilding exercises and compare the differences in their lexicon versus the lexicon bodybuilding books use. Once we have identified consistent patterns across platforms, we will build a tool designed to help bodybuilding authors and videographers convey safe and straightforward physical exercise instructions to their respective audiences.

5. Acknowledgements

The author would like to acknowledge Demetrius Woodard for helping to choose which physical exercise instruction sets were good and which were bad. While we did not always agree, we appreciate his thoughtfulness and rationale behind his choices.

References:

[1] S. Bird, E. Klein, & E. Loper, *Natural language processing with python* (Beijing: OReilly Media, 2009).

[2] Explosion AI. (2016). Architecture · spaCy API Documentation. Retrieved from https://spacy.io/api/

[3] Chartbeat, Inc. (2016). API Reference. Retrieved from https://chartbeat-labs.github.io/textacy/api_reference.html

[4] A. Schwarzenegger, & B. Dobbins, *The new encyclopedia of modern bodybuilding* (Simon & Schuster USA, 2012).

[5] R. G. Price, *Ultimate guide to weight training for running* (Chicago, IL: Price World Publishing, 2014).

[6] R. G. Price, *Ultimate guide to weight training for baseball and softball* (Cleveland, OH: Price World Publishing, 2004).

[7] R. G. Price, *Ultimate guide to weight training for golf* (Cleveland, OH: Price World Publishing, 2006).

[8] J. R. Little, *Beginning bodybuilding: real muscle/real fast* (New York, NY: McGraw-Hill, 2008).

[9] *Zyzz's bodybuilding bible* (2011).

[10] K. Brungardt, *The complete book of abs* (New York, NY: Random House, 1999).

[11] Rudolf Flesch, *How to Write Plain English* (University of Canterbury, 1981).

[12] Kincaid JP, RP Fishburne Jr, RL Rogers, BS Chissom (February 1975). "Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy enlisted personnel" *(PDF)*. Research Branch Report 8-75, Millington, TN: Naval Technical Training, U. S. Naval Air Station, Memphis, TN.

[13] Gunning, Robert, *The Technique of Clear Writing* (McGraw-Hill, 1952).

[14] Senter, R.J., E.A. Smith. (November 1967). "Automated Readability Index". Wright-Patterson Air Force Base: iii. AMRL-TR-6620.

[15] Conzett, L. (2017, October 10). What are Readability Metrics? Retrieved from https://raven.zendesk.com/hc/en-us/articles/202308564-What-are-Readability-Metrics-

[16] Faigenbaum, A. and Micheli, L. (2017). "Youth Strength Training". Indianapolis, IN: American College of Sports Medicine.

[17] Gupta, S. (2018, May 15). Overview of Text Similarity Metrics in Python – Towards Data Science. Retrieved from https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50

[18] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. (2014). The Stanford CoreNLP Natural Language Processing Toolkit In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.

[19] Lynten. (2018, August 01). Lynten/stanford-corenlp. Retrieved from https://github.com/Lynten/stanford-corenlp

[20] Beatrice Santorini, "Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision)", . July 1990.

A COMPARISON OF AUTOMATIC EXTRACTIVE TEXT SUMMARIZATION TECHNIQUES

Alex Day[1] , Soo Kim[1]
[1]Clarion University of Pennsylvania, Computer Information Science Department
A.D.Day@eagle.clarion.edu, skim@clarion.edu

ABSTRACT

Text summarization is a method of shortening a document by producing a set of representative information. It can be generally categorized into extractive and abstractive summarization. Extractive summarization - the focus of this paper - is extracting sentences and phrases that are already present within the document in order to produce an outline. Term Frequency - Inverse Document Frequency is a score-based summarization technique that assigns a numerical value to each term in the document and then scores the relative importance of each sentence. TextRank is a graph-based summarization technique which uses a graph data structure to rank the document into the relevant sentences. This paper describes those two extractive text summarization methods and presents the results of running the algorithms on three different corpora: *Moby-Dick* by Herman Melville, a selection of Reuters news articles, and a selection of posts on Reddit. The algorithms rank the sentences by relevance to the document, take the top five sentences, and return them as a summary.

KEY WORDS

Natural Language Processing, Extractive Summarization, TF-IDF, TextRank, Graph Summarization, Ranked Summarization

1 Introduction

The goal of automatic summarization is finding a small amount of information that is most representative of a larger set of information. This can be done with images, video, or text. The need for efficient and accurate summarization algorithms has only risen with the onset of the Internet. According to the study done by the Media Insight Project [1], about 6 in 10 Americans only read the headline of a news article and delve no deeper than that. If people are willing to read one sentence headline, they may also be willing to read a 5-sentence summary. This summary would result in the people getting more information about the story, letting them be better informed.

Automatic summarization is categorized broadly into two techniques: extractive and abstractive. Extractive summarization takes words and sentences already within the document to form a summary, whereas abstractive summarization tries to understand the information within the document and summarize it by creating new sentences.

Arguably, one of the most cited and earliest works in automatic summarization is that of Luhn [2], which provides an overview of work by IBM in automatically generating technical paper abstracts using an early TF-IDF-esq algorithm. The original idea for this paper was inspired by Samir Bajaj [3] and his paper entitled *Shakespeare in One Hundred Words*. Samir investigated four abstractive summarization methods, such as TF-IDF–based relevance ranking, K-means clustering algorithm, singular value decomposition, and PageRank. In addition, Dipanjan Das, et. al [4] surveyed different summarization methods: naive-Bayes methods, rich features and decision trees, hidden Markov models, log-linear models, neural networks and third party features, deep natural language analysis methods, abstraction and information fusion, topic-driven summarization and maximal marginal relevance, graph spreading activation, centroid-based summarization, and so on. Dipanjan Das, et. al discussed some unconventional approaches that can be useful in the future of summarization research and elaborated some evaluation techniques as well as the standards for evaluating summaries automatically.

This paper focuses on extractive summarization methods: Term Frequency - Inverse Document Frequency (TF-IDF) and TextRank. These algorithms summarize a document in completely different ways. TF-IDF scores each sentence on relevance to the document as a whole, while TextRank is a graph-based sorting algorithm that separates each sentence into a node and links it to other sentences based on similarity. The methods TF-IDF and TextRank are explained in Section 2 and Section 3, respectively. The implementation details and the results of running these algorithms on three different corpora are presented in Section 4 and Section 5. Finally, the conclusion is given in Section 6.

2 Term Frequency - Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) [5] is one of the most used weighting schemes for term importance. TF-IDF assigns a value to each word in a document

and this assigned score indicates the importance of the word to the document. It is proportional to the frequency of the term in the document and offset by the term's use throughout the whole corpus. This offset ensures that common words such as "the", "and", "or", etc., do not skew the score since they have such low inverse document frequencies. The calculations used are shown in Equation 1. It should be noted that Equation 1 is one of the ways to calculate a TF-IDF score [3] and other non-logarithmic scales can be used, depending on the desired outcome.

$$\begin{aligned} tf(t,d) &= f_{t,d} \\ idf(t,D) &= log\frac{|D|}{|\{d \in D : t \in d\}|} \\ tfidf(t,d,D) &= tf(t,d) \times idf(t,D) \end{aligned} \tag{1}$$

where t = Term to calculate the score of
d = Document to calculate the score relative of
D = Corpus of documents such that $d \in D$

The *tfidf* is a score that is assigned only to a specific word within a document, so it cannot be used directly to assign importance-values to a whole sentence. In order to get an importance-value of a whole sentence, the average of each *tfidf* score is taken for every word within that sentence as shown in Equation 2.

$$\frac{1}{|S|} \times \sum_{w \in S} tfidf(w,d,D) \tag{2}$$

where w = Word present in d to calculate the score of
S = Set of sentences in d
d = Document to calculate the scores relative of
D = Corpus of documents such that d ∈ D

Algorithm 1 uses both of these equations. It calculates the *tfidf* score for each sentence in a document and then returns a list of the sentences within the document sorted by the sentence scoring. The term frequency score for each term is individual for the document that the term is in. The inverse document frequency of a term will only have to be recalculated each time when a document is added to the corpus. The order of the sentences as they appear in the original document is also important to the summary. In order to represent this fact, the sentences in the summary are sorted by the original location of the sentences within the document.

3 TextRank

PageRank is a web page ranking algorithm developed for use with the Google search engine [6]. PageRank determines the importance of each web page returned in the search results through the number of web pages that link back to it. For example, a page such as an article on Wikipedia is important because many pages are linked to it. TextRank [7] takes

Algorithm 1: Return the ranked sentences for a document using TF-IDF

Input: A set of documents, C
A document to summarize, D, such that $D \in C$
Output: A list of sentences sorted based on relevance
tfidf_scores[] ← 0;
for *sentence* ∈ D **do**
  tfidf_sum ← 0;
  for *word* ∈ *sentence* **do**
    tfidf_sum += $tfidf$(word, document, corpus);
  end
  tfidf_scores[word] ← $\frac{tfidf\_sum}{|D|}$;
end
return **sorted**(tfidf_scores)

the same graph-oriented approach as PageRank; however it deems a node in the graph important if that node is similar to many other nodes.

TextRank begins by separating each of the sentences in documents into nodes in a graph. Each node is linked to every other node with the weight of that edge being the similarity score. The most relevant sentences are represented by the nodes with the highest cumulative total of all edges that are connected to the node. The algorithm used in the code for this paper is shown in Algorithm 2 and an example of a document similarity graph is shown in Figure 1. In this example, the document consists of three sentences S_1, S_2, and S_3. S_1 has the similarity score 0.3 with S_2 and the similarity score 0.9 with S_3, so the sum of the similarity scores for S_1 is 1.2. The three sentences would be ranked in $[S_1, S_3, S_2]$ with their sums of similarity scores of all connected edges being $[1.2, 1.0, 0.4]$, respectively. As the result, S_1 would be ranked as the most relevant sentence and selected for the summary.

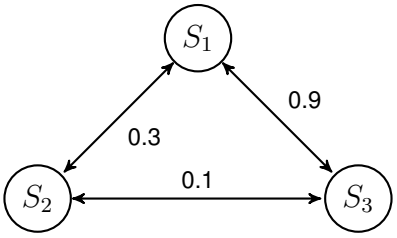


Figure 1: A sentence similarity graph for a document containing the sentences S_1, S_2, and S_3 represented as nodes on the graph where the edges represent the sentence similarities

TextRank relies on a concept called sentence embeddings to calculate the sentence similarity. A sentence embedding is a vector representation of that sentence. This vector representation is usually produced by a neural network that has been trained on a corpus. The neural network maps the sentence to a vector representation. It can glean this information from position in the document or surrounding sentences.

Cosine similarity is used to calculate the similarity of two

Algorithm 2: Return the ranked sentences for a document using TextRank

Input: A set of documents, C
A document to summarize, D, such that $D \in C$
Output: A list of sentences sorted based on relevance

```
G ← empty graph ;
for sentence ∈ D do
    G.add_node(sentence);
end
for outer node ∈ G do
    for inner node ∈ G : inner node ! = outer node do
        G.add_vertex(outer_node, inner_node,
          similarity(inner_node, outer_node));
    end
end
sentence_scores[] ← 0;
for node ∈ G do
    sentence_scores[node] = sum(G.get_edges(node));
end
return sorted(sentence_scores)
```

vectors rather than Euclidean distance due to the high dimensionality of the data. Cosine similarity measures the cosine of the angle between two vectors, whereas the Euclidean distance calculates the length of the line segment that connects them. For example, two similar vectors, $the\,man\,is\,good$ and $the\ king\ is\ just$, could be far apart from each other in distance, but this may only due to the different context they are used in. However, the result of the cosine similarity is the angle between the two vectors, so it will take into account more the similarity of specific indices of each vector.

It may be simpler to think first about word embeddings. The same technique can be used to produce vectors for words. Translating words to vectors opens up the possibility of performing vector operations on words. For example, one can produce the vectors for king, queen, woman, and man. Then the vector resulting from the operation $king - (man + woman)$ would be relatively close to $queen$. As well as simple addition and subtraction, the same similarity calculations for sentence embeddings can be performed in word embeddings.

4 Implementation

All of the programs were written in Python 3.6, using the following packages: NetworkX [8], PRAW [9], NLTK [10], and NumPy [11]. The codes can be found at the authors' GitHub repository [12].

5 Results

These two algorithms have been tested on three separate corpora composed in three different styles. The first is the novel *Moby-Dick* by Herman Melville, which is a semi-formal fictional novel. The second is a corpus of Reuters news articles. The third, and last corpus, is informal posts on the Reddit subreddit, "Legal Advice." The main purpose of testing the algorithms on three separate corpora was to determine how each algorithm will act in different use cases. Section 5.1 presents the summary results of *Moby-Dick*, Section 5.2 presents the summary results of Reuters news articles, and Section 5.3 presents the summary results of Legal Advice subreddit. Note that each summary contains five sentences as a result.

- On the contrary, passengers themselves must pay.
- Whaling voyage by one ishmael."
- For to go as a passenger you must needs have a purse, and a purse is but a rag unless you have something in it.
- Why upon your first voyage as a passenger, did you yourself feel such a mystical vibration, when first told that you and your ship were now out of sight of land?
- Right and left, the streets take you waterward.

Figure 2: Summary of Chapter 1 of *Moby-Dick* using TF-IDF

5.1 Moby-Dick

The complete work of *Moby-Dick* was obtained through the NLTK package [13]. This novel was then split into chapters and each chapter was summarized. The corpus that was used contained all 135 chapters and approximately 253,129 words. Figure 2 shows the summary result of Chapter 1 of *Moby-Dick* using TF-IDF, and Figure 3 shows the summary result of Chapter 1 of *Moby-Dick* using TextRank.

5.2 Reuters News Articles

The second round of testing was done on another corpus from NLTK [13]. The corpus contains 10,000 news articles from Reuters. These articles combine works from different writers which produce interesting results. However, they are all achieving the same tone and follow Reuters style guide. Figure 4 shows the summary result of Reuters news articles using TF-IDF, and Figure 5 shows the summary result of Reuters news articles using TextRank.

5.3 Legal Advice Subreddit

The last corpus is a collection of posts from a site called Reddit. Reddit is a website that allows the general public to post within forums dedicated to specific topics. The forum chosen for this corpus is called "Legal Advice". This forum contains posts by users describing situations for which they believe they need advice from legal counsel. Using the Python package PRAW [9], the top 1,000 most liked posts of the last 30 days were retrieved, and then the posts long enough to warrant a summary were passed through the algorithms. Figure

- Deep into distant woodlands winds a mazy way, reaching to overlapping spurs of mountains bathed in their hill-side blue.
- Why did the poor poet of tennessee, upon suddenly receiving two handfuls of silver, deliberate whether to buy him a coat, which he sadly needed, or invest his money in a pedestrian trip to rockaway beach?
- Well, then, however the old sea-captains may order me about – however they may thump and punch me about, I have the satisfaction of knowing that it is all right; that everybody else is one way or other served in much the same way – either in a physical or metaphysical point of view, that is; and so the universal thump is passed round, and all hands should rub each other's shoulder-blades, and be content.
- But wherefore it was that after having repeatedly smelt the sea as a merchant sailor, I should now take it into my head to go on a whaling voyage; this the invisible police officer of the fates, who has the constant surveillance of me, and secretly dogs me, and influences me in some unaccountable way – he can better answer than any one else
- With other men, perhaps, such things would not have been inducements; but as for me, I am tormented with an everlasting itch for things remote.

Figure 3: Summary of Chapter 1 of *Moby-Dick* using TextRank

- New crop sales were also light and all to open ports with June/July going at 1,850 and 1,880 dlrs and at 35 and 45 dlrs under New York july, Aug/Sept at 1,870, 1,875 and 1,880 dlrs per tonne FOB.
- March/April sold at 4,340, 4,345 and 4,350 dlrs.
- April/May butter went at 2.27 times New York May, June/July at 4,400 and 4,415 dlrs, Aug/Sept at 4,351 to 4,450 dlrs and at 2.27 and 2.28 times New York Sept and Oct/Dec at 4,480 dlrs and 2.27 times New York Dec, Comissaria Smith said.
- Cake sales were registered at 785 to 995 dlrs for March/April, 785 dlrs for May, 753 dlrs for Aug and 0.39 times New York Dec for Oct/Dec.
- Liquor sales were limited with March/April selling at 2,325 and 2,380 dlrs, June/July at 2,375 dlrs and at 1.25 times New York July, Aug/Sept at 2,400 dlrs and at 1.25 times New York Sept and Oct/Dec at 1.25 times New York Dec, Comissaria Smith said.

Figure 4: Summary of a news article using TF-IDF

- There are doubts as to how much of this cocoa would be fit for export as shippers are now experiencing difficulties in obtaining +Bahia superior+ certificates.
- March/April sold at 4,340, 4,345 and 4,350 dlrs.
- April/May butter went at 2.27 times New York May, June/July at 4,400 and 4,415 dlrs, Aug/Sept at 4,351 to 4,450 dlrs and at 2.27 and 2.28 times New York Sept and Oct/Dec at 4,480 dlrs and 2.27 times New York Dec, Comissaria Smith said.
- Cake sales were registered at 785 to 995 dlrs for March/April, 785 dlrs for May, 753 dlrs for Aug and 0.39 times New York Dec for Oct/Dec.
- Liquor sales were limited with March/April selling at 2,325 and 2,380 dlrs, June/July at 2,375 dlrs and at 1.25 times New York July, Aug/Sept at 2,400 dlrs and at 1.25 times New York Sept and Oct/Dec at 1.25 times New York Dec, Comissaria Smith said.

Figure 5: Summary of a news article using TextRank

- I will just be saying MY home or MY driveway so I don't have to keep typing "my parent's driveway" or "my parent's home" over and over.
- My parent's neighbor's kid (a very immature 20 year old) has a beater he leaves parked in front of my parent's front yard.
- Now the parents are retaliating too.They finally moved the beater, but only to move their cars from the driveway to taking up the two spaces in front of our yard adjacent to their driveway.
- The one car parked just enough to have the front poking into our driveway.
- TLDR - neighbors parked all their cars in front of my parents home and wont move them, only rearrange them.

Figure 6: Summary of a Reddit posts using TF-IDF

6 shows the summary result of Reddit legal advice posts using TF-IDF, and Figure 7 shows the summary result of Reddit legal advice posts using TextRank.

