



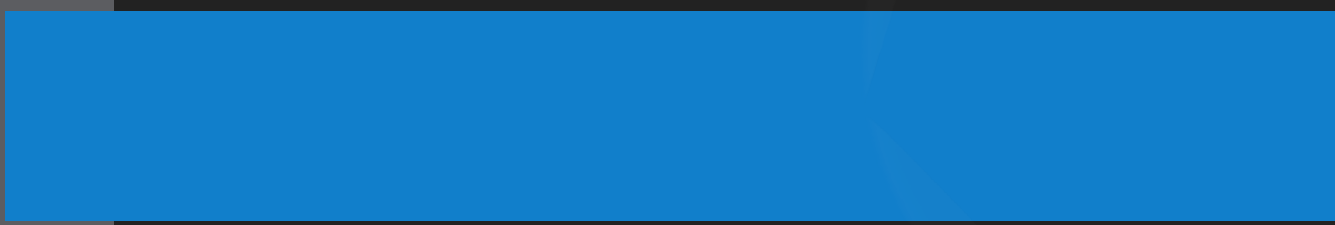
*Hack Research*  
HACKR2022

---

Proceedings of the 3<sup>rd</sup> Hack Research Hackathon  
University of Texas Rio Grande Valley

*HackR* 2022

Tim Wylie, Editor



Copyright © 2022 for the individual papers by the papers' authors. Copying permitted only for private and academic purposes. This volume is published and copyrighted by its editors.

## Preface

Hack Research took a two-year hiatus, as most things did, because of the pandemic. This year required restarting some of the machinery that had made the first couple successful. Even though there may be a record number of students, there are, overall, fewer engaged in research due to less engagement between students and faculty during the past two years.

With that in mind, Hack Research proved to be a good platform to begin some of this engagement and to find problems that are accessible for new students in order to bring them into different research areas. We had more faculty participation, and thus a greater variety of problems, than we have had before.

The goal of Hack Research is to have a competition focused on algorithmic and theoretical skill development rather than software. This gives students who excel in these areas an opportunity to apply that knowledge, and provides a meaningful way to connect with faculty and their research. This focus, however, does not mean there is a lack of software development; many of the problems were based in software with machine learning, data mining, and visualization applications. These problems target the development of research skills rather than a single unsolved question.

The questions are posed by faculty and students in various research groups. Finding and proposing problems requires a difficult balance between it being interesting and yet approachable. Attempting to find such open problems is a struggle when the problems should be nontrivial. This year, there was a greater variety of problems and all of them were better scoped for the timeframe of the event. There was more focus on exploratory skills such as data mining, but there were also many small approachable theoretical problems.

As usual, we limited our advertisement of the event to upper level courses and word of mouth. Thus, although the event was open to anyone, most of the students already had relationships with the professors from courses. This is another tricky aspect of the event. We do need to advertise more, but we also need to make sure students understand the point of the event.

Many people contributed to the success of the event, and I have tried my best not to miss anyone in the acknowledgements section. We were graced with several key sponsors this year - GitHub, Major League Hacking, Overleaf, DHS, and UTRGV all contributed in one way or another. I am indebted to the support of the CS department and its chair, Dr. Emmett Tomai, and to the vast amount of organization help from the administrator Ms. Lisa Moreno.

Overall, *HACKR* had a successful return, and we learned a substantial amount in the process. We are looking forward to next year.

December 3-4, 2022  
Edinburg, TX, United States

Tim Wylie, Editor

## Acknowledgements

This event would not have been possible without the help of many people of whom we are extremely grateful to know. Without a doubt, the event would not have been possible without the understanding and support of the Computer Science department. They have consistently supported and helped fund the event. Dr. Emmett Tomai, the department chair, championed support and funding for the event while we were scrambling to make it happen. His work and the department made the event possible.

The other person who was instrumental in forging the event into a reality was Lisa Moreno, who is the Administrator of the Computer Science department. She took over a huge portion of the administrative aspects this year and saved us hours and hours of work. She arranged purchasing, food, reservations, delivery, and all accounting. Hack Research has grown enough to be unruly when planned without help, and she made it possible for us to focus on the bigger items.

There are several faculty who contributed their time and skills to making this event a success. In no particular order, problems were contributed by Robert Schweller, Marzieh Ayati, Qi Lu, Emmett Tomai, Bin Fu, Eric Martinez, and Tim Wylie. Many of these faculty, as well as Erik Enriquez, came to most of the event to help the students.

We would be remiss if we didn't mention our families, who were supportive and sacrificed a great deal for us to pursue these outreach activities, and we are thankful for them and their patience. Our students in the ASARG group also volunteered a lot of time to help organize and run the event. No one will forget the trips to get food in pouring rain.

Finally, we are grateful for the sponsors who gave us the support we needed to make the event a reality. As mentioned, the Computer Science department at UTRGV backed the effort both monetarily and with the support of faculty and staff. Overleaf graciously sponsored us a second year with material and some money for prizes. GitHub sent us classroom material for the teams. Major League Hacking sponsored the meals at the event (as a Pizza Fund event) and also handled registration through their website. The rest of the support came from the Department of Homeland Security, who are encouraging outreach events through DHS grant number 21STSLA00009-01-00.