6 Conclusion

Both TF-IDF and TextRank algorithms produced a much more representative summary of the Reddit posts, while they performed decently with the fictional works and the formally styled news articles. This disparity between summary quality may be because the Reddit posts are focused only around one situation that happened, whereas in both *Moby-Dick* and the news articles, the reader is presented with a background story and sometimes a story arc. The algorithms presented for extractive summarization do not perform optimally when presented with a story line. However, if a summary needs to

- These neighbors moved in a year or two ago and have made life so uncomfortable for my parents they are actually talking about selling their home to move, their marriage home with ALL the memories.
- He has it parked in the middle so that it takes up ALL the space and no one can park on either side of it without blocking a driveway.
- I assumed mom would just tell 'Billy, go move your damn car' or something and it would be taken care of.
- I get the dad yelling at me to *** off and get off his property as the mom (from another room) starts bellowing about how I did NOT just tell her how to parent and he can do whatever he wants and **** me and my parents.
- Not only do they say the street parking is technically public parking and they can't officially complain about it, but they complain the neighbors will only retaliate worse.

Figure 7: Summary of a Reddit posts using TextRank

be generated for a news article focused on one specific topic or a short story focused on one situation, both TF-IDF and TextRank are good choices.

It seems that TextRank has a tendency to favor longer sentences. This could be because longer sentences have more space to include any buzzwords. It can be offset by scaling the similarity score for each sentence by the length, just like the TF-IDF algorithm, although this behavior may be preferred for a more complex corpus.

In the future, abstractive summarization will be explored to determine an algorithm that returns a summary by utilizing natural grammar structures. This will be done with the aim of producing more representative summaries, specifically for *Moby-Dick* and the Reuters articles. And, the usefulness of neural networks, specifically long short-term memory neural networks (recurrent neural network) [14], will be tested to increase the performance. In addition, evaluation methods of text summarization will be investigated to measure the quality of the summaries.

References

[1] T. Rosenstiel, J. Sonderman, K. Loker, M. Tran, T. Tompson, J. Benz, D. Junis, The personal news cycle, *Chicago, The Media Insight Project*.

[2] H. P. Luhn, The automatic creation of literature abstracts, *IBM Journal of research and development*, *2*(2):(1958), 159–165.

[3] S. Bajaj, Shakespeare in one hundred words, *CS224N/Ling284 Final Projects 2013-2014*.

[4] D. Das, A. F. Martins, A survey on automatic text summarization, *Literature Survey for the Language and Statistics II course at CMU*, *4*:(2007), 192–195.

[5] J. Beel, B. Gipp, S. Langer, C. Breitinger, Research-paper recommender systems : a literature survey, *International Journal on Digital Libraries*, *17*(4):(2016), 305–338, ISSN 1432-5012, doi:10.1007/s00799-015-0156-0.

[6] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web., Technical report, Stanford InfoLab, 1999.

[7] R. Mihalcea, P. Tarau, Textrank: Bringing order into text, in *Proceedings of the 2004 conference on empirical methods in natural language processing* (2004).

[8] A. Hagberg, P. Swart, D. S Chult, Exploring network structure, dynamics, and function using networkx, Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[9] j. Bryce Boe, et al., PRAW: The python reddit api wrapper, 2012–, [Online; accessed Feb 2nd 2019].

[10] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, P. Blunsom, Teaching machines to read and comprehend, in *Advances in Neural Information Processing Systems*, pages 1693–1701 (2015).

[11] T. Oliphant, NumPy: A guide to NumPy, USA: Trelgol Publishing, 2006–, [Online; accessed Feb 13th 2019].

[12] A. Day, Atuomatic extractive summarization, https://doi.org/10.5281/zenodo.2563845, 2019.

[13] S. Bird, E. Klein, E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit* (" O'Reilly Media, Inc.", 2009).

[14] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, Y. Shi, Spoken language understanding using long short-term memory neural networks, in *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 189–194 (IEEE, 2014).

DATA PREPROCESSING AND FEATURE SELECTION FOR AN INTRUSION DETECTION SYSTEM DATASET

Zachary Groff, Stephanie Schwartz
Millersville University
{zmgroff, stephanie.schwartz}@millersville.edu

ABSTRACT
Automated intrusion detection is a necessity in today's business environment with the frequency of data breaches. These Intrusion Detection Systems detect when there is a network intrusion and report it to supervising security professionals. The automation of this detection allows critical resources within a business to be utilized elsewhere. There are currently a number of factors that hinder intrusion detection performance including a lack of publicly available datasets for examination and the substantial number of false positives and false negatives reported by intrusion detection systems. This project explores the possibility of reducing false negatives and reducing or maintaining false positives reported by intrusion detection systems through determining which network traffic features are the most reliable indicators of an intrusion. Data cleaning and preprocessing (key stages in the machine learning process) with the selected dataset are thoroughly covered along with the feature selection process.

KEY WORDS
MACHINE LEARNING, INTRUSION DETECTION, FEATURE SELECTION

1. Introduction

As the cyber security threat landscape expands and our dependence on technology grows, so does our need for automated intrusion detection. It is no longer feasible to employ a security team to both manually detect and prevent intrusions. Intrusion Detection Systems, or IDSs, are now commonly used by companies to detect security breaches. An IDS is primarily a monitoring system used to notify a security professional when an intrusion is detected. A well configured and efficient IDS has the potential to save a company significant amounts of money by assisting in preventing data breaches and allowing resources to be freed. Research on the impact of data breaches highlights this need for companies to be proactive in ramping up security. In 2018, the average cost to a company for a data breach was $3.86 million and the cost is increasing at over 6% per year according to IBM's 2018 report [1]. IBM also found that a single stolen record cost $146 on average. When considering the fact that many companies have millions of records, it's easy to understand the need for better IDS technology.

Dorothy Denning is widely credited with creating the first real-time IDS model that most modern applications follow [2]. The model described by Denning monitors and logs all actions performed on a system before checking for malicious activity in those logs using rule-based pattern matching. Modern Intrusion Detection Systems have deviated from Denning's model in implementation, but remain largely the same in structure and purpose. Denning's model is described as host-based and anomaly-based. *Host-based* models monitor actions and processes on the host computer before evaluating them as normal or malicious whereas *network-based* models monitor network traffic and attempt to pick out malicious traffic. An *anomaly-based* IDS operates by detecting deviation from a baseline of normal user activity, while *signature-based* IDSs look for specific patterns, or signatures, in network traffic that are commonly used by malware. Unfortunately, both signature and anomaly detection have their pitfalls.

Signature-based detection systems must be frequently updated with new signatures, which is a daunting task considering millions of different malware samples are collected every year [3]. A reliance on pre-existing signatures also results in poor performance when new malware is introduced. Anomaly-based systems outperform signature-based systems when introduced to new malware as all malware is compared to the same baseline of user activity or traffic.

Anomaly-based IDSs can experience a significant number of false positives and false negatives, which reduce the overall effectiveness of automated intrusion detection. False negatives refer to when an IDS falsely determines that network traffic or user activity is not malicious. A false positive occurs when the IDS incorrectly detects malicious activity. As the main goal of an IDS is to alert a security professional when malicious activity is detected, false positives and false negatives are a significant performance metric which must be reduced. An IDS reporting a substantial number of false negatives is essentially allowing malicious activity to fly under the radar, while a significant number of false positives will drive up operational labor costs and mask urgent security events [4]. With the significant increase in data breaches

in recent years, wasting valuable time by pursuing false positives is an issue that cannot be ignored.

The overall purpose of our project is to explore the existing machine learning efforts in IDS with a specific aim of reducing false positives (without, of course, increasing false negatives). This paper will first explain the process of selecting a dataset appropriate for this exploration as well as the preprocessing steps required for the selected dataset. This will be followed by an introduction to feature analysis and selection, an explanation of our chosen feature selection methods, and the results of performing initial feature analysis on the selected dataset. Future work will then be proposed and discussed.

2. Datasets

To evaluate the performance of various IDS models, it is crucial to have a benchmark dataset. IDS datasets are typically network traffic flows that are generated in a simulated environment and labelled as benign or malicious so that supervised machine learning methods can be applied. Network traffic flows are simply a summary of packets sent between two hosts during a connection. These flows are typically terminated by a TCP FIN flag sent by one of the hosts or a UDP timeout. Datasets are typically split into a training dataset and a testing dataset. This step must be done before training a machine learning model because the model needs to be tested on data that it has not previously seen. Otherwise the model will not be tested on its ability to classify new records, but on its ability to memorize previously seen records. Due to the complexity of generating such a dataset and the resources required to do so, there are a very limited number of datasets that are both publicly available and free. The pool of potential datasets for this experiment is reduced even further when additional criteria, such as feature count and data quality, are taken into consideration. This section will discuss a historically relevant dataset, a recent improvement on that dataset, and, finally, the selected dataset for this experiment.

The KDD Cup 1999 dataset [5] was generated in a military network environment for use in a data mining competition at the KDD-99 (Knowledge Discovery and Data Mining) international conference. The goal of this competition was simply to build a network-based IDS to classify records as benign or malicious. The full dataset consists of benign traffic along with 38 different attacks (14 of which are only found in the test data), and 41 features. The attacks, which are used as labels, can be categorized into four distinct groups. *Denial of Service*, or DoS, attacks are any attack that aims to prevent a host from servicing clients either temporarily or permanently. *Probe attacks* attempt to perform reconnaissance on a host. *Remote to user*, or r2l, attacks aim to grant the attacker user-level access on a host machine. *User to root*, or u2r, attacks attempt to escalate local privileges on the host machine from a normal user to root access. The dataset's features can also be separated into three fields. These fields are TCP connection features, system-level features gathered during a connection to the host, and network traffic features gathered over a two second interval.

Despite its age, KDD Cup '99 has acted as the de facto benchmark dataset for IDS evaluation since its inception mostly because the dataset is publicly available and free. Researchers from the University of New Brunswick have shown the dataset has many flaws [6]. The KDD dataset was found to have a significant number of duplicate samples: 78% in the test set and 75% in the training set. Many machine learning algorithms used on this dataset can become biased as a result of seeing the same sample multiple times in the training set. Additionally, testing results become skewed because the test set only contains a small portion of unique data. The dataset also has substantial class imbalance, with just two DoS attacks accounting for 71% of the data in the training set [7]. Between the duplicate samples and class imbalance, the KDD Cup '99 dataset is not an ideal dataset for benchmarking Intrusion Detection Systems.

The previously mentioned researchers from the University of New Brunswick created a revised dataset, NSL-KDD [8], with the flaws of KDD Cup '99 in mind. NSL-KDD is essentially KDD Cup '99 with the redundant data removed. An additional feature, difficulty, is also added to the dataset. This difficulty measure was determined by having 21 different machine learning models classify each record. The difficulty of a record is simply the number of times the record was correctly classified by the 21 models. The number of records in NSL-KDD was further reduced by selecting records from each difficulty such that the number of selected records is inversely proportional to the percentage of records classified accurately by the 21 models. The removal of duplicate records and selection of records based on difficulty trims the train set from 4,898,431 records to 124,972 distinct records and the test set from 311,027 records to 22,544 distinct records.

NSL-KDD is certainly an improvement over the KDD Cup '99 dataset, but it is still far from ideal. The improved dataset has two significant deficiencies that carry over from the original dataset. One of these deficiencies is that the dataset is simply outdated as most of the attacks represented in the dataset have long since been patched. According to OWASP's 2017 report [9], SQL Injection, or SQLi, is the most exploited vulnerability in web applications. As such, any modern IDS dataset should include SQLi traffic and the NSL-KDD dataset does not. However, it is to be expected that a modification of a 20-year old dataset would be outdated. The more pertinent deficiency in NSL-KDD is the *class imbalance* (which will be defined and discussed as it

relates to the selected dataset for this experiment in the preprocessing section). Of the 24 classes found in this dataset, 10 classes have 50 or fewer records between both the train and test set. Additionally, approximately 85% of the records are split between the *normal* (or benign) and *neptune* classes. The NSL-KDD test set has 8 classes with fewer than 10 records, and 2 classes actually don't have a single record in the test set. The disproportionate spread of records across the 24 classes found in this dataset makes it far from ideal as a benchmark dataset for IDS model comparisons.

Due to the issues with both KDD and NSL-KDD, the selected dataset for this experiment is CICIDS2017 [10], referred to as IDS2017 throughout the remainder of this paper. IDS2017 was selected as the dataset for this experiment because of the vast number of network traffic features, more than 80, and the relevance of its contained attacks in the current state of information security. The dataset was created by researchers from the University of New Brunswick. PCAPs, or packet captures, were generated over the course of the five days and network traffic features were extracted from said PCAPs using CICFlowMeter [11]. The details of the environment on which the PCAPs were taken are outlined in the related research paper by Iman Sharafaldin et al. [12]. The researchers' focus on generating realistic background traffic should provide a dataset with records that mimic a live environment more accurately.

3. Data Cleaning

According to the 2016 CrowdFlower data science report [13], 60% of surveyed data scientists spent more time cleaning their data than any other task. Data cleaning, often referred to as data wrangling, is the act of modifying, formatting, and occasionally removing data from a dataset in order to make the dataset more usable. This process took the majority of the time spent in this experiment as many modifications needed to be made to the IDS2017 dataset before it was usable for the purposes of feature analysis.

During initial observations, it was discovered that the dataset contains two columns named "Fwd Header Length1" and "Fwd Header Length2" that contained identical data. One of these columns was removed and the other was renamed to "Fwd Header Length". This saved a significant amount of memory and disk usage. When calculating feature columns "Flow bytes/s" and "Flow packets/s", NaN and Infinity values were generated in nearly 5000 records. This occurred as a result of the "Flow duration" column having a value of 0. The simplest solution was simply to remove these records, which were mostly seen in the benign and DoS-Hulk classes. The CSV header, the first row in a CSV file which lists the names of the columns, was not properly delimited. Typically, each column name in the header is separated by a comma or a comma followed by a space. The headers in the provided dataset files mixed these conventions in a seemingly random way. The obvious fix was to simply remove the spaces that trailed commas.

The "Thursday-WorkingHours-Morning-WebAttacks" file was found to have 288,602 rows that were missing data. This left each of these rows consisting of more than 80 commas. After removing these rows, the true number of records in this CSV was 170,367. A significant amount of time was also spent debugging a character encoding problem in this same file. It was discovered that a non-utf-8 byte, 0x96 specifically, had been used in place of a hyphen in various class names. For example, the label "Web Attack - Brute Force" was instead "Web Attack \0x96 Brute Force". This was seen in multiple instances, seemingly at random. As a solution, each of these bytes were simply replaced with a hyphen.

A final discovery made (and one that epitomizes the inconsistency of this data) was that records in the "Monday-WorkingHours" CSV, a file that contains benign traffic exclusively, did not follow the same date format as the other records. The day and month in this file had leading zeroes, such that the value was "03/07/2017". For every other CSV file in this dataset, the format would have been "3/7/2017". This was certainly not significant, but it did require cleaning as all CSV files were concatenated to create a single CSV for this experiment.

By themselves, none of the inconsistencies in this dataset were time consuming to fix after they were discovered. However, the time spent debugging exceptions thrown in various analysis scripts as a result of these inconsistencies was far from insignificant. Data cleaning is step that must be taken before working with almost any dataset and spending time performing the data science equivalent of janitorial work simply must be done.

4. Normalization

Machine learning encompasses far more than selecting the proper model and feeding it training data. Data preprocessing, in which raw data is transformed to a more usable format, is often a long and tedious process that must be performed before training a model. Data normalization is one crucial step in the pre-processing phase in which a dataset that has columns varying in scale is normalized, or transformed into the same scale.

Data that has been normalized allows a model to be trained in a shorter time span and often results in higher prediction accuracy. The decrease in training time occurs in neural networks specifically because neural networks initialize weights as random values in a range of [-1, 1]. The back-propagation process, which modifies these weights, has to perform more computation to calculate how much each weight needs modified if the data is on a

different scale than the activation function. If a neural network is trained with features varying greatly in scale, it is possible that the feature with a larger scale will dominate the feature with a smaller scale. For instance, if the values in feature column x have a range of [100,000, 250,000] and feature column y has a range of [-1, 1], it is possible for the optimizer to focus on column x and ignore the data in column y when updating weights in the back-propagation process.

There are various methods for normalizing input data. Standardization scaling is one such method, in which each value is subtracted by the mean of the column and then divided by the column's standard deviation. In median normalization, each column value is transformed by dividing each value by the median of the column. More complex functions for data normalization are described in the work of S.C. Nayak, B.B. Misra, and H.S. Behera [14].

Min-Max normalization is perhaps the most common method and the method used in this experiment for the sake of simplicity. In min-max normalization, all values are transformed linearly to be in the range [0, 1] or [-1, 1]. Consider a column with values [1,2,3], a min-max normalization algorithm normalizing this column to a range of [-1,1] will transform the values in this column to [-1,0,1]. The implementation of min-max normalization used the MinMaxScaler class from the sklearn python package [15]. Figure 1 and figure 2 show an example of how five feature columns from the IDS2017 dataset were transformed with this implementation.

Max Packet Length	Average Packet Size	Avg Fwd Segment Size	Avg Bwd Segment Size	Fwd Header Length
163.00	31.13	15.64	65.20	368.00
1575.00	393.75	315.00	525.00	336.00
351.00	52.14	91.25	0.00	136.00
351.00	52.00	91.00	0.00	136.00
7240.00	994.75	57.00	1932.50	200.00

Figure 1 - Records before normalization

Max Packet Length	Average Packet Size	Avg Fwd Segment Size	Avg Bwd Segment Size	Fwd Header Length
-1.0000	-1.0000	-1.0000	-0.9325	1.0000
-0.6009	-0.2473	1.0000	-0.4566	0.7241
-0.9468	-0.9563	-0.4949	-1.0000	-1.0000
-0.9468	-0.9566	-0.4965	-1.0000	-1.0000
1.0000	1.0000	-0.7236	1.0000	-0.4482

Figure 2 - Records after normalization

5. Class Representation

Intrusion detection and the field of information security are further complicated by the infrequency of attack indicators in network traffic. As such, it makes sense for an IDS dataset to have an overwhelming percentage of benign traffic. This is true of the IDS2017 dataset, which is approximately 80.339% benign traffic. Live network traffic is almost exclusively benign, but that does not necessarily mean an IDS dataset should be. A dataset with disproportionate class representation can become biased towards overrepresented classes. This is often referred to as class imbalance and is the most significant problem with IDS2017.

For a dataset with almost three million records and only fifteen classes, IDS2017 has significant class imbalance. IDS2017 has approximately 19.661%, or 556,556, records spread between fourteen classes. An even distribution would leave each class with nearly 40,000 records. Figure 3 shows the true distribution of records across the non-benign classes in this dataset. With some classes having over 100,000 records and others having fewer than 100, class imbalance is glaringly obvious. With a generous split of 80%-20% between train and test data and rounding the number of test records up, the dataset is left with only eight Heartbleed records to train on and four to test on. This is not nearly enough data to properly train or accurately test a machine learning model.

There are two methods to overcome class imbalance, *oversampling* and *undersampling*. Undersampling an overrepresented class is one method of overcoming class imbalance in data by gathering a subset of the overrepresented, or majority, class. Japkowicz discusses two simple ways to undersample a majority class [16]. The first is referred to as *random sampling* and is simply creating a subset of the majority class by randomly sampling the majority class. The second method is referred to by Japkowicz as *focused sampling*. This method involves creating a subset of the data by eliminating outliers from the majority class. Both of these methods have their faults, a significant one being that both methods have the potential to remove important examples of a traffic flow that indicates a specific class. More complex undersampling techniques are discussed by Chawla [17].

Oversampling can be used to create a larger dataset for an underrepresented, or minority, class.
Random oversampling [18] is the simplest oversampling method. In random oversampling, randomly selected data from a minority class is duplicated to increase the size of the minority class. *Synthetic Minority Over-sampling Technique*, or SMOTE [19], is another frequently used oversampling technique. SMOTE creates "synthetic" data using a sample from the minority class and one of its neighbors, determined by the k-nearest neighbors algorithm [20]. This is done by taking the difference

between the sample's feature value and its neighbor's feature value for each feature in the feature vector. A random number, between 0 and 1, is then generated and multiplied by the difference previously mentioned to give the new synthetic sample's value for that feature. Borderline-SMOTE builds on SMOTE and results indicate that it outperforms SMOTE by strengthening the border between the minority class and majority classes in the dataset [21].

DoS Hulk	**230,124**
DoS GoldenEye	**10,293**
Heartbleed	**11**
Web Attack Brute Force	**1,507**
Web Attack XSS	**652**
Web Attack SQLi	**21**
Infiltration	**36**
Bot	**1956**
DDoS	**128,025**
Port Scan	**158,804**

Figure 3 - IDS2017 class representation

SMOTE excels at creating noise, or slight variance in the feature vector across a class, to reduce overfitting in a minority class. Overfitting occurs when a machine learning model fits the boundary between classes too closely, effectively learning the dataset itself as opposed to the trends across classes. This results in the model experiencing a high training accuracy, but a low accuracy in predicting the class of new data. However, when the number of records is incredibly small, as in the case of the IDS2017 Heartbleed class, SMOTE is significantly less effective. When creating a synthetic record, SMOTE takes into account a pre-existing record and a randomly selected record from its 5 nearest neighbors. In the case of the Heartbleed class, which only has 11 records, many records share multiple neighbors and therefore when oversampling the class, a significant amount of duplicate data is created. This would be far less of a problem if the data being duplicated is significant in differentiating Heartbleed from the other classes, but with only 11 records there is a distinct possibility that trends are being created in the data for Heartbleed that are not indicative of the class in a live environment. This is less of a concern in the classes with significantly more data but is certainly a concern for the SQLi and Infiltration classes which have only 21 and 36 records respectively.

For this experiment, a combination of oversampling and undersampling techniques are used in an attempt to overcome the class imbalance in the IDS2017 dataset. The majority classes, those with more than 10,000 records, are undersampled using *random undersampling* to significantly reduce the number of records to the arbitrary value of 10,000 records. Oversampling is performed on the minority classes, those with fewer than 10,000 records, using the SMOTE implementation found in the imbalanced-learn Python package [22]. Each minority class has enough synthetic samples created to bring the number of records to 10,000. Using these techniques, a new dataset was generated with 150,000 records split evenly between the 15 classes. Future work will attempt to determine an optimal number of samples to reach for each class through oversampling and undersampling as well as a better method of oversampling the three classes with fewer than 50 samples.

6. Feature Analysis

In order to minimize the number of false positives and false negatives in a neural network-based IDS model, it is crucial to select the optimal set of features. Reducing the number of features used in the IDS2017 dataset will also increase the interpretability of the model and decrease the chance of overfitting by reducing the complexity. However, with the large number of classes in the IDS2017 dataset, this is no trivial task. Having to distinguish between five different denial-of-service attacks and three different web application attacks increases the difficulty of this task even further. Fortunately, there are numerous feature analysis methods that can be used to simplify this step. This section will discuss the features that were dropped or modified prior to the feature analysis tasks, the algorithms used, and the various tests that were performed.

Two features, FlowID and Timestamp were immediately removed before the feature analysis process began. The timestamp feature was dropped due to the nature of how the data was gathered. Only a subset of classes is contained in the gathered data for each day. Distributed-denial-of-service, or DDoS, attacks are only seen with the timestamp "7/7/2017" for example. This could lead the model to incorrectly correlate that date with DDoS attacks. An additional feature, FlowID, was also removed from the dataset. FlowID is calculated by concatenating source IP, destination IP, source port, destination port, and protocol in the format "SourceIP-DestinationIP-SourcePort-DestinationPort-Protocol". This field is redundant since it can be calculated from other fields and would further complicate the model without providing additional information.

An additional three features, source IP, destination IP, and protocol, in the IDS2017 dataset were modified before feature analysis tests were performed. The protocol feature was observed to have only three possible values in this dataset. These values are 0, 6, and 17 denoting the HOPOPT, UDP, and TCP protocols respectively. With the small number of possible values for this feature, the decision to one-hot encode the feature was made. One-hot encoding a feature is an operation that creates a new feature for each unique value. These new features are Boolean values in practice that indicate whether the unique value is or is not present. Figure 4 shows the result of one-hot encoding the protocol column for three

records. The source and destination IP address fields have also been modified using the python ipaddress library [23]. This library allows IP addresses to be converted into an integer format. An IP "192.168.0.1" is converted into the integer value "3232235521", or $192 * 256^3 + 168 * 256^2 + 0 * 256^1 + 1 * 256^0$. This transformation is required to allow the classifier to interpret these IP features.

Protocol	HOPOPT	UDP	TCP
0	1	0	0
7	0	1	0
17	0	0	1

Figure 4 - Protocol one-hot encoding

The Boruta [24] algorithm is the first used in the feature analysis stage of this experiment. This algorithm is implemented using a *random forest*. A random forest is composed of numerous decision trees that are trained individually using different samples from the dataset. The importance of a feature is calculated in this algorithm by adding additional randomly selected features to the dataset for each tree and measuring the decrease in classification accuracy that occurs from adding this noise. Shadow attributes are also added to the dataset before measuring classification accuracy in each tree. These shadow attributes are computed by taking the values of a feature column and shuffling them. This allows the algorithm to use the set of shadow feature importance as a reference to determine which features are truly significant to the classification decision. This process is repeated many times to give a statistically accurate conclusion of feature importance. All features with an importance score that meets a given threshold are given the rank 1 while remaining sorted by the original importance score. These are still ordered by importance score. All other features are then ranked by importance score beginning with rank 2.

Recursive Feature Elimination, RFE, is the second feature analysis algorithm used. This algorithm can be found in sklearn's feature_selection module [25]. RFE works with numerous estimators, but a random forest classifier was used in this experiment. The algorithm proceeds by recursively evaluating the importance of each feature in the feature set. After the importance for each feature is calculated, the algorithm removes a given number of the least important features. This process continues until all features have been eliminated, or a given number of features are selected. In this experiment, all features are eliminated so a full ranking of feature importance can be seen.

In our feature selection experiments, three different types of tests are run with both feature analysis algorithms. The results presented in each figure show the five most and least significant features in distinguishing between two classes. Fourteen separate datasets are generated, each of which consists of 1000 records from a non-benign class and 1000 benign records, using randomly selected records from IDS2017. Each of these datasets are then analyzed with RFE and Boruta to determine the most significant features in distinguishing each attack from benign traffic. Figure 5 presents the results of the Boruta and RFE algorithms for distinguishing the DoS-Hulk class from benign traffic. The second test attempts to determine the most significant features in distinguishing each DoS attack from the other DoS attacks. Four different datasets are generated for this test. Each of these generated datasets is composed of 15,000 records from one of the four DoS classes and 15,000 records from each of the other three DoS classes that have had their labels changed to "anomaly". This results in 60,000 records, three quarters of which are labelled as "anomaly", in each of the four datasets. The results of running the DoS-Hulk dataset through the Boruta and RFE algorithms are presented in Figure 6. The final test performed attempts to distinguish benign traffic from malicious traffic. For this test, 14,000 randomly selected benign records and 1,000 randomly selected records from each of the other 14 classes form a dataset of 28,000 records. Each non-benign record has had its label changed to "anomaly". The results of running this dataset through the feature selection algorithms are presented in figure 7.

	Boruta	RFE
Feature 1	Source IP	Source IP
Feature 2	Source Port	Bwd Packet Length Std
Feature 3	Destination IP	Packet Length Variance
Feature 4	Destination Port	Packet Length Std
Feature 5	Flow Duration	Packet Length Mean

Figure 5 - Top 5 features for Benign vs. DoS Hulk

	Boruta	RFE
Feature 1	Source Port	Init_Win_Bytes_Fwd
Feature 2	Flow Duration	Flow IAT Mean
Feature 3	Total Fwd Packets	Fwd IAT Min
Feature 4	Total Bwd Packets	PSH Flag Count
Feature 5	Fwd Packet Len	Avg Packet Size

Figure 6 - Top 5 features for DoS Hulk vs. DoS Group

	Boruta	RFE
Feature 1	Source Port	Flow Duration
Feature 2	Fwd Packets/s	Source Port
Feature 3	Flow Packets/s	Flow IAT Mean
Feature 4	Flow Duration	Average Packet Size
Feature 5	Bwd Packets/s	Flow IAT Max

Figure 7 - Top 5 features for Benign vs. Anomaly

7. Future Work

The research presented in this paper constitutes the beginning stages of an investigation into whether or not an emphasis on feature selection can reduce the number of false positives while maintaining or improving the false negatives reported by neural-network based IDS. However, there are still a number of alternative processes to explore before results will be evaluated. SMOTE and random undersampling were used exclusively to handle class imbalance in this experiment. There are a variety of additional oversampling and undersampling techniques that will be explored in the future. A min-max scaler is used to scale the feature values to a range of [-1, 1] in the normalization phase of this work and while this seems to be the best method at first glance, there may be better ways to normalize data for this purpose. Finally, as a number of features in the IDS2017 dataset seem to be calculated exclusively with other features, collinearity tests can be performed to eliminate redundant features. The elimination of these features will simplify the process of selecting ideal features and evaluating the performance of a neural network-based IDS.

Significant class imbalance makes many of the classes in the IDS2017 dataset unusable for this research. Two possibilities exist moving forward. The massively underrepresented classes in this dataset could simply be ignored in future work. There are a total of eight classes with more than 5,000 records in this dataset that would still be used. It is worth noting that this may not be ideal as more than half of those classes are variations of DoS attacks. Alternatively, a different dataset could be used. There are few IDS datasets that are free, publicly available, and contain a variety of modern attacks, but exploring the possibility of using another dataset may be worthwhile. Generating an IDS dataset for future work is not outside the realm of possibility, although it is probably not feasible in the current time frame for the project. Regardless, it's safe to say that improvements could still be made to the body of existing datasets for this domain.

8. Conclusion

This paper details the preliminary stages of a larger work, which attempts to determine if careful feature selection can decrease the number of false positives and false negatives reported by neural network-based intrusion detection models. Commonly used intrusion detection datasets are introduced and discussed, with faults in these datasets highlighted. The process of cleaning the IDS2017 dataset is outlined and the steps taken to overcome the numerous deficiencies with this dataset are provided. Preliminary results of various feature analysis tests are presented. Future work will build on the processes and results presented in this paper in an effort to answer the overarching question of feature selection's role in reducing IDS false positives and false negatives.

References:

[1] Anon. 2018. 2018 Cost of a Data Breach Study: Global Overview. (2018). 2018_Global_Cost_of_a_Data_Breach_Report.pdf

[2] Dorothy E. Denning. An Intrusion-Detection Model. Retrieved January 13, 2019 from https://dl.acm.org/citation.cfm?id=22862

[3] Iman Sharafaldin, Amirhossein Gharib, Arash Habibi Lashkari, and Ali A. Ghorbani. 2017. Software Networking: Towards a Reliable Intrusion Detection Benchmark Dataset. (July 2017).

[4] Huseyin Cavusoglu, Birendra Mishra, and Srinivasan Raghunathan. 2005. The Value of Intrusion Detection Systems in Information Technology Security Architecture. (March 2005).

[5] Anon.Retrieved February 12, 2019 from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[6] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali Ghorbani. 2009. A Detailed Analysis of the KDD CUP 99 Data Set. (2009). https://www.ee.ryerson.ca/~bagheri/papers/cisda.pdf

[7] Kingsly Leung and Christopher. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. http://crpit.com/confpapers/CRPITV38Leung.pdf

[8] Anon. Search UNB. Retrieved February 12, 2019 from https://www.unb.ca/cic/datasets/nsl.html

[9] Anon. The Ten Most Critical Web Application Security Risks. Retrieved 2017 from https://www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf.pdf

[10] Anon. 2017. Search UNB. (2017). Retrieved February 12, 2019 from https://www.unb.ca/cic/datasets/ids-2017.html

[11] Anon. Search UNB. Retrieved February 12, 2019 from https://www.unb.ca/cic/research/applications.html#CICFlowMeter

[12] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

[13] Anon. 2016. 2016 Data Science Report. (2016). https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf

[14] S.C. Nayak, B.B. Misra, and H.S. Behera. 2014. Impact of Data Normalization on Stock Index Forecasting. (2014).

[15] Anon. sklearn.preprocessing.MinMaxScaler. Retrieved February 12, 2019 from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[16] Nathalie Japkowicz. The Class Imbalance Problem: Signicance and Strategies. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=4C632BB460493D3DA9DEACC8F7EEDF88?doi=10.1.1.35.1693&rep=rep1&type=pdf

[17] Nitesh V. Chawla. Data Mining for Imbalanced Datasets: An Overview. https://www3.nd.edu/~dial/publications/chawla2005data.pdf

[18] Samrat J. Dattagupta. A Performance Comparison of Oversampling Methods for Data Generation in Imbalanced Learning Tasks. https://run.unl.pt/bitstream/10362/31307/1/TEGI0396.pdf

[19] Nitesh Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W.Phillip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. (2002). https://arxiv.org/pdf/1106.1813.pdf

[20] Altman, N.S.. 1992. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. American Statistician https://www.researchgate.net/publication/238181607_An_Introduction_to_Kernel_and_Nearest-Neighbor_Nonparametric_Regression

[21]Hui Han, Wen-Yuan Wang, and Bing-Haun Mao. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced ... Retrieved February 12, 2019 from https://sci2s.ugr.es/keel/keel-dataset/pdfs/2005-Han-LNCS.pdf

[22] Scikit-Learn-Contrib. 2019. scikit-learn-contrib/imbalanced-learn. (January 2019). Retrieved February 12, 2019 from https://github.com/scikit-learn-contrib/imbalanced-learn

[23]Anon. ipaddress - IPv4/IPv6 manipulation library. Retrieved February 12, 2019 from https://docs.python.org/3/library/ipaddress.html

[24] Miron B. Kursa and Witold Rudnicki. 2010. Feature Selection with the Boruta Package. (September 2010). https://www.jstatsoft.org/article/view/v036i11/v36i11.pdf

[25] Anon. sklearn.feature_selection.RFE. Retrieved February 12, 2019 from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

IMPLEMENTATION OF COGNITIVE RADIO TECHNIQUES USING TRADITIONAL RELAY NETWORK PRINCIPLES

Joshua Varone and Sangkook Lee
School of Engineering, Shippensburg University of Pennsylvania
{jv8864, slee}@cs.ship.edu

ABSTRACT
This paper outlines the background research necessary to understand several wireless networking concepts in the development of an experimental cognitive radio network based on the techniques of conventional networks. Further, it includes a discussion of the design and implementation of an initial simple relay network prototype. A discussion of cooperative diversity supports the value of this work, while a closer examination of relaying schemes and the measurement and analysis of wireless networks sets the stage for larger-scale implementation. An exploration of cognitive radio techniques describes the intended behavior of such networks and illustrates how such ideas enable the furthering of wireless network research. This review of related literature supports a greater understanding of necessary foundational knowledge and promotes the goal of building a prototype network based on cognitive radio techniques. As a first step towards this implementation, the design and development of a simple relay network is discussed and reviewed.

KEY WORDS
cognitive radio, cooperative diversity, relaying schemes

1. Introduction

The demand for radio frequency spectrum has drastically increased in recent years with the rapid deployment of new wireless devices and applications. As a result, the current fixed spectrum assignment policy has become a bottleneck for efficient spectrum utilization, as it becomes unevenly saturated. Cognitive radio looks to solve this problem by monitoring available frequency bands and adapting its transmission patterns to make the most efficient use of available radio resources.

Conventional wireless networks generally involve point-to-point communication links, which can suffer from unreliable transmission in less-than-optimal conditions. In contrast, cooperative wireless networks—specifically, in this case, relay networks—offer greater network reliability as other users in the network act as relays to support the transmission of data. Such networks enable the creation of a virtual antenna array, exploiting spatial diversity to improve overall system performance.

The intended goal of this research is to develop a prototype network based on cognitive radio techniques, using these ideas to understand and analyze the network in comparison to conventional wireless networks. The implementation of an initial small-scale relay network is a necessary step in establishing a baseline for performance comparison, and as such is valuable to discuss as well.

2. Literature Review

Many important concepts must be considered in exploring how cognitive radio techniques may be implemented. It is first necessary, however, to understand the background of traditional network and cognitive radio technologies.

The first section of this review will explore the idea of cooperative diversity and discuss its applications to improve wireless networks. The second section will discuss traditional categories of relaying schemes and the third will cover network measurement and analysis. The fourth section will examine cognitive radio techniques and their relationship to traditional concepts. Finally, concluding ideas and relevance to the intended area of research will be presented.