## Sponsors

The University of Texas  
Rio Grande Valley

<https://www.utrgv.edu/csci>



<https://www.mlh.io>



Collaborative writing and publishing

<https://www.overleaf.com>



<https://www.dhs.gov>

# GitHub

<https://www.github.com>

DHS grant number 21STSLA00009-01-00

## Organization

*HackR* 2022 was organized by the Algorithmic Self-Assembly Research Group (ASARG) with the help of the Department of Computer Science at the University of Texas Rio Grande Valley. The event was graciously hosted by UTRGV in Edinburg, TX. Further, the event was supported in part by Major League Hacking (MLH), Overleaf, GitHub, and the Department of Homeland Security (grant number 21STSLA00009-01-00).

The Program Committee are the faculty that judged the submissions. Those faculty also submitted problems along with the faculty listed in Volunteers. Please note that the ASARG members are default volunteers.

### Directors

Tim Wylie	timothy.wylie@utrgv.edu	University of Texas Rio Grande Valley
Lisa Moreno	lisa.moreno@utrgv.edu	University of Texas Rio Grande Valley

### Program Committee

Marzieh Ayati	marzieh.ayati@utrgv.edu	University of Texas Rio Grande Valley
Robert Schweller	robert.schweller@utrgv.edu	University of Texas Rio Grande Valley
Emmett Tomai	emmett.tomai@utrgv.edu	University of Texas Rio Grande Valley
Qi Lu	qi.lu@utrgv.edu	University of Texas Rio Grande Valley

### Volunteers

Catalina Cantu  
Erik Enriquez  
Bin Fu  
Eric Martinez  
Scarlett Quintana

### ASARG Members

Robert Michael Alaniz  
Ari Gutierrez  
Elise Grizzell  
Chris Hinkle  
Andrew Rodriguez

## Results

Hack Research is still experimental, and therefore registration was restricted to maintain a small size. The event had around 70 students participate in the 24-hour event. Of those students, 20 students submitted a paper to compete for the prizes. In total, there were 9 submissions. Seven professors contributed problems, and the faculty attended some portion with several staying the majority of the time in order to encourage and assist students. Those faculty are listed in the program committee (who also judged the submissions), and as volunteers.

## Winners

There were many quality papers submitted. The judges narrowed it down to these papers as the top submissions based on quality, effort, teamwork, and difficulty.

### 1. First Place

- Anime Recommendation  
*Kolade Alabi, Kofi Nketia*
- The Cycle of Inequality Constraint Games\*  
*Andrew Rodriguez, Robert M. Alaniz*

### 2. Second Place

- Anime Recommendation  
*Ryan Knobel, Luis Ruiz, Mike Panuelos, Juan Perez*
- Student Data Visualizations\*  
*Elise Grizzell, Gaukhar Nurbek, Nicholas Houghton*

### 3. Third Place

- Robot Teams and the Bug Traps  
*Angel Peredo, Konrad Phillips-Lenz*
- An Initial Foray into Object Detection  
*Kyara Valdez*

\* *Graduate Team*

## Table of Contents

<b>Obtaining Useful Insights from Historical UTRGV Data</b>	<b>1</b>
Miguel Ramirez, Hector Hinojosa . . . . .	
<b>The Cycle of Inequality Constraint Games</b>	<b>2</b>
Andrew Rodriguez, Robert M. Alaniz . . . . .	
<b>A Beginner’s Guide to AI, By a Beginner</b>	<b>4</b>
Simon Elizondo . . . . .	
<b>Anime Recommendation</b>	<b>6</b>
Ryan Knobel, Luis Ruiz, Mike Panuelos, Juan Perez . . . . .	
<b>An Initial Foray into Object Detection</b>	<b>8</b>
Kyara Valdez . . . . .	
<b>Robot Teams and the Bug Traps</b>	<b>9</b>
Angel Peredo, Konrad Phillips-Lenz . . . . .	
<b>Student Data Visualizations</b>	<b>11</b>
Elise Grizzell, Gaukhar Nurbek, Nicholas Houghton . . . . .	
<b>Anime Recommendation</b>	<b>13</b>
Kolade Alabi, Kofi Nketia . . . . .	
<b>An AI for 5ive Straight</b>	<b>15</b>
Jose Martinez, Brandon Tiu, Jose Amaro . . . . .	



# Obtaining Useful Insights from Historical UTRGV Data

Miguel Ramirez \*

Hector Hinojosa †

## Abstract

We analyzed a dataset containing 7 years of UTRGV anonymized enrollment data using the pandas Python library. Through this process we have isolated a few enrollment problems and curiosities that exist in the data.

## 1 Introduction

We set up our Python environment using PyCharm and installed the Pandas[1] library, which we used to select, slice and modify the provided data.

For this project we analyzed two data sets:

- Degree completion data that included student ID, major, degree of study and graduation date.
- Enrollment data containing semester, student ID, major, subject, course, section and grade.

We utilized Data Science and Statistics principles in order to establish relationships between the various fields present in the data.

## 2 Related Work

Performing frequent analysis of a university’s enrollment statistics is a key way that revenue is estimated correctly and programs receive their proper funding.

By referencing prior data we can get an idea of what future semesters could look like, and prepare accordingly.

## 3 Our Results

We calculated the pass rate of every Computer Science course offered across the dataset and isolated the following as having the lowest pass rates:

Course	Pass Rate
3333	60.02%
2322	62.50%
3334	64.33%
2380	64.49%
1370	67.21%
1380	67.67%
1170	68.32%

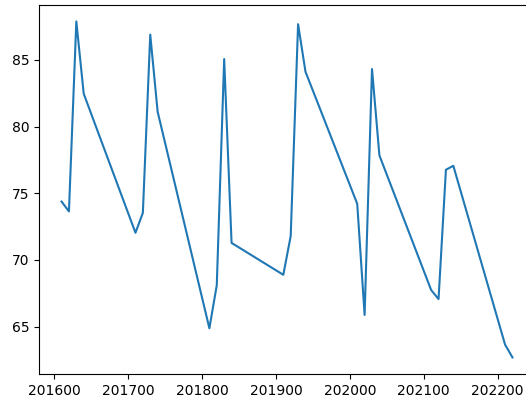


Figure 1: Fall Slump.