2.1 Cooperative Diversity

Traditional wireless networks are susceptible to performance degradation as a result of signal fading [1]. Such fading may result from multipath propagation, which describes a situation in which a signal is received via multiple different paths, potentially creating interference and causing issues with the reconstruction and interpretation of the signal. A significant approach that has been applied to combat this issue is diversity.

Space diversity describes the use of multiple antennas to cover a larger physical area and provide additional opportunities for successful signal transmission by alleviating some line-of-sight obstacles [2]. Other types of diversity exist as well, most notably time and frequency, where the former relates to the distribution of transmission over a wider range of time and the latter describes the use of multiple frequencies with the intention of improving transmission quality [5].

Cooperative diversity is considered a specific type of space diversity, in which the multiple antennas used are part of unrelated terminals, each operating independently with its own data. Under this scheme, remote systems will cooperate and share their resources with the target system, forming a virtual multiple-antenna system.

A major consideration in this area is the identification of motivating factors that would impel cooperating terminals to participate, and much of the work in this area has been devoted to describing mutual benefits between the primary system and secondary users [1].

In some circumstances, independent networks already listen to external transmissions, but often ignore data from those terminals. With cooperative diversity, these signals could be retained and forwarded to the appropriate endpoint. The sharing of resources between, for example, two networks has the potential to improve the connection quality for both networks as in situations in which frequency bands suffer from impairment, either network may be able to transmit via an alternate transmission path, supporting the motivation to pursue this idea.

With a goal of mimicking a physical multiple-antenna system, some related work has also explored the effectiveness of this virtual array in comparison to a traditional setup. In [6], it was determined that a virtual array as is generally utilized in cooperative setups will be limited in comparison to a physical multiple-antenna array in terms of multiplexing gain, which reflects the correlation between system rate and the signal-to-noise ratio (SNR). A higher multiplexing gain would indicate that the overall rate of transmission of the system would increase more strongly with increasing SNR.

2.2 Relaying Schemes

Cooperation between terminals and independent networks is often based on conventional relaying schemes, such as amplify-and-forward and decode-and-forward. A representation of conventional relay networks can be seen in Fig. 1. Under such relaying schemes, the transmitting terminals actively listen for cooperating nodes in addition to handling their own primary information, and in doing so can act as relay nodes for external cooperating networks [1]. It is further possible, as described in [2], to extend these relaying schemes to larger networks.

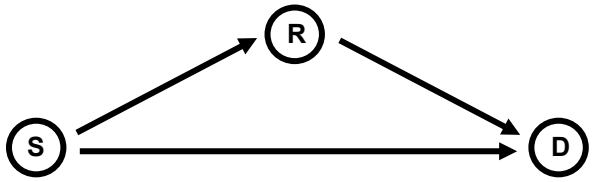


Fig. 1. Representation of the traditional relay network model, with a dedicated relay node (R) to support transmission from source (S) to destination (D).

In [1], three random coding schemes are identified as they relate to relay networks, each supporting the enhanced transmission of the target signal by assisting the primary link in varying ways:

- *Facilitation*: This scheme describes a scenario in which the relay node does not retransmit or otherwise strengthen the source signal, but instead attempts to minimize potential interference to support its successful transmission.
- *Cooperation*: This scheme describes a scenario in which the relay node receives the source signal, fully decodes the signal, and retransmits a reencoded signal to the destination. It is roughly analogous to the decode-and-forward scheme.
- *Observation*: This scheme describes a scenario in which the relay node receives the source signal and encodes and retransmits the signal without fully decoding. It is roughly analogous to the amplify-and-forward scheme.

As explored in [5], additional performance gains through the use of relaying, particularly with amplify-and-forward, are possible through an analysis of diversity order. It was concluded that the combination of correlated relay links may improve performance not through increased diversity but through coding gains resulting from increased repetition of the source signal. Further, it logically follows that any additional signal processing that could occur at the relays would support greater overall performance across the link [5].

2.3 Network Measurement and Analysis

The analysis of cooperative diversity schemes requires the understanding of several additional concepts. The signal-to-noise ratio, *SNR*, reflects the quality of the transmitted signal. Other necessary quantities and their notations include *W*, which represents allocated bandwidth for a given user or cooperative group; *R*, which represents the target transmission rate of a given user in bits per second; and *P*, which represents the transmission power per unit frequency in watts per hertz. The constant N_0 represents the "typical noise power density at room temperature" and is equal to 4.0 * 10-21 watts per hertz [6].

Capacity, *C*, represents the amount of information that can be transmitted over a given time. Calculations of capacity vary by the chosen cooperative protocol, but generally include consideration of the bandwidth and power for a given user, as well as the channel states (reflecting the connection quality within frequency bands), often represented as $|h_{t,r}|^2$ where *t* is the transmitting node and *r* is the receiving node, which are "independent complex Gaussian random variables with mean zero" and variances that depend on the distance between transmitting and receiving nodes in simulations [6]. When considering noncooperative communications, such a calculation is simplified as it requires only the consideration of the source and destination nodes. However, in cooperative cases, additional modifications

must be made to account for relay nodes, of which there may be one or more [6].

Often, additional consideration must be given to limitations for specific applications of cognitive radio techniques and cooperative diversity protocols. Limiting factors may include power, spectrum, or time, among others, and necessitate the analysis of optimal assignment for efficient performance and transmission. One example of such a constraint was analyzed in [3], where power efficiency was a primary concern since wireless networks often consist of limited-power wireless devices. Further, similar analysis can determine the optimal level of power usage subject to some other minimum requirement [3].

While accounting for limitations, such approaches typically also attempt to maximize some positive quantity. For example, in [6], the goal is to maximize the transmission rate of secondary users while minimizing power used by the primary user. This type of analysis is necessary to establish a cognitive radio scheme that efficiently solves an existing problem in the area of wireless communications.

2.4 Cognitive Radio

The traditional model of radio frequency spectrum allocation has assigned specific frequency bands to primary users who have sole control over that band. However, with such a setup, spectrum efficiency, measuring the total utilization of the overall spectrum, has suffered. This occurs because idle times, where the primary user is not transmitting data, result in lost opportunities. Cognitive radio has been a major research focus for its potential approach to reducing that issue of wasted spectrum: with the use of cognitive radio techniques, unused radio frequency spectrum, or idle times in a primary user's band, could be better utilized by secondary users to improve overall efficiency and allow a greater quantity of information to flow over the same, limited spectrum [6]. This idea is represented in Fig. 2.

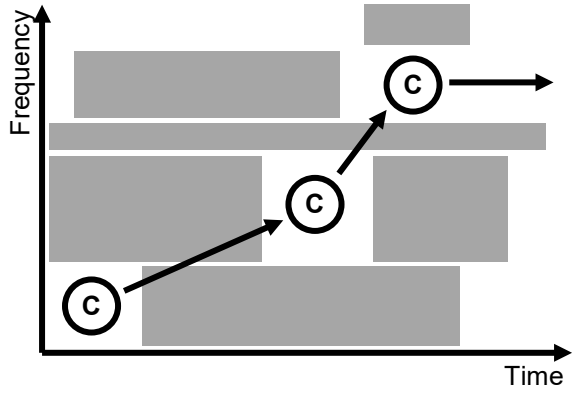


Fig. 2. Idealization of the cognitive radio concept, reflecting the utilization of idle (open) transmission blocks by secondary sources of information (C).

As an application of cooperative communication principles, cognitive radio has the potential to increase total network capacity while reducing power consumption and transmission latency [6]. By following some of the ideas laid out in cooperative diversity schemes, cooperating nodes within a cognitive radio network may share resources to promote better quality transmissions that are also more robust and resistant to outages.

A cognitive radio approach, considering both primary and secondary users, can utilize cooperative ideas in a few ways as described in [6]. First, cooperation can occur among primary users, which correlates to the traditional cooperative model as discussed in a previous subsection. Second, cooperation can occur among only secondary users, which again follows the traditional model, but is further enhanced by additional spectrum-sharing ideas. Finally, cooperation can occur between primary and secondary users. This final scenario represents the most complicated approach and the one that is most heavily investigated in related works. One representative model of this situation is shown in Fig. 3.

The model proposed in [6] and shown in Fig. 3 reflects a cooperative approach of the third type above. In this model, with total bandwidth of W and total time of T, the bandwidth is divided into W_1 and W_2, and the time into T_1 and T_2. For simplicity in analysis, T_1 and T_2 are each considered to be $T/2$. With this division of resources, the secondary user is granted spectrum W_2 across both time slots for its own data transmission. W_1 during time slot T_1 is retained by the primary user for its own data transmission, and W_2 during time slot T_2 consists of the secondary user relaying the primary user's data. Analysis of this structure reveals potential power savings with increased spectrum efficiency as more total data is transmitted.

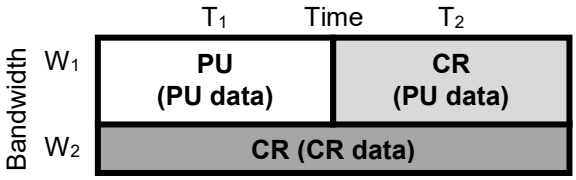


Fig. 3. A model of the time-bandwidth cooperative transmission idea [6].

3. Design and Implementation

Equipment used for this work included two Ettus Research USRP N210 SDR devices and one Nuand bladeRF SDR device, each of which was connected to a separate computer. Each device-computer pair acted as one node in the implemented network, which was modeled after a traditional relay network where one pair was the source, one pair was the relay, and one pair was the destination. This setup is shown in Fig. 4.

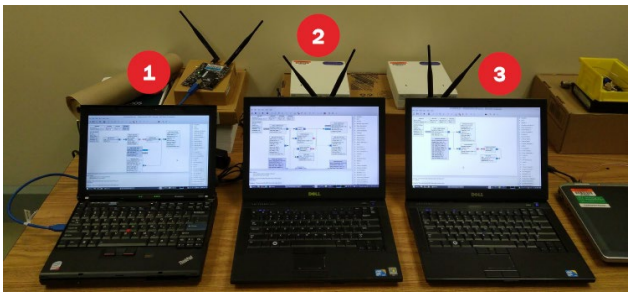


Fig. 4. Equipment setup including one Nuand bladeRF SDR device representing the source (1) and two Ettus Research USRP N210 SDR devices representing the relay (2) and the destination (3), respectively.

Using GNU Radio Companion, block diagrams for each of these nodes were developed to facilitate the encoding and transmitting of a file at the source, relaying of the file, and receiving and decoding of the file at the destination.

In the preliminary implementation of this experimental network, the relay node exhibited amplify-and-forward behavior, such that the signal was simply received and retransmitted, with no additional processing at this intermediate stage. The experiment used a text file to generate the sample signal, which was sent repeatedly to allow the continued generation of frequency versus power plots and verify continued transmission. Fig. 5, Fig. 6, and Fig. 7 show the frequency (x-axis) and power (y-axis) of the test signal at a single point in time.

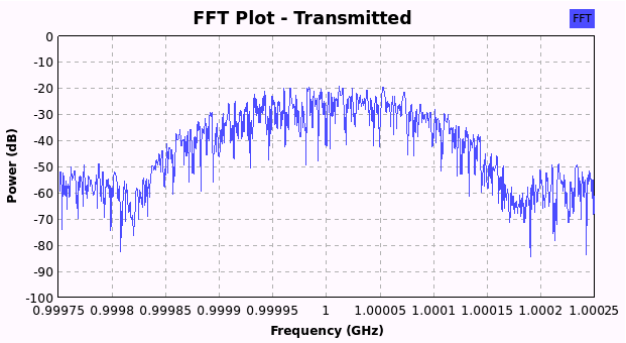


Fig. 5. Plot of the signal being transmitted by the source node.

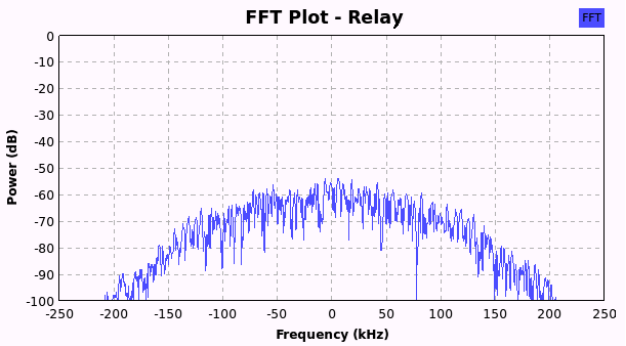


Fig. 6. Plot of the signal being retransmitted at the relay node.

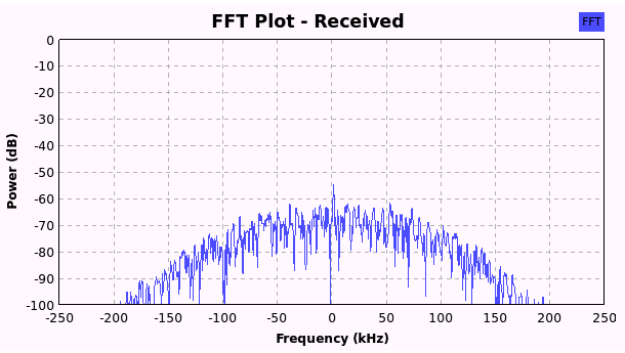


Fig. 7. Plot of the signal being received at the destination node.

The implemented network experienced a decrease in power as the signal passed from each node to the next, which was expected as a result of the wireless transmissions. Additionally, it should be noted that in traditional relay networks the relay node generally receives and retransmits the signal on the same frequency band. In this work, the relay node received and retransmitted the signal on different bands so that its effects could be isolated and its performance observed.

4. Conclusion

Based on this review of related literature and initial experimental implementation, it is possible to develop a clear plan of action for pursuing the development of a prototype network based on cognitive radio techniques. The foundation provided by the literature review was necessary in the implementation of a traditional relay network, which supports the further analysis and comparison of these two network types.

Inspired by the previous research covered by these related works, the purpose of this project will be to implement a cognitive radio network by exploiting variations of existing relaying schemes and compare its performance to that of a traditional network through simulations or by other means. The implemented simple relay network provides a baseline for measuring traditional network performance and offers additional opportunities for growth, including the implementation of alternate relaying schemes and the extension of the simple model to additional nodes. The results gathered from this experimental relay network implementation support the continued progress of this research.

5. Acknowledgements

This work was supported by a Student/Faculty Research Engagement (SFRE) Grant from Shippensburg University.

References:

[1] J. N. Laneman, D. N. C. Tse, and G. W. Wornell, Cooperative diversity in wireless networks: Efficient protocols and outage behavior, *IEEE Transactions on Information Theory*, *50*(12), 2004, 3062-3080.

[2] J. N. Laneman and G. W. Wornell, Distributed space-time-coded protocols for exploiting cooperative diversity in wireless networks, *IEEE Transactions on Information Theory*, *49*(10), 2003, 2415-2425.

[3] W. Su, S. Lee, D. A. Pados, and J. D. Matyjas, Optimal power assignment for minimizing the average total transmission power in hybrid-ARQ Rayleigh fading links, *IEEE Transactions on Communications*, *59*(7), 2011, 1867-1877.

[4] W. Su, J. D. Matyjas, and S. Batalama, Active cooperation between primary users and cognitive radio users in heterogeneous ad-hoc networks, *IEEE Transactions on Signal Processing*, *60*(4), 2012, 1796-1805.

[5] M. Yuksel and E. Erkip, Diversity in relaying protocols with amplify and forward, *GLOBECOM '03: Proc. of the 2003 Global Telecommunications Conference*, San Francisco, CA, 2003, 2025-2029.

[6] M. Yuksel and E. Erkip, Diversity-multiplexing tradeoff in cooperative wireless systems, *Proc. of the 40th Annual Conference on Information Sciences and Systems*, Princeton, NJ, 2006, 1062-1067.

DOCUMENT CLASSIFICATION PROBLEM: DOES A WEB PAGE CONTAIN AN ARTICLE?

Nicholas Rummel and Dr. C. Dudley Girard
Shippensburg University
nickarummel@gmail.com, cdgira@ship.edu

ABSTRACT
Web crawlers and content extraction tools often search for web page relevance. Identifying relevant web pages is a document classification problem. This research focuses on the document classification problem of determining whether web pages can be classified as news articles. Two types of attributes are used to identify web pages containing news articles: visual HTML tag features and link identification features. The first set of attributes refers to the visual structure of the web page. Commonly recognized visual features of an article include the title, author, and publication date. The second set of attributes, link identification features, is used to detect information in the URL of a web page. Using the ID3 algorithm, a decision tree is constructed to examine whether using both types of attributes improves the classification accuracy of pages from news websites.

KEY WORDS
News Articles Link Analysis Decision Tree

1. Introduction

In recent years, the format of how news is viewed on the internet has changed. Instead of traditionally viewing a news article, there has been a shift to present news through other media such as videos and photos. Additionally, news is shared on various social media platforms and people are looking at news from mobile devices rather than desktop or laptop computers. Because of these trends, news websites have altered their web pages for how news articles are displayed online. Web pages may now include a wide range of multimedia content. As a result, it is more difficult to classify web pages that are news articles.

Determining if a web page contains an article is a document classification problem. According to Park, Park, and Choi, "Document classification analyzes the content of a document and recognizes its classification level, given pre-defined classification specification such as a taxonomical hierarchy [1]." Specifically, documents can be classified into different categorical labels like content topic categories. For example, a web crawler may use document classification to categorize web page content for a search engine. It can also be used to classify a document as an article or not an article, which is valuable for article content extraction and summarization [1].

Research and development conducted on the two examples mentioned earlier often has experiments that avoid the document classification problem altogether. For instance, the experiment's dataset may only contain web pages or documents that are classified as a specific topic or as an article [1]. While avoiding the problem may help to simplify the experiment, it does not solve the document classification problem. As web pages evolve, this problem must be solved for implementing document classification into real-world applications.

This project will focus on a document classification problem for news articles. The following definition of a news article will be used from Pasternack and Roth:

> "An article is a contiguous, coherent work of prose on a single topic or multiple closely related topics that alone comprises the main information content of the page. In addition to the general requirements for an article, a news article must be a story or report at least two paragraphs and eight total sentences in length. The length requirement serves to exclude those pages that are merely brief summaries (typically with a link to the full article) [2]."

The problem description for this project is framed by the news article definition. Therefore, given a web page, does it contain a news article? This project aims to give insight to this question.

2. Literature Review

The related literature on this document classification problem propose a few topics of interest. The topics to be covered in this section will include uniform resource locator (URL)-based detection, the document object model (DOM), visual HTML tag feature detection, and the ID3 Algorithm.

2.1 URL-Based Detection

One strategy for this document classification problem is detecting whether a web page has an article can be determined by the page's URL. It is often faster to use

URL detection because "the length of a URL is a tiny fraction of the typical length of a web page [3]." A URL may contain words or phrases that will help with the classification of a web page. This is an assumption of URL-based detection, where "…URLs often contain meaningful words that are relevant to the page content [4]." Often, this approach is used by crawlers in search engines. "A crawler can potentially bypass examining the content of a page if a confident decision can be made based on the URL string [5]." To use URL-based detection, the features of the URL must be extracted by decomposing the URL into smaller pieces of information.

The most common way to break apart the URL is into keywords. According to Baykan, Henzinger, Marian, and Weber, there are "…four different methods (plus variants) to extract features from URLs: using tokens, n-grams derived from tokens, n-grams directly derived from the URL, and explicitly encoding positional information into tokens or n-grams [3]." The typical approach is splitting the URL into keyword tokens. The URL could be broken apart by considering the "/" character as the delimiter. However, another method is to use other characters as the delimiter. "Each URL is lower-cased and split into a sequence of strings of letters at any punctuation marks, numbers, or other nonletter characters [3]." Because relevant keywords are normally found after the domain name, there is an additional recommendation to only extract tokens that come after the web page's domain name [5]. After the URL is separated into tokens, the tokens can be used for determining relevant information about the web page [3].

Ferreira et. al. proposed a content extracting variant of link analysis called link target identification, which "benefits from several features extracted from the link structure in order to classify the link-target page as a 'news article' or an 'irrelevant page' (advertisement, non-text information, etc.) [6]." In addition to looking at the content of the web page, the proposed method identifies URLs to other web pages and examines the information embed into the link structure. The authors state that there are six features used to classify the link structure. The link could have a number, a date, a longer link length, slashes at the end, uses a reserved word, and the specific number of slashes. By implementing the link target identification with a machine learning algorithm, the authors were able to extract the content of the processed web pages for text summarization [6].

2.2 Document Object Model

When the document classification problem is discussed regarding web pages, many researchers will use the document object model (DOM). The DOM presents the HTML content of the web page as represented in a tree structure [7]. However, errors can occur with identification if the HTML structure is abnormal [8]. Before using a DOM, it is necessary to understand how the HTML is positioned into a DOM tree structure.

DOM trees follow a typical hierarchical tree structure, where "documents are character strings that adhere to the HTML syntax and can then be represented as DOM nodes [7]." These nodes are added into the tree, where a node is often represented by an attribute in the HTML code or computed by a web browser [7]. Figure 1 shows an example where a sample document is represented by a document object model tree [7]. Because of the tree structure used by DOMs, the HTML can be viewed and manipulated with programming languages.

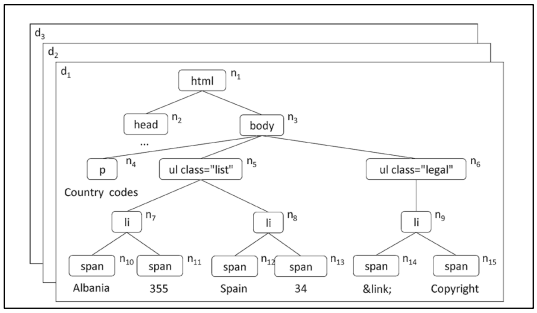


Figure 1: A sample DOM Tree [7].

Document object model trees are often created automatically by a web browser when rendering a web page. Client-based JavaScript can be used to access these trees from a web browser [4]. The JavaScript can also be used to eliminate performance issues, as it "can access the DOM tree of the HTML document to fetch the needed page parts (headings, meta keywords, etc.) with essentially zero overhead [4]." With the advantages of a web browser doing much of the work of translating the HTML into a DOM tree, it is beneficial to consider programming languages such as JavaScript that can access these trees [4].

2.3 Visual Features Detection

As a different approach for identifying web pages with articles, visual features can give insight to the type of content contained within a web page. These visual features are not related to images. Instead, visual features are created with HTML tags and CSS properties. Visual features are usually split into two categories: independent features and dependent features. Independent features for news articles are easily identified in web pages, consisting of font, size, position, and text features [9]. "Dependent features characterize the layout relationships among news attributes. We classify such dependent features into direction features and neighboring features [9]." Because visual features use different properties than the body content of the article, there are sets of criteria that must be matched for identifying these features [9].

Some of the common visual features found within a web page containing an article include: a category, source, publication date, title, author, content related links, and

comment links. For identifying relevant visual features, there are specific areas of a web page that must be searched. "For example, title almost always uses a notable font and is located in the upper part of a page [9]." For the title attribute, the font choice would be the independent feature and the location being near the top of the web page would be the dependent feature. The title may even be identified with a different tag (e.g. H1-H6) than the rest of the content [10]. The location of these visual features can be relevant for identifying where another feature may be in the page. "In other words, most layout relations do not appear in news pages at all. For example, content is never on top of title, and category and related news links are impossible neighbors [9]." Detecting the mentioned visual features and the visual features' locations provides criteria for classifying a web page as an article [9, 10].

Another type of visual feature used to recognize articles is a line-break. A Web article that has visual structure is unstructured in the HTML/DOM [11]. "For example, the article body usually consists of contiguous paragraph blocks occupying the main area of the Web page [11]." However, these contiguous paragraph blocks visually are broken up in the DOM, usually with *br* or *hr* HTML tags [11]. Therefore, a line-break occurs in a DOM if the property is considered a block or is either the *br* or *hr* HTML tags. Line-breaks, consequently, are distinguishable as a visual feature because it separates paragraphs or separates paragraphs from other visual features [11].

The HTML tag and CSS properties are used to recognize visual feature attributes unique to a news article. Liu, Tan, and Xiao propose a series of extraction rules for each news attribute [9]. Each set of rules are translated into an algorithm for identifying that respective attribute in a news article. Figure 2 shows the title attribute ruleset as an algorithm [9]. The input for this algorithm is a set of text blocks. The output is a candidate set, which contains any remaining text blocks that satisfied all the title extraction rules. It is entirely possible that the output will not contain any candidates.

To start, the set of text blocks inputted are saved as the candidate set. Starting with the first candidate in the candidate set, the first rule will check if the font size of the text block is between 15 pixels and 45 pixels. If the font size is outside of this range, this text block is removed from the candidate set. Otherwise, the text block moves on to the next rule. This rule checks to see if the text has a black or blue font color. If it is not either color, the text block is removed from the candidate set. The next rule checks to see if the text block's position is near the top of the page. If it is not near the top of the page, the block is removed from the candidate set. The fourth rule checks that the position of the text block is within the height of the screen, verifying that the title can be seen without scrolling. If the

```
Input: textBlockSet;
Output: titleCandidateSet;
1   titleCandidateSet= textBlockSet;
2   For each candidate c in titleCandidateSet do
3   // filter with rule 1
4       If 15px<=c.FontSize<= 45px
5           Remove c from titleCandidateSet;
6   // filter with rule 2
7       If c.FontColor is not black or blue
8           Remove c from titleCandidateSet;
9   // filter with rule 3
10      If c.y>page. height/2  // title must be in the
    upper part of the page
11          Remove c from titleCandidateSet;
12  // filter with rule 4
13      If c.y>screen.Height // title must be seen without
    pagedown
14      Remove c from titleCandidateSet;
15  // filter with rule 5
16   If c.TextLength <8 or>50
17      Remove c from titleCandidateSet;
18  // filter with rule 6
19   If c.Hyperlink=true
20      Remove c from titleCandidateSet;
21  Return titleCandidateSet;
```

Figure 2: An algorithm to find the title [9].

block fails this rule, then it is removed from the candidate set. The fifth rule determines if the block of text has a length between 8 and 50 characters. If the length does not fall in this range, the block is removed from the candidate set. Finally, the last rule checks if the block of text is a link. If the text is a link, then it will be removed from the

2.4 ID3 Algorithm

To design a solution for this document classification problem, a machine learning technique can be used for training a system from datasets. An ancestor to John Quinlan's C4.5 Algorithm, the Iterative Dichotomiser 3 (ID3) Algorithm is used to create a decision tree from given attributes for a dataset [12]. The ID3 algorithm is a greedy algorithm. "It has generally been found to construct simple decision trees, but the approach it uses cannot guarantee that better trees have not been overlooked [12]." Additionally, the algorithm utilizes supervised learning, where labeled training data is used as training examples to produce a function or method to label new examples.

The ID3 algorithm can generate a decision tree by using two measurements: information entropy and information gain. Information entropy measures the amount of information in an attribute. Similarly, information gain is a statistical property that measures how well a given attribute separates training examples into targeted classes. The decision tree is generated by maximizing the information gain of each attribute [12]. The following two equations are used for calculating information entropy and information gain:

$$Entropy(D) = -\sum_{d=1}^{|k|} p_d \log_2 p_d$$

$$Gain(D, A) = Ent(D) - \sum_{v=1}^{V} \frac{|D_v|}{|D|} * Ent(D_v)$$

where D is the sample set, $|k|$ is the total number of classes for an attribute, p_d is the proportion of the set belonging to a class of an attribute, V represents all possible values of an attribute A, D_v is the subset of the training sample for which the attribute A has a value V, $|D_v|$ is the total number of elements in subset D_v, and $|D|$ is the total number of elements in the sample set [13]. These equations are used together to generate a feasible decision tree.

The ID3 Algorithm is recursive, where it selects the next node using information gain but using a subset of the remaining data. Figure 3 shows John Quinlan's ID3 Algorithm [13].

Input: Training set D = $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
Attribute set A = $\{a_1, a_2, \dots, a_d\}$
Output: a decision tree
Generate node (D, A)
If samples in D belong to the same class
  Then: The node is labeled as class C leaf node; **Ret**
If A = ϕ or the values in A are same in D
  Then: The node is labeled as leaf node; **Ret**
Maximum Information Entropy is chosen as a heuristic strategy to select the optimal splitting attribute a_j from A.
For every value a_i^v in a_j do
  Generate a branch for node
  D_v is the sample subset that has value a_i^v from a_i in D
  If D_v is empty
    Then: The branch node is labeled as a leaf node;
      Ret
  Else
    Take Tree Generate (Dv, A\{a_i}) as the node that can continue to be divided

Figure 3: The ID3 Algorithm [13].

This algorithm takes in the training set and attribute set as inputs. The decision tree is the output. The leaf node is chosen if the training samples belong to the same class, if the attribute is null, or if all the values for a given attribute are the same in the training sample. When one of these conditions are true, then the node is selected as the leaf node. After a leaf node is selected, the branches from the node are selected from the values of that attribute. However, if a subset of the data for a given value is empty, then the branch becomes the leaf node. Finally, the algorithm moves on to the next iteration of selecting a leaf node using the new data subsets split on the attribute with maximum gain. The time complexity of this algorithm for n attributes with m instances in the training set is $O(n * m \log m)$. By using this algorithm, the machine would learn from the dataset and identify web pages with news articles from the generated decision tree.

3. Primary Objective

The primary objective of this project is to assess the predictability of a news article existing in a web page using ID3 with the attributes outlined in "Extracting multiple news attributes based on visual features" [9] with and without URL identification attributes outlined in "Applying Link Target Identification and Content Extraction to improve Web News Summarization" [6].

4. Solution Description

We propose to use URL-based detection to initially process each web page in a dataset to determine if the link contains relevant article information about the web page. Then, a content-based approach will identify article attributes as visual features. These features are then analyzed from a document object tree, which is storing the relevant data from the HTML file. The ID3 Algorithm would provide the statistical learning and training using decision trees to classifying web pages with news articles.

To achieve the primary objective, there are two tools that will be used for the project. First, the programming language Java will be used with the Eclipse Integrated Development Environment (IDE) for the code development. Additionally, the free program HTTrack is used for obtaining the dataset for this project. The dataset will consist of 330 web pages obtained from news websites using HTTrack. Because some web pages have an article while some do not, each HTML file is manually labeled as an article or not an article. After obtaining the dataset, the content-based detection of identifying visual features as attributes, implemented in Java, is applied. JSoup is then used to generate and interact with the DOM tree. Next, the link target identification for URL attributes, implemented in Java, is applied. Finally, the ID3 Algorithm, implemented in Java, is trained and tested with the obtained dataset.

Because the ID3 Algorithm builds the decision tree using attributes, there are sets of visual feature and URL attributes that are used. For this solution, a modified version of the ID3 algorithm was used. In this modified version, the algorithm removes an attribute from being used again once used as a leaf node in the tree. Additionally, the algorithm will continue to use all attributes even if the information gain is found to be the same. This will yield a larger decision tree overall.

The attributes for the ID3 algorithm that have been selected for this project are outlined in "Extracting multiple news attributes based on visual features" [9] and "Applying Link Target Identification and Content Extraction to improve Web News Summarization" [6]. The visual feature attributes will be the title, publication date, author, comment link, source, content, category, and related news link. The expected input for identifying the visual feature attributes will be the web page's HTML file. The URL attributes will be a number, a date, a longer link length, a specific number of slashes, the use of a reserve word, and slashes at the end of the link. The input for the link target identification is the web page's URL. The expected output

is the count of web pages that were classified correctly and incorrectly.

5. Hypothesis

The hypothesis for the project is that adding URL-based attributes to the ID3 decision tree using content-based attributes increases the accuracy of identifying web pages with news articles.

6. Experiment Design

The dataset contains a total of 330 web pages. By using the jackknife method, 300 of the web pages are randomly chosen as the training set for each trial, while the remaining 30 are used for the test set. 33 web pages (16 with an article and 17 without an article) are downloaded from each of the following 10 news websites: CNN, New York Times, Washington Post, Yahoo News, Fox News, BBC, The Verge, Huffington Post, ABC News, and CBS News. Each web page was manually labeled. Table 1 shows the experiment factors and Table 2 shows the experiment block design.

Factors	Values
URL Identification	{false, true}
Sample	Each of the 20 trials will randomly select the 300 web pages for the training set and the 30 web pages for the test set.

Table 1: Experiment Factors

	Visual Features + Link Analysis		
Visual Features Only	**Correct**	**Incorrect**	**Row Totals**
Correct	295	11	306
Incorrect	16	278	294
Column Totals	311	289	600

Table 2: Block Design

7. Results

Figure 4 is a box and whisker plot showing the average accuracy for the experiment's 20 trials. The plotted data on the left shows the average accuracy for visual feature detection without link analysis. The plotted data on the right shows the average accuracy for visual feature detection with link analysis. The y-axis shows the percent accuracy. The line in each box is the median accuracy of all the trials. The box represents the interquartile range, or middle 50% of the data. There are not 20 data points on each set of plotted data because not all values were unique.

Figure 4: Average Accuracy of Experiment

The experiment yielded an average accuracy of 51.00% with a standard deviation of 6.93% for visual feature detection only. When adding link analysis to the visual feature detection, the average accuracy was 51.83% with a standard deviation of 6.97%. Because of sample overlap in the trials, the statistical test used to analyze the experiment's results is McNemar's test. A variant of the Chi-squared statistical test, McNemar's test compares two proportions of dependent samples.

Table 3 shows the contingency table demonstrating the proportions for web pages correctly and incorrectly classified whether it is an article. The null and alternative hypotheses for the statistical test are below.
H_0: $p_1 = p_2$
H_a: $p_1 < p_2$

	Content-based Detection without URL Identification	Content-based Detection with URL Identification
300 training/ 30 test pages	X	X

Table 3: Contingency Table of Results

The null hypothesis is that there is no difference in accuracy between visual feature detection only (p_1) and visual feature detection with link analysis (p_2). The alternative hypothesis is that the visual feature detection with link analysis (p_2) is more accurate than the visual feature detection only (p_1).

For a contingency table of this size, the degree of freedom is 1. For the calculations, the sample size is n = 600. The results are tested with 95% confidence ($\alpha = 0.05$). The calculated McNemar's Chi-squared statistic is 0.5925. This value is used to calculate a p-value of 0.4414. Because the p-value is greater than alpha, the data fails to reject the null hypothesis. Therefore, the accuracy values are not significantly different.

8. Discussion and Conclusion

The data analyzed from the experiment shows that link analysis did not statistically improve the accuracy of detecting web pages containing articles. The average accuracy for both cases were just above 50%, suggesting that there are several issues with the experiment. One of the first areas to investigate is what kind of errors the code is exhibiting. For this, an error matrix is considered. Table 4 is the layout of the error matrix. Table 5 is the data for the error matrix. Due to time constraints, only the first trial in the experiment is considered.

For the first trial, both visual feature detection with and without link analysis had identified the same 17 out of 30 web pages correctly, an accuracy of 56.67%. Upon further investigation, the 17 web pages that were correct identified were not articles. These were considered true negatives, meaning that the web pages were rejected correctly. The remaining 13 web pages were labeled as articles but were misidentified as not an article. These remaining web pages represent a false negative, meaning that it is a miss and resulting in a type II error. In this trial, no data points were a true positive or a false positive. There are several issues to consider that could cause the experiment's code to incorrectly predict web pages with articles.

	Is Article	**Is Not Article**
Article Found	True Positive	False Positive
Article Not Found	False Negative	True Negative

Table 4: Error Matrix for Experiment

	Is Article	**Is Not Article**
Article Found	0	0
Article Not Found	13	17

Table 5: Error Matrix with Data (Trial 1)

The first issue to consider is the visual feature and link analysis detection rules. Technology changes quickly over time. The HTML tags and CSS styling used in the visual feature rules may be outdated. Additionally, the structure for links may have changed. For example, websites that dynamically create web pages from content stored in databases may not have the same structure as another website keeping individual files for each web page.

Another issue to consider is related to the web pages used in the dataset. In recent years, the source code of web pages contains large amounts of scripts and other code. HTTrack (www.httrack.com) acts as a web browser when retrieving web pages. However, it is possible that client and server code was not executed properly during the dataset retrieval process with HTTrack. Without this code fully executing, the pages would not have finished rendering all elements and the web page would be incomplete.