A prominent feature in the data that we have observed is the "Fall Slump", whereby every Fall semester there is a noticeable drop in the pass rate of students who take CS courses.

This can be correlated with some of the low pass rate courses as they tend to be courses taken by Freshmen or students in a transitory academic period. A student may believe they want to study Computer Science but change their mind after a semester of CSCI 1101 or CSCI 1170/1370 and drop those courses, leading to lower overall pass rates.

## 4 Conclusion

Taking into account these factors a pattern emerges: Many incoming Freshmen who start a course of study in Computer Science might not consider the field a good fit after experiencing the requirements in a real context. This is useful in order to explain why Fall semesters show less success in terms of academics.

## References

- [1] Pandas. <https://pandas.pydata.org>.

\*Department of Computer Science, University of Texas Rio Grande Valley, [miguel.ramirez05@utrgv.edu](mailto:miguel.ramirez05@utrgv.edu)

†Department of Computer Science, University of Texas Rio Grande Valley, [hector.i.hinojosa01@utrgv.edu](mailto:hector.i.hinojosa01@utrgv.edu)

# The Cycle of Inequality Constraint Games

Andrew Rodriguez \*

Robert M. Alaniz †

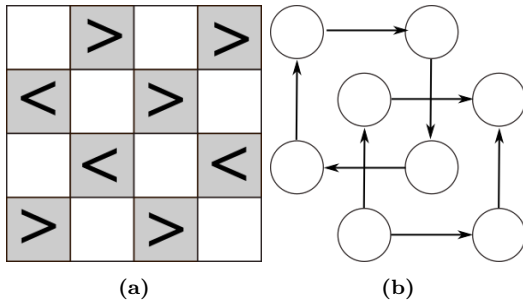
## Abstract

The Inequality Constraint game is a game where users place available numbers on a board populated with inequalities. Here we investigate the problem of *K-placement*: Given an ICG board and an integer  $K$ , can you place  $K$  pieces on the board? We show that this problem is in  $P$  by a polynomial-time algorithm. Along the way we provide an algorithm to find cycles in a given graph as a stepping stone to our final result which outputs if placing  $K$  pieces is possible.

## 1 Introduction

Recent studies in combinatorial game theory have brought attention to a type of game known as *Inequality Constraint* games. These games were first introduced in [2] with a basic  $1 \times n$  board. Current ICG works have shifted focus to problems involving  $n \times n$  boards. The 1-player  $n \times n$  ICG was shown in [1] to be solvable in  $P$ . In [3] it was shown that the Unique Fewest Moves Inequality Constraint problem is NP-complete. We study a similar problem, only instead of placing exactly  $K$  moves such that there are no more available moves, we ask can you place  $K$  moves on a given instance of the Inequality Constraint game.

## 2 Definitions



**Figure 1:** An example Inequality Constraint Game starting board is shown in figure 1a. Its corresponding graph is shown in 1b, note how a cycle appears in the upper-left graph.

### Definition 1 Inequality Constraint Game.

The *Inequality Constraint Game (ICG)* is a game where players try to fill an  $n \times n$  checkerboard with integers. This board is pre-filled with inequalities on “black” spaces, causing players to only be able to place

numbers in “white” spaces where inequalities are satisfied. The total number of “pieces” players have to play with are  $\frac{n^2}{2}$  pieces  $(1, 2, \dots, \frac{n^2}{2})$ , this also being the number of “white” spaces on the board.

### Definition 2 *k-Placement*.

Given an instance of the Inequality Constraint Game and a  $k \in \mathbb{N}$ , can  $k$  numbers be placed on the ICG board?

**Definition 3** A *cycle* is a valid sequence of vertices in which the first and last vertices are identical. A cycle  $c$  is unique if no other cycles are a cyclic permutation of  $c$ .

### Definition 4 Directed Board Graph

Given an ICG board  $B$ , we can construct a directed board graph  $G = (V, E)$  where each  $v \in V$  maps uniquely to one white space, therefore:  $|V| = \frac{n^2}{2}$ . Following methods used in [1], we can build this graph in  $O(n^2)$  time. An example of what this graph would look like can be seen in figure 1b, which was derived from figure 1a.

### Definition 5 Most Common Node

The most common node  $n_M$  in a set of unique cycles  $C$  is the node that appears in the most cycles.

## 3 Our Results

The results in [1] show us how to find if a given puzzle has a solution in  $O(n^2)$  time. This is equivalent to us being given a  $k$  equal to  $\frac{n^2}{2}$ . However, while these results would hold for  $k \leq \frac{n^2}{2}$ , they would not hold if our constructed graph contains any cycles in it. This is important as these instances of the game may still have a solution with  $k$  numbers (iff  $k < \frac{n^2}{2}$ ). We noticed that given a graph with cycles, you could put at most  $\frac{n^2}{2} - C$  numbers, where  $C$  is the number of unique cycles in our graph.