The last issue to consider is related to the ID3 algorithm. After running a small test by training and testing on the web pages from the same source, web pages from The Verge had an accuracy rate of 6.06% while web pages from CNN has an accuracy rate of 96.96%. From sifting through the generated decision trees for each source, the modified ID3 algorithm is to blame. By forcing all attributes to be used in a generated decision tree, the statistical property of information gain is disrupted. The best local choice will continue to be made but the algorithm does not stop if the information gain of all attributes is the same. As a result, this creates a larger decision tree, increasing the difficulty of deciding on an accurate prediction.

Additionally, link analysis features were used more often in branches than the visual features. From this small test, generated decision tree for The Verge did not have any visual feature attributes. The visual features were not used so the branches of the tree were based strictly on link analysis rules. This suggests that the rules used from the sources in the primary objective could not locate visual features inside the web pages.

To correct these issues in future research, the rules for the visual features and link analysis need to be verified and updated to match new technology standards. Using a different dataset retrieval tool that fully renders the web pages would eliminate errors related to client and server code not running correctly. A new experiment could be used to determine whether link analysis is better than visual feature detection. Additionally, using a different machine learning technique like support vector machines and neural networks could help with issues resulting from the ID3 algorithm. Finally, the training and testing should be used with a 10-way validation approach, allowing for every combination of the training and validation groups to be involved with training.

To conclude, the URL identification did not increase the prediction accuracy of web pages with news articles with content-based detection in this experiment. Upon further investigation, several issues have been identified with potential solutions. Additionally, new research for this document classification problem could be developed in future work.

9. Acknowledgements

I would like to thank Dr. Girard for providing the idea and mentoring me throughout all phases of the research project. Additionally, I would like to thank Dr. Armstrong and Dr. Wellington for challenging my understanding of the topic throughout the research and development process.

References:

[1] J. Park, J. Park, and J. Choi, web-based document classification using a trie-based index structure, *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, Washington, DC, 2007, 52–55.

[2] J. Pasternack & D. Roth, Extracting article text from the web with maximum subsequence segmentation, *Proceedings of the 18th International Conference on World Wide Web*, New York, NY, 2009, 971–980.

[3] E. Baykan, M. Henzinger, L. Marian, & I. Weber, A comprehensive study of features and algorithms for URL-based topic classification, *ACM Trans Web*, *5*(3), 2011, 15:1–15:29.

[4] A. Anagnostopoulos, A. Z. Broder, E. Gabrilovich, V. Josifovski, & L. Riedel, Web page summarization for just-in-time contextual advertising, *ACM Trans Intell Syst Technol*, *3*(1), 2011, 14:1–14:32.

[5] S. D. Gollapalli, C. Caragea, P. Mitra, & C. L. Giles, Improving researcher homepage classification with unlabeled data, *ACM Trans Web*, *9*(4), 2015, 17:1–17:32.

[6] R. Ferreira, R. Ferreira, R. D. Lins, H. Oliveira, M. Riss, & S. J. Simske, Applying link target identification and content extraction to improve web news summarization, *Proceedings of the 2016 ACM Symposium on Document Engineering*, New York, NY, 2016, 197–200.

[7] P. Jiménez & R. Corchuelo, Roller: A novel approach to web information extraction, *Knowl. Inf. Syst.*, *49*(1), 2016, 197–241.

[8] J. Prasad & A. Paepcke, Coreex: Content extraction from online news articles, *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, New York, NY, 2008, 1391–1392.

[9] W. Liu, H. Yan, & J. Xiao, Extracting multiple news attributes based on visual features, *J. Intell. Inf. Syst.*, *38*(2), 2012, 465–486.

[10] J. Fan, P. Luo, S. H. Lim, S. Liu, P. Joshi, & J. Liu, Article Clipper: A system for web article extraction, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, 2011, 743–746.

[11] P. Luo, J. Fan, S. Liu, F. Lin, Y. Xiong, and J. Liu, Web article extraction for web printing: A DOM+Visual based approach, *Proceedings of the 9th ACM Symposium on Document Engineering*, New York, NY, 2009, 66–69.

[12] J. R. Quinlan, Induction of decision trees, *Mach Learn*, *1*(1), 1986, 81–106.

[13] Y. Wang, Y. Li, Y. Song, X. Rong, & S. Zhang, Improvement of ID3 algorithm based on simplified information entropy and coordination degree," *Algorithms*, *10*(4), 2017, 1–18.

ARE UNIT TESTS GOOD ENOUGH FOR DESIGN PATTERNS? AN EMPIRICAL STUDY

Austin Smale[1] Courtney Rush[2] Chen Huo[3]

Software Engineering
Shippensburg University of Pennsylvania
{as3871[1], cr5603[2], chuo[3]}@cs.ship.edu

ABSTRACT

Design patterns, such as the observer pattern, capture the knowledge from previous software construction and reify the reusability of good designs. Unlike data structures, design patterns are usually not reusable code but high level design concepts due to their nature. When programmers attempt to write code that follows the design, the traditional tests can only tell whether the outcome of the program is correct since in general tests cannot examine the interactions in an execution. However, if the unit tests are written with the awareness of the underlying design patterns, most mistakes in design could be detected by tests. In this paper, we test the hypothesis that well written tests can entail the correctness of design pattern implementations. We developed an effective behavioral checking framework which verifies the expected interactions during an execution. We used a design-patterns course project as the subject of our empirical study. All the 42 student submissions involving 4 design patterns, which passed well-written tests by the instructor, also passed our behavioral checking framework. In this case knowledgeable unit test writers can capture the validity of the patterns. We conclude that good tests entail correct use of design patterns in common cases.

KEY WORDS

Software Testing, Design Patterns, Software Engineering

1. Introduction

Design patterns, in a narrow sense, refers to the designs illustrated in the *gang of four* book[1] by Gamma, Helm & Johnson in 1994. The book shows a catalog of 23 patterns in the `C++` programming language (`C++`). The included design patterns had been brewing for several years before the release of the book[2]. Over the years, these design patterns mentioned in the book are widely spread among different object-oriented programming languages in both fields of software development and programming education. As the Java programming language became more and more popular in the new millennium, Gamma, Helm & Johnson's design patterns have been adapted by the `Java` community. For simplicity, the word design patterns in this paper refers to the 23 design patterns from Gamma, Helm & Johnson's book[1] unless otherwise specified. New books introducing the same set or a subset of these design patterns in `Java` started to emerge, e.g., *Head First Design Patterns*[3], *Design Patterns for Dummies*[4]. There are also numerous Java tutorials on design patterns on the internet.

Though popular, the design patterns originated from Gamma, Helm & Johnson started to receive criticism ever since its birth[5]. It is the *code reusability* often questioned for such a catalog view of software constructs, e.g., [5, 6, 7]. A *catalog* design pattern usually only has informal descriptions in English (or other nature languages) and occasionally diagrams (i.e., `UML` class diagrams)[5] that can be applied to different domains but not the implementation. For example, if a programmer writes code reflecting the *strategy pattern* in one project, source code in the first project is generally not reusable in another project. Such criticism usually comes from another family of design patterns with the framework-component view[8, 5]. For a *framework-component* design pattern, the concrete framework can be reused and the programmers just need to provide the components. Such vulnerability makes it difficult for programmers to write programs that follow the `catalog-style` designs. More discussions about the arguments on *code reusability* will be provided in Section 2.1.

Despite of being accused of lacking code reusability, primarily by the framework-component view, the family of catalog-style design patterns is proven to be helpful in industry. It also turns out to play an important role in teaching beginners the object-oriented programming paradigms because of the alleged low code reusability. Such design patterns demonstrate the proper use and the power of object-oriented programming paradigms, such as proper uses of the *inheritance* and the *composition* relation, method *overriding* and *overloading*, etc. Since the programmers cannot reuse the implementation, they will

have to repeat coding for implementations from case to case. In contrast, the users of the framework-component view designs only need to provide components to obtain complete programs and meanwhile the framework part is too technical for beginners to grasp for educational occasions due to the nature of high code reusability.

For this reason, books like *Head First Design Patterns*[3] are suitable as Java textbooks for the second or third programming course for undergraduate students. However, as mentioned above, since code written according to a catalog design pattern is in general not reusable, it is impossible to document every possible usage of such design patterns. Design patterns are a key component of the effective learning of object-oriented programming paradigms. It is imperative for a programmer or a student to know whether their implementations follow certain design patterns.

In the process of software development, software testing is commonly used to ensure the quality of the software. However, tests can only reveal observable erroneous states of the program[9]. In general, the design of the system under test is transparent to tests — no internal variables or functions or their interactions are visible to tests. As long as an implementation can provide expected functionality, it can pass a proper set of tests. For this matter, manual work is considered necessary for checking the actual design of the code. In this paper, we demonstrate that, although tests cannot check all kinds of interactions in a program, passing *well-written* tests entails the correct use of design patterns in the code. Section 2.3 explains what are considered well-written tests.

Our work has the following contributions: (1) A behavioral checking tool based on Mockito that can identify incorrectly implemented design patterns (but can produce the right output), (2) A collection of design pattern variants that categorized and used to show the effectiveness of the behavioral checking tool, and (3) An empirical study on 42 student submissions for 4 design patterns that shows good tests entails the correct implementation of design patterns.

In the rest of the paper, Section 2 shows related work of our research. Section 4 describes the details of the behavioral checking tool and our experiment design. Section 5 presents the subject and the result of the empirical study. The section also discusses possible explanations to the seemly contradictory conclusion. Threats to validity is discussed in Section 6. Finally, Section 7 wraps up the paper with our conclusion.

2. Related Work

2.1 Design Patterns

Gamma, Helm & Johnson[1] describes a catalog of 23 design patterns which can be divided into three categories based on the intent: creational, structural, and behavioral. The authors summarized design patterns as:

> Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to this problem in a way that you can use this solution a million times over, without ever doing it the same way twice.

Note the word *describe*. *The core of the solution*, a design pattern, can be applied to two similar problems. However, the resulting source code usually can't be used interchangeably. This limit on reusability also makes the identification of design patterns difficult.

Pree published *Design Patterns for Object-oriented Software Development* the same year. In the book, Pree presented several frameworks as the design patterns. The frameworks are not only high level descriptions of solutions but also source code which the users can just provide components to complete the software. Pree & Sikora[5] later presented a tutorial in which he stated the advantages of his framework perspective over Gamma, Helm & Johnson's catalog perspective.

2.2 Software Testing

Comparing to formal methods in software engineering, such as model checking[10], software testing is an *optimistic* method[11]. Formally speaking, testing is not *sound* which means a program that fails the tests must be bad but a program that passes the tests may also be bad. Despite of the limit, testing brings the balance of effectiveness and efficiency to the industry and is widely accepted as a crucial process in software development.

Tests are conceptually simple—they are composed of test inputs and test oracles. Test inputs are used to execute the program under test. Test oracles are used to verify the execution induced by the inputs produces the expected results, i.e., `assertEquals` statements in `JUnit` tests. The test oracles are hard to write and they can only access the final and observable states of the program[9]. In general, it is impossible and also unnecessary for tests to check how the program reach the final states. To know *how*, one must examine the interactions that happened during the program execution. In the following subsection, we will examine the possibilities for detecting such interactions inside the program, or the *behavior*.

2.3 Test Criteria

What makes a good test? How do you know your tests are effective? The testing community developed a variety of test criteria for measurements over the years. These criteria work at different levels, such as the Federal Aviation Administration (FAA) requires the Modified Condition/Decision Coverage (MCDC) criterion for life-critical part of the program which is quite strict[9]. In most cases for commercial applications, structural coverage and input domain coverage is sufficient. In

our paper, well-written tests will have good structural and input domain coverage. In our case, good structural coverage means every reachable statement in every publicly accessible method will be executed during testing. Good input domain coverage means all kinds of inputs with respect to the specification of the programs will be used in the tests. In Section 3, it will be expanded with a concrete example. [Austin: explain why our tests are good in the motivating example section.]

2.4 Examining the Behavior

There are generally two ways to analyze a program, static analysis and dynamic analysis. Typical static analysis includes pointer analysis at various level, liveness analysis, etc. Typical dynamic analysis includes tracing, profiling, etc. There are tools that you can use out of the box and tools that allow you to build your own from scratch based off of them. For the best usability in this study, we chose to use `Mockito` framework[12]. `Mockito` is an open source Java mocking framework which can verify if certain methods are invoked and even for how many times during the program execution.

3. Motivating Example

In this section, we will briefly go over one variant that was created when building our tool to catch the issue. When creating this variant we had to mimic what could be a possible solution one student could come up with. In the case below, we analyze the performance of our tool on a variant of the observer pattern that instead of using a list to maintain its observers, the student used an instance variable for each observer.

3.1 The Observer Pattern

The observer pattern is a very useful pattern that allows a subject to maintain a list of observers. If the state of changes that requires an observing class to know, the subject will automatically notify all its observers of the state change. As seen in Figure 1, the object that is a subject must maintain a list of observers. This allows for a new observer to be added at any point and be notified when the state changes of something that it needs to know about. Without the use of the list, the subject is only limited to the amount of instance variables the subject has. Also, if there was no list and only instance variables, the user would have to keep track of which instance variable is being used and which is empty. This can ultimately slow down the system greatly with a large amount of instance variables in the subject.

3.2 Variant

Our variant, shown in Figure 2, removes the list of observers from the `SimpleTimer` class and has nine instance variables to hold an observer in. This variant breaks the pattern simply by not allowing more than nine subjects to view a state change in the `SimpleTimer`, a round update in this case. If ten observers were to be

```
public class SimpleTimer extends Thread
 ↪  implements Timer {

  private int round; // the current round
    private List<TimerObserver> observers;
     ↪  // the list of observers
```

Figure 1: The correct implementation of a strategy pattern.

observing the `SimpleTimer`, only nine of those ten would be notified a round has changed, and the tenth would be stuck on the same round forever. This could be a very common mistake for someone new learning the pattern, they know that the subject, `SimpleTimer`, must notify an observer, `TimeObserver`, of when a round changes. Creating an instance variable for each observer also creates very messy code when adding, removing, or even getting the number of observers a subject maintains.

3.3 Testing

JUnit was used to test the functionality of `SimpleTimer`. The design of each JUnit test for `SimpleTimer` was simply checking that it could update the timer, add, remove, and simply get the size of the list that should be there. However, the test does not know the number of observers that the actual application could have, and it just simply verifies the functionality of the `SimpleTimer` with at most 3 observers. This is only one third the number of observers that the `SimpleTimer` can manage. Figure 3 shows an example test that proves adding an observer works, as well as updating the round and checking to see that `TimeObserver` was also updated. With the black box nature of JUnit, the tests already created will also pass the variant created. This created the need for our tool built in Mockito to fill in the gaps that were created by the unit tests.

3.4 Checking

Mockito is a very useful framework used for testing pieces of code. We leveraged Mockito and combined it with our test to verify the inner workings of the `SimpleTimer`. We wanted to verify that the `SimpleTimer` was maintaining a list of observers. Shown in Figure 4, you can see that we first had to spy an `ArrayList<TimeObserver>` to add into our `SimpleTimer`. Since there is no way to set the list in `SimpleTimer`, we had to use reflection to replace the current observer object with our `ArrayList<TimeObserver>`. We then added a `MockSimpleTimeObserver` to the `SimpleTimer`. Then we used Mockito's verify functionality to see if our spied `ArrayList<TimeObserver>` called the `add()` method inside of the `ArrayList` functionality.

4. Methodology

This section will go over the way we leveraged Mockito to create our tool. It will explain our process of creating mutants, also known as variants, and then using our tool

```
public class SimpleTimer extends Thread
↪  implements Timer {

  private int round; // the current round

    // collection of observers
    private TimerObserver ob1;
    private TimerObserver ob2;
    private TimerObserver ob3;
    private TimerObserver ob4;
    private TimerObserver ob5;
    private TimerObserver ob6;
    private TimerObserver ob7;
    private TimerObserver ob8;
    private TimerObserver ob9;
}
```

Figure 2: A wrong implementation of the singleton pattern.

```
@Test
public void testAddOneAndUpdate()
{
  MockSimpleTimerObserver mock1 = new
  ↪  MockSimpleTimerObserver();
  SimpleTimer s =
  ↪  Mockito.spy(SimpleTimer.class); //
  ↪  mock up a simpletimer
  s.addTimeObserver(mock1);
  assertEquals(s.getNumObservers(), 1);

  // update the time
  s.timeChanged();
  assertEquals(s.getRound(), 1);
  Mockito.verify(s,
  ↪  Mockito.atLeast(1)).timeChanged();

  // check the mock1 is also on round 1
  // we don't need to add a getter because
  ↪  these classes are in the same
  // file.
  assertEquals(mock1.myTime, 1);

  // update time again
  s.timeChanged();
  assertEquals(mock1.myTime, 2);
}


// Other Tests
...
```

Figure 3: Some JUnit tests of the singleton pattern.

```
@Test
public void testToMakeSureTimerUsesList()
↪  throws Exception
{
  // fields
  ArrayList<TimerObserver> observers = new
  ↪  ArrayList<TimerObserver>();
  ArrayList<TimerObserver> mockedList =
  ↪  Mockito.spy(observers);
  MockSimpleTimerObserver mock1 = new
  ↪  MockSimpleTimerObserver();

  // set the private field to our mocked
  ↪  ArrayList for mockito testing
  SimpleTimer simple = new SimpleTimer();
  Field f = SimpleTimer.class

              ↪  .getDeclaredField("observers");
  f.setAccessible(true);
  f.set(simple, mockedList);

  simple.addTimeObserver(mock1);

  // verify it was added, should fail unless
  ↪  SimpleTimer has an ArrayList
  Mockito.verify(mockedList,
  ↪  Mockito.times(1))
    .add(Mockito.any(TimerObserver.class));
}
```

Figure 4: Our mockito test implementation of the singleton pattern.

in order to kill the mutant. This section will additionally explain what a good unit test consists of.

4.1 The Mockito Framework

Mockito is a mocking framework that offers greater functionality in constructing tests that focus on Behavior Driven Development (BDD). We chose to use this tool as it offered the ability to inspect the internal interaction when the code is executed. With this ability, we can construct a tool that can identify an improperly implemented pattern even the implementation can pass well-written test.

We developed a behavioral checker using the Mockito framework, referred as the *Mockito tool* in later sections, that can verify the internal interactions of programs with our predefined rules that capture the essence of design patterns. An example of such rules can be found in Section 3. Section 4.4 briefly lists the variants we summarized for each design pattern. The complete set of rules is verbose and can be found in the repository of the variants and the Mockito tool.

To show that our tool is effective, we created one to three problematic implementations, called *variants*, for each design pattern. Each variant can produce the expected outputs for the regular unit tests while using a slightly different (incorrect) implementation for the design pattern.

The following sections are organized as following: (1) Section 4.2 discusses the relation between our idea of variants of design patterns and a commonly used technique in software testing—*mutation analysis*., (2) Section 4.4 briefly describes the variants we created for each design pattern., and (3) Section 4.5 describes how the effectiveness is evaluated and the results.

4.2 Variants and Mutation Analysis

A similar idea can be found in an area of software testing, called *mutation analysis*. Briefly speaking, mutation analysis is a way for measuring the effectiveness of a test suite. The tester can plant a *static* fault in the application, i.e., changing `if (a > b)` to `if (a >= b)`, and then run the tests to see if one or more tests fail. Such a static fault planted intentionally is called a *mutant*. When the mutant cause failure of the tests, we say the mutant is killed. Intuitively, killing mutants is a positive sign for the quality of the tests. (If the mutant mentioned above is killed, it is very likely that the tests are written carefully with the consideration of the boundary conditions.) Researchers now have confidence that artificial faults are good representatives of real-world software faults [9, 13, 14].

Our design pattern *variants* are analogous to the mutants in the context of mutation analysis. Our tool is analogous to the tests. If our tool can kill such variants, it would give us more confidence about the effectiveness of our tool. The variants are designed based the domain experience of the author and also a faculty member who teaches the design patterns course. All the variants produce the same results with the correct designs and the changes made are believed to be the commons mistakes when implementing the design patterns. Table 1 provides a summary of the design patterns and their variants.

4.3 Good Unit Testing

The unit tests provided in this course are considered well-written because they examine the APIs and inputs thoroughly. Details can be found in Section 2.3. The testing was all automated upon submission and feedback was given immediately of what failed. The students were then able to make the changes necessary to make the tests pass. In our unit tests, each input domain was thoroughly tested. If a class had overloaded its constructors, there was at least one test for each constructor to verify that the object would be structurally correct in each case. Also, there was a case that dealt with data that fit inside the range of what was needed. As well, as border cases to further verify that the structure of the code was written correctly, and tests that fell outside on each side of the border cases. For example, when testing the `Alien` class, the `Alien` has three different constructors. Throughout that test, each constructor was used at least once to show the `Alien` would still behave the same as an `Alien` that created by another constructor. Also the `Alien` had some methods to go along with it, `getRecoveryRate()`, `getMaxLifePoints()`, `recover()`, and `updateTime()`, and others inherited from `Lifeform`. Each method from `Alien` and `Lifeform` was used at least once to show that the functionality should work in the submission. Because of this in-depth testing, each possible node inside the `Alien` class was touched by at least one test. If something was not touched, it gave the feedback to the students that there was a wrong or missing part in their submission and where it occurred. This allowed them to fix it and properly build the patterns into the game.

4.4 Design Patterns and Their Variants

We first evaluate the effectiveness of our tool using variants of the design patterns. An implementation of a design pattern is said to be *correct* when it follows the design exactly and always produces the expected results. A variant of a design pattern also produces the expected results, however, it does not follow the expected design. In the following sections, we will first discuss the mutation analysis methodology and then list each of the six design patterns and its variant. The introductions to the design patterns are intended to be brief in this paper. More details can be found in materials mentioned in Section 1.

4.4.1 Created Variants

A possible variant of the strategy pattern is identified when the `Subject` does not use the interface that holds the desisired behavior. Instead, the behavior is copied and pasted into each `Subject`. This is incorrect because purpose of this design pattern is to organize code into

recognizable patterns that are easier to identify, which then increases readability. Without the use of the interface, the code becomes less readable and more prone to errors when editing it at a later time. Another possible variant of the strategy pattern deals with nonsensical variable names. This variant fails our white-box testing because it also decreases readability of the code.

A possible variant of the observer pattern is identified when the implementation omits the use of the `Subject` and `Observer` interfaces. Instead, like the strategy pattern above, the desired behavior is copied and pasted to each `Subject` and `Observer` that needs it. This fails our white-box testing because, like the strategy pattern, defeats the purpose of using a design pattern due to its decreased organization and readability. A nonsensical observer list name is also tested here, and as stated with the strategy pattern, it is an incorrect variant of the observer pattern. Another possible variant of the observer pattern includes the replacement of `ArrayList<Observer> list` with a series of `Observer ob1;` objects. This could cause many errors in future editing and usage of the `Subject`, such as the accidental overwriting or deletion of an object. In this case, the `ArrayList<Observer> list` is much more suitable for this pattern.

A possible variant of the decorator pattern is identified when the `Component` interface and or the `ConcreteDecorator` interface isn't used. Mentioned above, this is incorrect because it removes some of the organizational practices described by the definition of the pattern. Nonsensical variable names were also tested here. Another possible variant of the decorator pattern is a `ConcreteDecorator` that did not have a base `Component`. This is incorrect because the base `Component` is integral to the implementation of this pattern. It is supposed to serve as the simplest version of the thing being created. The `ConcreteDecorator` is then wrapped around the base as many times as needed to add more behavior to the object. Without that base `Component`, the `ConcreteDecorators` have nothing in common, like they are supposed to in the implementation of this pattern.

A possible variant of the singleton pattern is identified when a new `Singleton` object is created each time `getConstructor()` is called. This defeats the purpose of using the pattern at all because the class implementing the pattern is supposed to only create one object of itself. Creating a new object each time `getConstructor()` is called would overwrite any changes made in the original object. Another possible variant is identified when the constructor isn't made private. If it's public, the creation of the `Singleton` object could bypass the `getConstructor()` method and create more than the single object. This also would spread changes made across multiple objects, which isn't ideal.

A possible variant of the command pattern is identified when an `Invoker` creates the command instead of the `Builder`. This is incorrect because there is a specific pipeline that creates and executes commands in this pattern. This organizes the responsibilities needed to create and execute the desired command. Without following this pipeline, the organization falters and creates code that is more difficult to update and read.

A possible variant of the state pattern is identified when its `ActionState` is made public. This state should be private because the `ActionState` pattern needs to be able to change its behavior according to the `ActionState` without other objects being able to predict its behavior at any point in the execution of the program.

4.5 Evaluation with Variants

Table 1 shows the patterns used in the evaluation of the effectiveness of our Mockito tool. The first column describes how the pattern is implemented with our subject. The second column shows the number of lines of code each pattern uses. The third column shows the number of tests that relate with the pattern and our subject.

Table 2 shows a checklist of each variant either passing or failing JUnit/Mockito tests. The first three columns represent each variant and whether it passed or not the JUnit test. The last three represent our Mockito tests and whether the patterns passed or not on our tests. Most of the time no subject was able to pass our tests, but was able to pass the basic JUnit tests.

5. Evaluation

To answer our research question, we conducted an empirical study on a course project which develops an interactive graphic user interface (GUI) game. There are six design patterns involved in the project and the six design patterns are divided into seven stages of development. Each stage results in 10-16 student submissions if it's an individual stage and about 6-9 submissions if it's a group stage. In total, we analyzed over 40 submissions. Section 5.1 briefly introduces the design patterns and how they are used in the project. Please consult related materials [1, 3] for more detailed explanation and examples for the design patterns.

5.1 Applications of Design Patterns

The "Aliens vs. Humans" project was designed from the bottom up using a set of six design patterns. Starting with the strategy pattern to create the humans and aliens, the project continues through the observer, decorator, singleton, command, and state patterns. The following explains the context in which each pattern exists in the project.

5.1.1 Strategy Pattern

The strategy pattern separates a family of algorithms from the containing class. It allows the client to change the algorithm on the fly. For example, the `Aliens` in the games have the ability to recover over time. There would be three different ways to recover, e.g., linearly, by fraction,

Table 1: Considered design patterns.

Pattern	Description	LoC	# Tests Related
Observer	Updates the game time	105	6
Decorator	Wraps the weapons with attachments	301	12
Strategy	Sets the Recovery Behavior of the Aliens	95	12
Singleton	Single instance of environment	355	20
Command	A set of actions that are called that alter the game	451	7
State	A set of states that lifeforms can enter and their interactions	381	12
Total		401	30

Table 2: Design pattern variants.

	Unit Testing			Behavioral Testing		
Pattern	First	Second	Third	First	Second	Third
Observer	Yes	Yes	Yes	No	No	No
Decorator	Yes	Yes	Yes	No	No	No
Strategy	Yes	Yes	Yes	No	No	No
Singleton	Yes	Yes		No	No	
Command	Yes			No		
State	Yes			No		

or no recovery. By using the strategy pattern, the three recovery strategies are organized as the subclasses of `RecoveryBehavior`. The `calculateRecovery` method in `Alien` will consult the `RecoveryBehavior` field when the alien recovers. The `RecoveryBehavior` field can be set by the client on the fly to one of the three strategies.

5.1.2 Observer Pattern

The observer pattern specifies how the listeners (observers) to events and the emitter (the subject) of the events would interact. A listener would provide a call-back in `update()`. A subject would keep a collection of the observers and use `notifyObservers()` to execute the `update()` method on every contained observer.

The game has several uses of the pattern. The first use is for tracking the time passage. The `RecoveryBehavior` comes with a `rate` for recovery, e.g., a rate of 2 means the alien recovers every two game rounds. The weapons used by the aliens and humans can only fire certain amount of shots per game round. The second use is to update the game GUI if any part of the GUI needs to be redrawn. For example, when an alien moves from one location to the other, the GUI acts as an observer of such changes and its `update` method will make sure the change is reflected on the GUI.

5.1.3 Decorator Pattern

The decorator pattern attaches additional responsibilities on the fly by wrapping the *base* in a *decorator*. The decorator is a subclass of the base so that a method `m()` in the decorator adds responsibilities to the base. In the game, the pattern handles the creation of weapons.

The `Weapon interface` handles the method signatures for `fire(), getDamage(), getRange(), getRateOfFire(), and getMaxAmmo()` methods. `Weapon` is then implemented by three concrete classes: `Pistol, ChainGun, and PlasmaCannon`, which each have their own damage, range, rate of fire, and max ammo values. Another `interface` called `WeaponAttachment`, which also `implements Weapon`, is the decorator part of this pattern. `Scope, Stabilizer, and Booster implement WeaponAttachment`. When created, it's constructor takes in a `Weapon` and stores it. The methods in `Weapon` are written to edit the values of the original `Weapon`.

5.1.4 Singleton Pattern

The singleton pattern ensures that only one instance of a class is created throughout the execution of a program. This is achieved using a private constructor and a public `getConstructor()` method is used to create a new `Singleton` object only if an instance of this class has not been created yet. The game's environment uses this pattern and includes a 2D array of `Cell` objects, which each have the ability to hold one `LifeForm` and two `Weapons`. The `Environment` can't have more than one of its instances taking up memory or soiling the state of the game, so it's constructor is private and can only be accessed through the `getEnvironment(int row, int col)` method. Further steps are then taken to make sure that `getEnvironment(int row, int col)` returns a new `Environment` object if one does not exist, and returns the existing object if one does exist. No other process can be running as `getEnvironment(int row, int col)` runs, including other calls to that method.

5.1.5 Command Pattern

Using the command pattern, we can encapsulate a request as an object to better organize and prioritize the requests. The `Client` requests some behavior which is sent to a `Receiver`. The `Receiver` then tells a builder to make the request object, and then sends it to an `Invoker`. The `Invoker` then calls all the functions needed to carry out the request in the object. This allows the user to control LifeForms movement, collection of weapons, and the firing of their weapons. There is an `Invoker` that is represented by buttons in this project's GUI. When the buttons are pressed the corresponding `Command` is `execute()` by the `Invoker`. Each `Command` has a `Receiver` that the `Command` is executed upon. The `Receiver` in this case could be different `LifeForm` objects within the `Environment` which are being moved, picking up weapons, and firing weapons.

5.1.6 State Pattern

The state pattern allows an object to change its behavior when its state changes. The state pattern in our example has an `AIContext` has an instance of `ActionState` that switches based on conditionals, e.g., HasWeapon, NoWeapon, OutOfAmmo, and Dead. The objects' behavior is determined by its current `ActionState`. The `ActionState` must be private to protect from other classes accessing that information. This is because we want only the object with the `ActionState` to know how it's going to react to other objects.

5.2 Empirical Study

We wanted to verify that student submissions followed the core behaviors of the design patterns. During the class, we would build our game and run a simple JUnit test on it to verify that the start and end of each behavior would work as intended. In doing so, student submissions could have missed the core behaviors of the patterns due to the lack of specificity in the JUnit tests. This is why we wanted to develop a tool to be able to catch those errors in behaviors.

5.2.1 Strategy

For the strategy pattern, we created Mockito tests to verify that the correct method was called, so if any other `RecoveryBehavior` was in place, it would still be called. The first step in our test was to create a Mockito spy of a `RecoveryBehavior` and then assign that spy to our `Alien`. We did this using reflection and setting the variable for recovery in `Alien` to our spied `RecoveryNone` class. After doing so, we created a `SimpleTimer` and added the `Alien` we previously created to that timer. We then had the `Alien` be attacked, and afterward had the timer change to notify all of its observers to update. This should then call `calculateReocovery()` inside the `RecoveryBehavior` of the Alien's RecoveryBehavior type. We just verified that any integer was used to indicate the current life points and the max life points of that `Alien`.

5.2.2 Observer

The observer pattern consists of an `ArrayList` of `Observers` that watch for a `Timer` to change rounds. In order to make sure this is accomplished, we created a Mockito spy of an `ArrayList` of `Observers` and using reflection, we inserted that spy into the `Timer`, which replaces the variable originally in place. This will allow us to make sure that the `MockSimpleTimeObserver` was added properly to the list. Once that test passes, we can then use black-box testing to make sure the observer was updated upon a round change.

5.2.3 Decorator

The decorator pattern was a difficult one for us to create tests for. Looking at our subject to build the Mockito tests off of, we saw a common pattern that the decorator had a variable as the `base` of the starting class before being wrapped. With this information we created a Mockito test that gathered all of the declared fields in the `Attachment` class. After this we verified that there was a variable that was named base. We also knew that a hierarchy was needed for a decorator class to be one. So we checked the hierarchy of the `PowerBooster` with a `Pistol` attached and made sure that the superclass was not `Object`.

5.2.4 Singleton

The singleton method is a simple one, in which we used reflection to make sure the `Environment` constructor was private. We then tested to make sure only one `Environment` object was created upon multiple calls to the `getEnvironment()` method.

5.3 Results

Table 3 shows the number of submissions that used our tool on and for which pattern we were testing with those submissions. The next column then shows the number of submissions that successfully passed our Mockito tool. All of the submissions were able to successfully pass our tool. This was because of the good unit testing was in place to prevent student submissions from straying away from a correct design pattern.

6. Threats to Validity

The first concern of our conclusion might be the false negatives. That is, we did not use any real-world programs with wrong implementations of design patterns, i.e., *real variants*, and can pass the tests. Rather we used *artificial variants* in Section 4.5 to show that variants that can pass the good tests cannot pass our Mockito tool. It is still possible that real variants can pass our Mockito tool.

However, in reality, most wrong implementations will not pass the tests. If they do, they are most likely correct. The first evidence is that to make the artificial variants, we must know the design patterns well. Without an appropriate level of understanding, one should not provide those variants

Table 3: Mockito Tests on Submissions that Pass all Unit Tests

Pattern	Total Submissions	Passed Mockito Tool
Observer	16	16
Decorator	15	15
Strategy	6	6
Singleton	5	5

themselves. The second evidence is analogous to the free theorem or the *parametricity* in programming languages. *Free theorem* says some properties of a program can be proved knowing only its type. For a simple example, if a function takes a parameter of type `A` and returns a value of type `A` for any type `A`, what could the function be? It has to be a function that simply returns the parameter[15]. There is no other choice here. In our case, the well-written tests are like the type information of the function and the implementations of the design patterns are like the implementation of the function. If one can pass the good tests, there are not many possibilities to provide wrong implementations of design patterns in real world programs.

The other concern is the scope of our empirical study. The subjects are student submissions in a design patterns course for one semester in one university. We use the student submissions of a design patterns course because it comes with the standard tests and a variant is always an unwanted mistake. However, we The on the artificial variants shows our behavior checking tool is effective. This can increase the scope of the empirical study by tracking more submissions from different semesters and courses.

7. Conclusion

We can summarize our conclusion as: *well-written tests can tell the correctness of design pattern implementations*. Note that the other way is not true since there could be many reasons the program fails the tests. To expand it, well-written tests means the tests have good structural and input domain coverage on the implementation. We did an empirical study involving 44 submissions in a design pattern course to support the conclusion. As it may be slightly contradictory at the first glance, we pointed out that this phenomenon could be related to the free theorem. A discussion is provided in Section 6.

In addition to our conclusion, we provide our Mockito tool and design pattern variants (the ones with unwanted design but give the same program output) in a public software repository: `https://gitlab.com/asmale/are-unit-tests-good-enough`. Readers can use the verification tool or expand the variant database.

References

[1] E. Gamma, R. Helm & R. Johnson, *Design patterns: elements of reusable object-oriented software* (Addison-Wesley, 1994).

[2] E. Gamma et al., Design patterns: Abstraction and reuse of object-oriented design, *Proc. of ECOOP '93*, Springer-Verlag, 1993, 406–431.

[3] E. Freeman & E. Robson, *Head First Design Patterns* (O'Reilly Media, 2005).

[4] S. Holzner, *Design Patterns For Dummies.* (For Dummies, 2006).

[5] W. Pree & H. Sikora, Design Patterns for Object-oriented Software Development (Tutorial), *Proceedings of the 19th International Conference on Software Engineering*, ICSE '97, Boston, Massachusetts, USA, 1997, 663–664.

[6] K. Arnout, From Patterns to Components, PhD thesis, Swiss Federal Institute of Technology Zurich (ETH Zurich), 2004.

[7] F. Lokke, Scala and Design Patterns, PhD thesis, University of Aarhus, 2009.

[8] W. Pree, *Design Patterns for Object-oriented Software Development* (Addison-Wesley, 1994).

[9] P. Ammann & J. Offutt, *Introduction to Software Testing* (Cambridge University Press, 2016).

[10] M. Huth & M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems* (Cambridge University Press, 2012).