Here we give our algorithm to find cycles in a directed board graph in theorem 1, followed by theorem 2 which returns how many numbers you can place on a board in  $O(n^6)$  time. We wrote a version of the algorithm from theorem 1 which can be found in this [GitHub repository](#).

Informally, we first turn the ICG board into a graph. We then find all of the unique cycles in that graph. The idea is to remove nodes from the graph to break all of the cycles. However, choosing to remove a node that appears the most in the cycles may break several cycles rather than just one. We repeat this removal process until all cycles have been broken. The remaining nodes in the graph can then all be used to place a piece.

**Theorem 1** Let  $G = (V, E)$  be a graph where every  $v \in V$  has a max degree of 4. There exists an algorithm that finds all unique cycles in  $G$  in  $O(|V|^2)$  time.

\*Department of Computer Science, University of Texas Rio Grande Valley, [andrew.rodriguez09@utrgv.edu](mailto:andrew.rodriguez09@utrgv.edu)

†Department of Computer Science, University of Texas Rio Grande Valley, [robert.alaniz01@utrgv.edu](mailto:robert.alaniz01@utrgv.edu)

**Proof.** We prove existence by giving the algorithm.

With each node having a max degree of 4, we know that a starting node has a maximum of 4 choices for paths, and every reachable node has a maximum of 3 choices for paths. Further, we know that a cycle will not repeat any nodes.

A brute force algorithm will find all cycles in  $G$  with a worst case runtime of  $|V| \times (4 \times 3(|V| - 1))$ . The cycles can be refined into a set of unique cycles  $C$  with a worst case runtime of  $|V|^2$ . The final runtime is  $\mathcal{O}(|V| \times (4 \times 3(|V| - 1)) + |V|^2) = \mathcal{O}(|V|^2)$ .  $\square$

**Theorem 2** Given an  $n \times n$  ICG board  $B$  and some integer  $k$ , there exists an  $\mathcal{O}(n^6)$  time algorithm that determines if  $k$  pieces can be placed on  $B$ .

**Proof.** We prove existence by giving the algorithm.

Construct a graph  $G_B = (V, E)$  from  $B$  in  $\mathcal{O}(n^2)$  time. By Theorem 1, we can find the unique cycles of  $G_B$  in  $\mathcal{O}(|V|^2)$  time. Let  $C$  be the set of unique cycles computed by the algorithm in Theorem 1.  $|C|$  will never exceed  $|V|$ . The size of each cycle in  $C$  will never exceed  $|V|$ .

If  $|C| = 0$ , output “Yes” if  $k \leq |V|$ , output “No” otherwise. If  $|C| > 0$ , find the most common node in  $C$ .

Trivially, the most common node  $n_M$  in  $C$  can be computed in  $\mathcal{O}(|V|^2)$  time, arbitrarily breaking ties. Remove  $n_M$  from  $V$  to create  $V'$ . Remove every edge  $e = (u, v) \in E$  where  $u = n_M$  or  $v = n_M$  to create  $E'$ . Compute  $C'$  by running the algorithm to find unique cycles on graph  $G' = (V', E')$ .

If  $|C'| = 0$ , output “Yes” if  $k \leq |V'|$ , output “No” otherwise. If  $|C'| > 0$ , repeat the process by finding the most common node in  $C'$ .

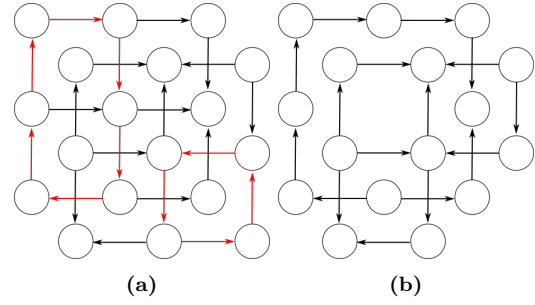
In the worst case, we repeat this process until we run out of nodes, resulting in a runtime of  $\mathcal{O}(|V| \times |V|^2) = \mathcal{O}(|V|^3) = \mathcal{O}(n^6)$ .

The remaining count of vertices is the max amount of pieces that can be played on the given ICG board. Therefore, for the current graph  $G = (V, E)$ , if  $k \leq |V|$  we can output yes.<sup>1</sup>  $\square$

## 4 Conclusion

We show that determining if a given  $n \times n$  instance of ICG can place  $k$  numbers is in P even if the directed board graph has cycles. We provide a polynomial time algorithm to find cycles specifically in directed board graphs. We further provide the full algorithm of deciding if  $k$  pieces can be placed.

<sup>1</sup>Should you want the algorithm to also output a winning strategy, we can follow the final process in [1]. This does not affect our runtime.



**Figure 2:** Figure 2a highlights the cycles found in this given directed board graph. After applying the algorithm described in theorem 2, our resulting graph would look like figure 2b. Counting the remaining vertices tells us if we can place  $K$  pieces.

## References

- [1] C. Chalk and B. Wang. One-player constraint inequality is easy. pages 7–8, 2018.
- [2] A. Levitin. *Algorithmic Puzzles*. 2011.
- [3] R. A. T. G. R. S. T. Wylie. Fewest moves inequality constraint puzzles. 2021.

# A Beginner's Guide to AI, By a Beginner

Simon Elizondo \*

## Abstract

What was once thought to be a pipe dream in the computer science world has now become a reality. What used to take days and months to compile to now mere minutes. A notable fact being that AI has gone from a master level computer science subject to now being used by intermediate level developers. The many changes into how deep learning AI has changed has become exponentially more efficient.