[11] M. Young & M. Pezze, *Software Testing and Analysis: Process, Principles and Techniques* (Wiley, 2007).

[12] T. Kaczanowski, *Mockito - Open Source Java Mocking Framework*, `http://www.methodsandtools.com/tools/mockito.php`.

[13] R. Just et al., Are Mutants a Valid Substitute for Real Faults in Software Testing?, *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, 654–665.

[14] J. H. Andrews, L. C. Briand & Y. Labiche, Is Mutation an Appropriate Tool for Testing Experiments?, *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, St. Louis, MO, USA, 2005, 402–411.

[15] R. Harper, *Practical Foundations of Programming Languages* (Cambridge University Press, 2016).

FACULTY ABSTRACTS

AN INTERACTIVE METHOD FOR TEACHING AND LEARNING CRYPTOGRAPHY

Dr. Lisa L. Kovalchick
California University
kovalchick@calu.edu

ABSTRACT

CryptoClub is a project funded by the National Science Foundation (NSF). The CryptoClub project consists of classroom and web-based materials, which teach cryptography and related mathematics to students. CryptoClub uses games, treasure hunts and other informal activities to engage students in cryptography and mathematics. It applies various mathematical topics, such as decimals and percent, division with remainder, common factors, negative numbers and pattern recognition. The program also includes cryptography games and activities from the CryptoClub website, http://cryptoclub.org/.

During this session, we will provide details on the CryptoClub project. We will also demonstrate a variety of games, Internet activities and stories to engage students.

GRADUATE ABSTRACTS

SOCIAL MEDIA ANALYSIS USING TABLEAU, FACEBOOK ANALYTICS, AND FACEBOOK INSIGHTS

Joey Kerle, Dr. Jayakumar V. Annadatha
Clarion University of Pennsylvania
j.kerle1@eagle.clarion.edu, jannadatha@clarion.edu

ABSTRACT

The objective of this capstone project was to help Great Deal Tires better understand how Facebook users engage with its Facebook page. For this analysis, Tableau, Facebook Analytics, and Facebook Insights were used. Visualizations were developed by Tableau and Facebook Analytics and Facebook Insights were used to analyze data. Tableau was used to show the top performing posts and how the engaged users of the Facebook page changed over time. Facebook Analytics and Facebook Insights were used to gain more information about the audience such as common ages, genders, and location.

PREDICTIVE MODEL FOR AUTOMOTIVE INSURANCE LOSSES

Jerry Shick, Dr. Jay V Annadatha
Clarion University of Pennsylvania
j.l.shick@eagle.clarion.edu, jannadatha@clarion.edu

ABSTRACT

The objective of this capstone project is to develop an optimized model for the prediction of automotive insurance losses using SAS – Enterprise Miner & SAS Studio. For predictive model development, four machine learning techniques were implemented: Decision Tree, MBR – Memory Based Reasoning, Regression and Neural Network. The best model was chosen based on the predictive accuracy of normalized losses – the average yearly loss payment. The automotive dataset from the UCI Machine Learning repository was chosen for the analysis. Data contains information relating to vehicle make, fuel type, number of doors, body style, engine location, wheel base, vehicle length, vehicle width, vehicle height, curb weight, engine type, number of cylinders, engine size, fuel system, bore, stroke, compression ratio, horsepower, peak rpm, city m.p.g., highway m.p.g. and price in addition to normalized losses for each vehicle. The four models were analyzed using mean squared error. The neural network model was found to be the best in predictive performance followed by the decision tree model.

UNDERGRADUATE ABSTRACTS

A VENTURE INTO ANDROID VIRTUAL REALITY DEVELOPMENT

Abigail Markish
Lock Haven University
amm2351@lockhaven.edu

ABSTRACT

The poster project "A Venture into Android Virtual Reality Development" was done to further the poster creator's knowledge of the subject. Also, it was completed to serve as a stepping stone for coursework that may be offered at Lock Haven University at a later date. Viewing this poster project will allow individuals to ascertain how simple it is to grasp the essentials of virtual reality development. It is understandable to believe that developing any type of application for mobile virtual reality would be difficult, but exposure and experience have allowed this poster creator to see that it is just as simple as developing a regular android application. After powering up Android Studio and getting past some slight learning curves, it is possible to make simple scenes and 3D video viewers in under an hour.

Android Studio's ability to turn itself into a mobile virtual reality development platform is a useful trick. When given the Oculus Mobile SDK and a template starter project, a developer can use Android Studio to make any VR application that they see fit. As with any mobile application, there will be a main scene that utilizes scene objects to make a working application. Over the course of this project, many functions of the Oculus Mobile SDK were tested. Some of the functions that will be presented and explained during the poster session include: movement within a 3D space, 360-degree video viewing, 3D object lighting and orientation, and controller interactions. As stated in the previous paragraph, this project was done to sate the creator's curiosity and to open the door to future virtual reality coursework at Lock Haven University.

IMAGES CONCENTRATED WITH ENCRYPTED DATA (ICED)

Joshua Del Toro
East Stroudsburg University
jdeltoro@live.esu.edu

ABSTRACT

We created a program that blends three cryptographic and steganographic techniques. These methods are SWIFFT, SDES, and Steganography. SWIFFT is used to create a one-way hash function that can only be obtained by having the plaintext. SDES is used to produce the ciphertext. The ciphertext and hash function are then placed into an image using spiral techniques and steganography. The combination of these three methods creates a secure and efficient way to transfer data between two users.

BIRDS-OF-A-FEATHER SESSIONS

CONTINUALLY UPDATING COMPUTER SCIENCE GENERAL EDUCATION FOR INFORMED CITIZENS BEYOND 2020

Mark Jones
Lock Haven University
mailto:mjones5@lockhaven.edu

ABSTRACT

As increasingly opaque and complex computing systems become integrated into most Americans' lives, computer science becomes at least as important to study on a college campus as English or Mathematics. In this Birds-of-a-Feather discussion we will address the need for broader computer science general education. We will discuss key issues and concepts that citizens should understand. We will discuss methods to continually update such offerings in the face of rapid change.

BIOGRAPHY

Dr. Mark Jones has taught graduate and undergraduate computer science in the Pennsylvania State System for more than seventeen years. He has presented a variety of papers and led a special sessions at many previous PACISE conferences. Each year, for over a decade, he has conducted a class with the title "Contemporary Issues in Computing."

BSE IN COMPUTER SCIENCE

Blaise Liffick and Nazli Hardy
Millersville University
Blaise.Liffick@millersville.edu, Nazli.Hardy@millersville.edu

ABSTRACT

There is new interest from PDE for 7-12 teaching certification in Computer Science. There is already a Praxis exam in CS, with the list of covered topics clearly delineated. It is likely that the core courses from a typical BS in CS would be nearly sufficient in terms of technical content (not including related education courses) for providing the fundamentals for teacher certification, particularly from any programs that are CAC-ABET accredited. Although the concept of a certification for CS teachers in PA was first broached with PDE some 30 years ago, it appears that PDE has finally reached a point where they are finally interested in trying to make it happen. But there are some open questions that need to be addressed in trying to meet this apparent need:

1. Will schools be required to have certified teachers in CS at some point in the future? If so, when?
2. Will there be sufficient demand in PA for someone with a BSE?
3. Will schools have sufficient full-time work for someone with just certification in CS?
4. What about transitional programs for those already teaching CS?
5. What impact would adding a BSE degree have on current programs? What are the resources implications?

SPECIAL SESSIONS

COOPERATION WITH MINIMAL COMMUNICATION

C. Dudley Girard
Shippensburg University
cdgira@cs.ship.edu

The EV3 bots that have been used for competitions the past two years have the ability to communicate with each other with the right equipment (Netgear N150). However, the bots used in the competitions have only a limited number of sensors to choose from: 1 Color Sensor, 1 Infrared Sensor, and 1 Touch Sensor. Additionally, they have a limited number of motors to work with as well: 2 Large motors and 1 medium motor. Lastly, each EV3 kit came with an IR beacon.

This workshop will discuss and try out different approaches to getting two or more bots to solve a problem together. The core problem will involve moving a ball between walled off areas and lifting the ball to a final location. Tape lines will be used to help guide the bots. The problem will likely involve 3 different bots having to work together. Additional details on the problems will be sent out ahead of time to give those interested time to think of and develop potential solutions (which will be difficult to test given most of us only have one EV3 bot).

POSTER SESSIONS

EVOLVING NEURAL NETWORKS FOR BETTER FUZZING

Connor Billings[1], Stephanie Schwartz[1], Edward J. Schwartz[2]
Millersville University[1], Carnegie Mellon University[2]
cjbillin@millersville.edu, stephanie.schwartz@millersville.edu, edmcman@cmu.edu

ABSTRACT

Computer programs are becoming more and more complex and so are the bugs and security vulnerabilities in these programs. Testing suites will typically be written for massive code bases, but while these tests can assert that code executes as expected, they themselves cannot produce test cases that present vulnerabilities. This is where program fuzzers become useful. Fuzzers produce test cases for a program in the hopes that a test case will trigger a vulnerability. However, with an infinite number of potential inputs, a fuzzer typically needs initial test files as seeds to get started. Our project hopes to improve this initial stage of seeding by producing various high coverage files as seeds. By using the NEAT neural network algorithm paired along with the AFL Fuzzer, we are training a network to learn to represent the file structure that a program would expect as input. By representing a file as a network, we can learn which bytes in a file are dependent on other bytes and how changing one changes the other. Some of the core difficulties of addressing this project include generating useful training input, defining what "good" code coverage is, and scoring the fitness of a network. Once these are overcome, such a network would allow for the creation of many high coverage, unique seeds for fuzzing and would make debugging code more automated, efficient, and complete.

SLIMY - A SIMPLE 2D PLATFORMER

Aaron Gerber
Shippensburg University Student - Undergraduate
ag0612@ship.edu

ABSTRACT

Slimy is a 2d platformer, where a small little slime is trapped in a box with spikes. The game revolves around Slimy trying to avoid being hit. As the player stays alive longer, their score increases and the number of spikes being shot at slimy increases. Once a player has been hit three times, slimy dies and they need to restart. Slimy is able to move left, right and is able to jump as part of it's movement options.

A TWO-PHASE SYMMETRIC KEY CIPHER

Kyle Guers, Michael Pokrinchak, and Eun-Joo Lee
East Stroudsburg University
kguers@live.esu.edu, mpokrincha@live.esu.edu, elee@po-box.esu.edu

ABSTRACT

In cryptography, a symmetric key cipher uses a shared secret key to encrypt and decrypt messages. In this two-phase symmetric key cipher, a combination of two different methods to strongly encrypt and decrypt our message. The first phase involves using Linear-Feedback Shift Register (LFSR) key generation and utilizes a Modified New Symmetric Key Algorithm (MNSKA). MNSKA alters the block of plaintext by combining binary ASCII values and dividing it by the generated key. The second phase implements the Double Transposition Algorithm by ordering the MNSKA ciphertext into a transposition table and shuffling the text out of order; the transposition algorithm is applied a second time to form our final ciphertext. Doing the phases in reverse order reverts ciphertext back to the plaintext via decryption.

MUSIC PLAYING APPLICATION POSTER

Vincent Hornak, John Carelli
Kutztown University
vhorn843@live.kutztown.edu, carelli@kutztown.edu

ABSTRACT

This poster presents a music application, designed in Python, that includes a multi-touch interface for playing chords selectable by the user through a simple button press, as well as a piano roll for playing arpeggios. It makes use of the Kivy extension to Python for developing touch-based applications and the Mido library for MIDI-based sound generation.

The application presents the user a layout of pre-selected chords determined by analysis to be most commonly used. The specific chords displayed depends on the song key, which is selectable by the user, and updates when the key is changed. The notes in the piano roll are limited to those in the currently selected chord and updates whenever a new chord is selected. It can be used to play individual notes in the current chord or as a strum plate.

Determination of most commonly used chords involved consideration of basic musical principles involving related musical keys in addition to data analytics. For the latter, a database of 6500 popular songs was analyzed to determine frequency of chord use.

The primary goal is to present musicians with an easy-to-use application for playing and composing music. It also provides an approachable platform on which educators can teach novices basic music theory involving chord progressions.

ACADEMIC PROGRESS TRACKER FOR ATHLETICS

Tamara Jennings
Kutztown University of Pennsylvania
tjenn300@live.kutztown.edu

ABSTRACT

Student athletes that wish to participate in practices and competitions at the collegiate level must abide by the academic eligibility standards outlined by their institutions and the organizations that regulate college athletics. The responsibility of maintaining this eligibility falls on student athletes and their coaches. This can prove to be a challenge, especially for coaches that must obtain consistent and timely updates from multiple student athletes. Currently, there are athletics programs that rely on inefficient paper submission systems and emails to professors to confirm academic progress. The Academic Progress Tracker for Athletics looks to streamline the process by utilizing a MySQL database and JavaScript/PHP development technologies to provide a web application. The application includes two user experiences– one for the student athlete and one for the coach. Student athletes will utilize the application to enter course and assignment information and view their progress. Coaches will utilize the application to manage users, view academic progress, and create reports. The application is intended to increase efficiency and productivity by collecting and displaying academic data in an online application.

CLUSTERING AND CUSTOMER SEGMENTATION: A B2B CASE STUDY USING SAS ENTERPRISE MINER

Joey Kerle, Dr. Jayakumar V. Annadatha
Clarion University of Pennsylvania
j.kerle1@eagle.clarion.edu, jannadatha@clarion.edu

ABSTRACT

The object of this capstone project was to transform and filter variables in order to find critical information such as variable worth and node segment profile to help a firm best target their 100,000 B2B customers without customizing materials for each individual B2B customer. For this analysis, clustering and segment profiles were used. Clustering was used to group customers who were most like each other. Segment profiles were used to refine the analyses in order to project the most effective marketing messages possible to the customers. Some of the most important findings in variable worth showed to be RFM, years purchased, and revenue class. By using clusters and segment profiles in this predictive model, the marketing team of the firm is better able to target like-kind customers with marketing materials, which is more efficient while targeting B2B customers and saves the firm money.

BEST POSTER AWARD

Best poster: Connor Billings, Stephanie Schwartz, and Edward J. Schwartz

EVOLVING NEURAL NETWORKS FOR BETTER FUZZING

PROGRAMMING COMPETITION

PROGRAMMING CONTEST PARTICIPANTS

SCHOOL	COACH	TEAM NAME	MEMBERS
East Stroudsburg University	Robert Marmelstein	ESU Code Warriors	Alex Morales
			Kurtis Trego
			Nick Khouri
Edinboro University	Dan Bennet	Caber Toss	Rebecca Aloisio
			Brandon Dankot
			Riley Vaughn
		Two and a Half Programmers	Gabriel Dougherty
			Christopher Persinger
			Ian Warner
Indiana University	David Smith	Crimson Coders	Jabari Jackson
			Eric Ritchey
			Tim Valentine
Kutztown University	Dylan Schwesinger	Defcon 1	Christopher Beaudoin
			Aaron Raff
			Warren Ziegenfus
Lock Haven University	Krish Pillai	Bald Eagles	Cassidy Lindner
			Zachary Packer
			Nicholas Page
Millersville University	William Killian	Biggie Cheese	Matt Fossett
			Zachary Mixa
			Jimmy Roche
		C++ Squad	Anthony Burnett
			Sean Malloy
			Jesse Schnupp
		It's Not Efficiency Club	Connor Billings
			Dan Hartenstine
			Henry Schmale

SCHOOL	**COACH**	**TEAM NAME**	**MEMBERS**
Shippensburg University	Chen Huo	Java Ship	John Gable
			Kim O'Neil
			Michael Permyashkin
		Ruby on Raiders	Matt Angle
			Anthony DePaul
			Steven Hetrick
Slippery Rock University	Abdullah Wahbeh	Runtime Terror	Kenzo Balerdi
			Kunj Champaneri
			Tanuj Rane
West Chester University	Linh Ngo	Golden Rams	Akash Kumar
			Andrew Valenci
			Gilbert Martinelli

PROGRAMMING COMPETITION WINNERS

First Place: It's Not Efficiency Club, Millersville University

Connor Billings
Daniel Hartenstine
Henry Schmale

Second Place: C++ Squad, Millersville University

Anthony Burnett
Sean Malloy
Jesse Schnupp

Third Place: Biggie Cheese, Millersville University

Matt Fossett
Zack Mixa
Jimmy Roche

SECURITY COMPETITION

PARTICIPATING TEAMS

SCHOOL	TEAM NAME	MEMBERS
Edinboro University	The Black Watch	Tessa Williams
		Anthony Criscione
		Kevin Cox
Millersville University	Just_Joe?	Joseph Dunton
		Zachary Groff
		Kasey Kocher

SECURITY COMPETITION WINNERS

First Place: Just_Joe?, Millersville University

Joseph Dunton
Zachary Groff
Kasey Kocher

Second Place: The Black Watch, Edinboro University

Tessa Williams
Anthony Criscione
Kevin Cox

BOARD OF DIRECTORS

BOARD OF DIRECTORS 2018-2019

Officers:

President	Lisa Kovalchick
Vice Pres.	Joseph Molnar
Treasurer	Soo Kim
Secretary	David Mooney

Web Master:

Nitin Sukhija

Members:

Bloomsburg University of Pennsylvania	Robert Montante	bobmon@bloomu.edu
California University of Pennsylvania	Lisa Kovalchick	kovalchick@cup.edu
Clarion University of Pennsylvania	Soo Kim	skim@clarion.edu
East Stroudsburg University of Pennsylvania	Robert Marmelstein	rmarmelstein@po-box.esu.edu
Edinboro University of Pennsylvania	Joseph Molnar	jmolnar@edinboro.edu
Indiana University of Pennsylvania	Soundararajan Ezekiel	sezekiel@iup.edu
Kutztown University of Pennsylvania	Joo Tan	tan@kutztown.edu
Lock Haven University of Pennsylvania	Mark Jones	mjones5@lhup.edu
Mansfield University of Pennsylvania		
Millersville University of Pennsylvania	David Hutchens	hutchens@cs.millersville.edu
Shippensburg University of Pennsylvania	David Mooney	djmoon@cs.ship.edu
Slippery Rock University of Pennsylvania	Nitin Sukhija	nitin.sukhija@sru.edu
West Chester University of Pennsylvania	Richard Burns	rburns@wcupa.edu

AUTHOR INDEX

AUTHOR INDEX