## 1 Introduction

Due to the recent advances in AI, it has become possible for intermediate level developers to train and deploy deep learning AI

At a high level we can view a normal computer program as "figure one". The User inputs data, the program calculates and manipulates the data then exports the data



Figure 1: A Normal Computer Program at a High Level[1]

An early interpretation can be seen in figure 2 where we can see that there are many added components to the model when compared to a normal computer program.

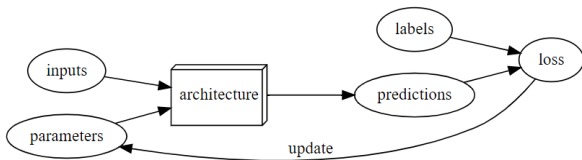


Figure 2: Early High Level Deep Learning Model[1]

But after years of iterations, a modern deep learning module at a high level can be visualized as figure 3

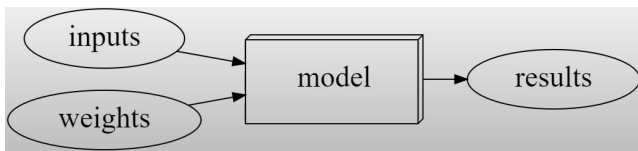


Figure 3: Modern High Level Deep Learning Model[1]

\*Department of Computer Science, University of Texas Rio Grande Valley, simon.elizondo01@utrgv.edu

## 2 Related Work

On a high level, Deep learning AI can be visualized as "Figure 4" Where it can be repeated multiple times in order to get as accurate of a AI as possible



Figure 4: High Level Deep Learning Process[1]

The concept itself is easy, Train the module, validate the data, then test the results, and the more times the model is tested and the larger the sample size, the more accurate the module will be.

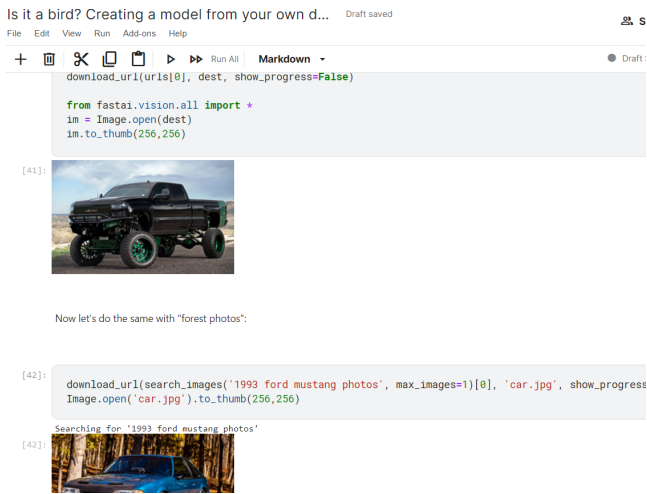
## 3 Related Work

During the hackathon i took the first two lessons of the FastAI online course which has been taught by Jeremy Howard. In these lessons I watched videos explaining the concepts of deep learning AI and was given a hands on experience with the interactive notebook version of the FastAI text book. The Notebook allows for manipulation of the code and can be used with services like Kaggle and Google Colab to offer free ways to learn and test deep learning modules for free if the user does not have a dedicated GPU. Withing these interactive lessons I was able to manipulate the deep learning values from comparing birds and tree, to comparing lifted Trucks to 1993 Ford Mustang.

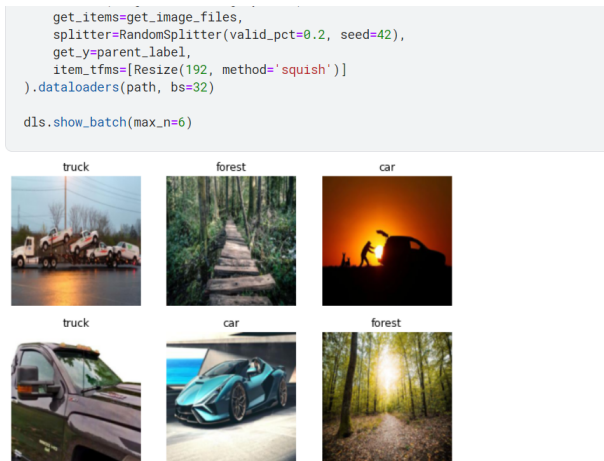
## 4 My Results

AI has come a long way since its first inception, and it is constantly improving. we are currently at a point where intermediate developers can get ahold of deep learning module documentation and begin development on AI projects and study the concepts.

There are some complications in the deployment of the simple deep learning model. This is to be expected since new versions of the FastAI and its related modules are constantly being deployed and debugging had to be done to achieve a successful deployment of the deep learning model. It was an enjoyable learning experience. Since I had no prior knowledge with working with AI it was a bit challenging understanding and deploying a program in such short time frame, but was an enjoyable learning experience.



**Figure 5:** Changing Search values to lifted trucks and a 1993 Ford Mustang



**Figure 6:** Running the comparison

## 5 Conclusion

While there were issues in the deployment of the deep learning Model was unsuccessful, there was success in my future interest in working with other AI models and AI related programming.

## References

- [1] H. Jeremy. Deep learning for coders with fastai and py-torch: Ai applications without a phd, 2020.

# Anime Recommendation

Ryan Knobel \*

Luis Ruiz †

Mike Panuelos ‡

Juan Perez §

## Abstract

We were given two data sets containing information pertaining to all anime and a user’s past watch history. Using this information we were tasked to design a system that takes this data and provides recommendations to users based on what they enjoy.

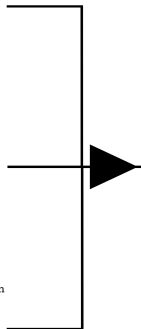
## 1 Introduction

Data mining has contributed to an abundance of valuable information by revealing patterns within large sets of data through the process of finding anomalies, patterns, and correlations, allowing the prediction of outcomes. Many models have been created for streaming services that recommend shows and movies based on a user’s history to understand why they watched or didn’t watch and enjoyed or didn’t enjoy what they viewed. Similarly, if you were given a user’s data that shows what anime they watched and how they rated it and also have access to the anime’s data such as its overall average rating by all the users that have watched it, the genres it pertains to, its popularity, and etc. Our goal is to develop a model that can determine whether a specific anime is a good recommendation for a user and then use this model to generate a recommendation for the user.

**Demon Slayer**  
Rating: 8.62  
Genres: Action, Demons, Historical, Shounen, Supernatural  
Popularity: #24

**Naruto**  
Rating: 7.91  
Genres: Action, Adventure, Comedy, Super Power, Martial Arts, Shounen  
Popularity: #24

**Jujutsu-Kaisen**  
Rating: 6.95  
Genres: Shounen  
Popularity: #4283



**Hunter x Hunter**  
Rating: 8.42  
Genres: Action, Adventure, Super Power, Fantasy, Shounen  
Popularity: #319

## 2 Model Construction

Our initial discussion sought to conclude what data from an anime could best provide insight into the user’s experience and explain why the user chose to watch it for the length that they did and rate it the score that they gave. With many contenders such as the anime’s rank, the number of episodes, the producer, and the time period that it was aired, we concluded that our initial model should be based on the anime’s genre and score (the average rating given by all users). Our model considers two features. The first feature is composed of the user’s top 25% favorite genres and the average rating of all the anime that they watched. The second feature is composed of an anime’s genres and its score.

## 3 Testing Procedure

To test the performance of our model, we first construct a testing data set with the provided user history. Each instance of the training data for the model intuitively can be broken into two components: what the user enjoys on average, and information about a specific anime.

### 3.1 Computing a User’s “Average Anime”

Given a user’s watch history, we consider every anime that the user rated a 7 or better. From this set, count the appearance of each genre, assigning a value of 1 for the most common genres (top 25 percent) or 0 for all other genres. Store these values in an array, where each position in the array corresponds to a different genre. Additionally, compute the average cumulative rating (across all users) for each anime that the user rated highly and store this value in the array.

UserID	AnimeID	Score	WatchedEpisodes
0	20	8	150
0	38000	9	24
0	38777	10	3

### 3.2 Computing Specific Anime Information

For any instance of anime, store the genres as binary values in an array, where each position in the array corresponds to different genre and the ordering matches the array described in section 3.1. Similarly, store the cumulative rating for the anime given by all users in this array as well.

### 3.3 Training Data for the Model

Using the methods described in sections 3.1 and 3.2, it is now possible to construct a feature array that our model can use as input to output whether the anime would be a

\*Department of Computer Science, University of Texas Rio Grande Valley, ryan.knobel101@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, luis.ruiz02@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, mike.panuelos01@utrgv.edu

§Department of Computer Science, University of Texas Rio Grande Valley, juan.m.perez02@utrgv.edu

good recommendation for the user. Since “good” recommendations are subjective, we define the following procedure to train the model:

- For each individual user:
  1. Compute the user’s “Average Anime”
  2. For any anime that the user has rated and watched, compute the respective information
  3. If the user rated the anime above a 7, consider this a good recommendation (1). Otherwise, consider this a bad recommendation (-1)

Given this procedure, we can then create 2 sets of data. The first data set contains the information about the user and each anime, and the second data set contains the corresponding label for the recommendation. The model then intuitively can make predictions for some instance of anime conditionally on what the user enjoys in order to make the best prediction possible.

#### 4 Performance

To test the performance of our proposed model, we split the data set into two different types of sets: a training set (80 percent of the data) and validation set (20 percent of the data). The training set is used to train the model, which includes both the user/anime data and the recommendation label. The validation set on the other hand is used to measure the performance of the model. Here, we provide the model solely with the user/anime data, make predictions using the model, and compare the results to whether the user enjoyed the anime. As a result, performance is decided by the accuracy of the predictions made by the model. Using 300,000 samples total, our model was able to predict whether an anime was a good recommendation for a user with 72 percent accuracy.

#### 5 Conclusion

Our model considers 2 factors when determining whether an anime is a good recommendation for a user. However, there are many other factors that can lead to a user either enjoying or not enjoying the anime. The process of constructing the model revealed this notion, which we leave for future work to potentially yield a higher prediction accuracy. The foundation of the model was further adapted to not only make a prediction for specific anime instances but also provide an anime recommendation that our model determines to be good. As shown in the image in the Introduction and the table in Section 3.1, our model was able to recommend the anime, “Hunter x Hunter” to a user who has watched “Demon Slayer” and scored it an 8, “Naruto” and scored it a 9, and “Jujutsu-Kaisen” and scored it a 10.

#### References

- [1] G. Akutami. Wiki Targeted (Entertainment).
- [2] Shueisha. Hunter A Hunter (Volume).
- [3] Viz. Naruto (Volume).

□

# An Initial Foray into Object Detection

Kyara Valdez \*

## Abstract

We attempted to incorporate object detection with images shot by a Tello drone. We integrated the YoloV5 framework for object detection with our given footage. The obstacles we met were the limitations of hardware and compatibility issues.

## 1 Introduction

Object detection is one of Computer vision’s techniques, used for identifying, locating and classifying objects in an image. For this research, we are integrating Yolov5, one of the models inside the Yolo family, an architecture primarily used for object detecting. We attempted to use Yolov5 in Python’s IDE, PyCharm, but ran into hardware issues. Our main objective was to improve the performance of the previous system in the DHS research project.

We used the images from a dataset provided for us and train our Yolov5 model to give better accurate results. Yolov5 is the default base model in the 5th version of the Yolo family but since most of our dataset images were large images, we needed to use the Yolov5s6 model. Our decision was based on the pixel size of most of the images, the P6 output layer is better suited for large images over 1280 pixels and there’s subcategories within the P6 output layer: nano, small, medium, large, extra-large. Although the P6 model is meant for larger images we needed the smallest subset of it, capable of giving good output results while also maintaining decent speed and precision.

## 2 Related Work

Using Python’s IDE, PyCharm, we cloned the GitHub repository to the Yolov5 framework onto a new virtual environment. We installed the proper packages using pip and installed PyTorch to get the best production process throughout development. At this stage during development, we noticed many difficulties such as the CPU usage instead of GPU usage.

In deep learning models, we’ve noted that the GPU is essential for achieving efficient speeds. In specific, NVIDIA GPU’s are the best in the field to train deep learning models, due to the fact that the recent NVIDIA cards from the RTX 2000 line of video cards and forward have CUDA cores integrated internally. CUDA, otherwise known as Compute Unified Device Architecture, is admired for its ability to compute processes in parallel.

\*Department of Computer Science, University of Texas Rio Grande Valley, kyara.valdez01@utrgv.edu

Our compatibility issues were the lack of updated resources from CUDA Toolkit, torchvision and torch only supports from CUDA version 11.7 and under. The available CUDA version on the Computer system used was 12.0 and downgrading wasn’t possible, other than the fact that PyCharm isn’t the most optimal Python IDE for CUDA.

## 3 Our Results



Figure 1: Example of validated images, this image help evaluate how effective the model is doing with the given footage

With this knowledge, we attempted to override device settings within the Yolov5 code to use the system’s RTX 2060 with CUDA enabled however many errors came forward. The only possible solution was using the CPU instead of the GPU, even though CPU usage was too hardware-intensive for the system to handle. Despite these issues, some training was possible and some output images were able to train but not with the best outcome.

## 4 Conclusion

With all the difficulties and obstacles, there’s a better understanding on how hardware affects deep learning models. The Yolov5 model, even though it wasn’t trained enough to what was expected, it gave some accuracy levels to the classifications specified in the VisDrone.yaml file.[1]

## References

- [1] H.-K. Jung. Improved yolov5: Efficient object detection using drone images under various conditions, 2022.



# Robot Teams and the Bug Traps

Angel Peredo

Konrad Phillips-Lenz

## Abstract

We proposed a deterministic algorithm to reach the destination avoiding obstacles.

## 1 Introduction

Goal: We designed an algorithm to plan a path (e.g., Path 1 or Path 2) from a mine location to the base. The algorithm is designed based on the capabilities of the robots. It works for avoiding rectangular, triangular, and bug trap shapes.

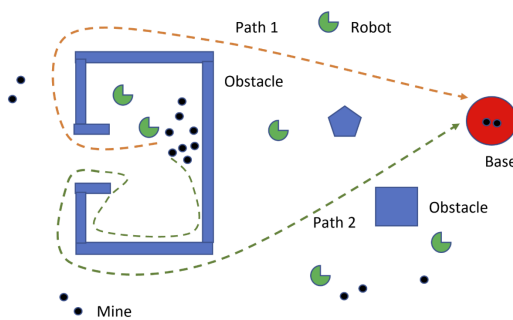


Figure 1: This is an example map, with a trap.

## 2 Set of rules

- State 1:
  - if all cells are empty: move towards the goal.
  - if normal wall equal to 3 sensors: save position(x,y) of obstacle if closer to goal. change to state 2
  - if angled wall greater than 3 sensors: save position(x,y) of obstacle if closer to goal. change to state 3
- State 2:
  - change direction to left  
move one step
  - check right
    - \* if wall right: check forward
    - \* if no wall forward: move forward  
else change to state 1
  - elif no wall right  
change direction to right  
move 1 step

- State: 3
  - move perpendicular to the wall
  - check perpendicular
  - if !wall:
    - \* check forward
    - \* move forward
  - else: change to state 1

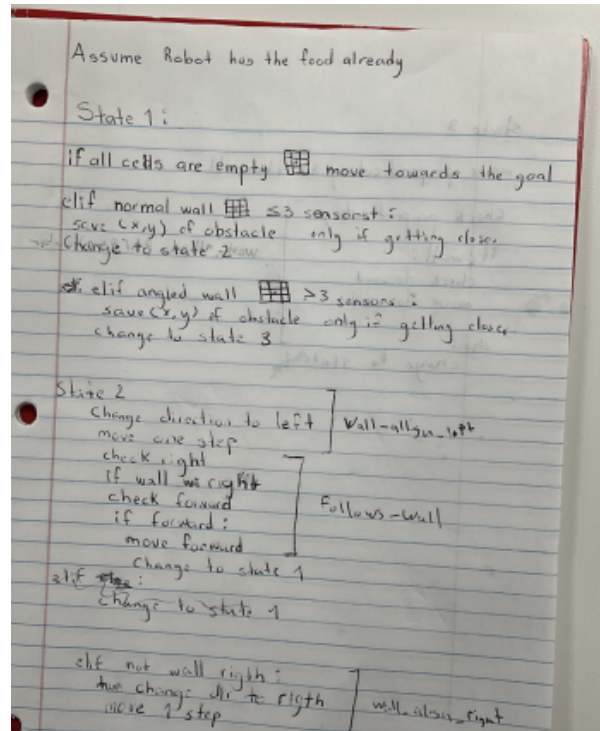


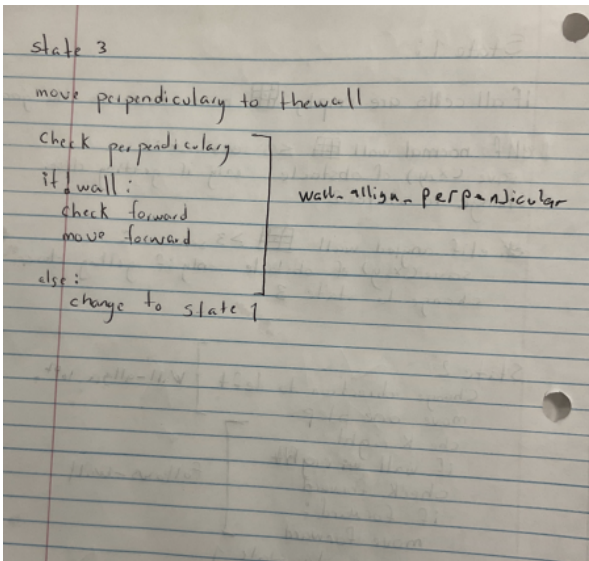
Figure 2: These are rules to escape the trap.

## 3 Conclusion

The set of rules that we created are able to allow the robot to reach its destination using a deterministic algorithm.

## References

- [ASV17] Alaa Eldin Abdelaal, Maram Sakr, and Richard Vaughan. Lost highway: A multiple-lane ant-trail algorithm to reduce congestion in large-population multi-robot systems. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 161–167, 2017.
- [Sto15] Karl Stolleis. The ant and the trap: Evolution of ant-inspired obstacle avoidance in a multi-agent robotic system. 2015.



**Figure 3:** These are rules to escape the trap.

[ZPR<sup>+</sup>13] Muhammad Zohaib, M Pasha, RA Riaz, Nadeem Javaid, Manzoor Ilahi, and RD Khan. Control strategies for mobile robot with obstacle avoidance. *arXiv preprint arXiv:1306.1144*, 2013.

# Student Data Visualizations

Elise Grizzell \*

Gaukhar Nurbek †

Nicholas Houghton ‡

## Abstract

Analyzing the UTRGV Computer Science Department’s anonymous student data is important to furthering the efficiency and effectiveness of our department and its resources. We present several previously unseen results in our work.

## 1 Introduction

A large portion of our non-plotted analysis is in the overlap between undergraduates who continued on to take master’s courses in the department. With the soon coming Ph.D. program to the Department of Computer Science, predicting whether a student will continue on is a worthwhile endeavor.

Our plotted data shows the grade and course flow data for students in the computer science department.

## 2 Methods

For this project multiple methods were used. First of all, we separated the data for CS students from the rest of the students and extracted information on how many courses each student took for consecutive semesters per year from 2016-2022. This was done by using pandas library in Python and Excel sheets. After extracting and preprocessing this information, we were able to visualize the flow of courses from Fall semesters to Spring semesters for each year starting from 2016 to 2022, using sankey plot from R highchart library. After that, we used html,CSS, js and bootstrap libraries to create summarized visualizations in a web page.

## 3 Our Results

In our data, we found several interesting correlations:

First, out of all the grad students in this department, only 83 had an in-scope major, and 45 took classes as undergrads but did not have a major in the cs fields.

Electives: The most highly correlated undergrad elective taken in the junior or senior years with continuing on with a master’s is internet programming.

Failing: Of the top 40 students by the number of classes failed, none graduated with fail counts above 10, and at 10, 4 graduated. The maximum number of classes failed by any one student is actually a tie between two students at 25 each.

Interestingly among undergrads who continued to grad school in the department, the most failed classes were systems programming (9), internet programming (8), and automata (6).

Three type of plots were created such as grades distribution for each semester per each year from 2016-2022 such as in Fig. 1. And the distribution of courses from Fall to Spring semester were shown for each year as in Fig. 2. All these plots were summarized in web page such as in Fig. 3. for each year in separate dashboards.

## 4 Conclusion

Visualization of courses and grades distribution was completed for each year from 2016-2022. To conclude we learned new tools of visualization such as Sankey diagram and were able to show distribution of students taking CS classes between Fall and Spring semester. Further investigation into the course correlation with dropping from the major is needed as well as potentially fine-grained professor-by-professor data. This is an interesting set of results.

\*Department of Computer Science, University of Texas Rio Grande Valley, [Elise.Grizzell101@utrgv.edu](mailto:Elise.Grizzell101@utrgv.edu)

†Department of Computer Science, University of Texas Rio Grande Valley, [gaukhar.nurbek01@utrgv.edu](mailto:gaukhar.nurbek01@utrgv.edu)

‡Department of Computer Science, University of Texas Rio Grande Valley, [nicholas.houghton01@utrgv.edu](mailto:nicholas.houghton01@utrgv.edu)

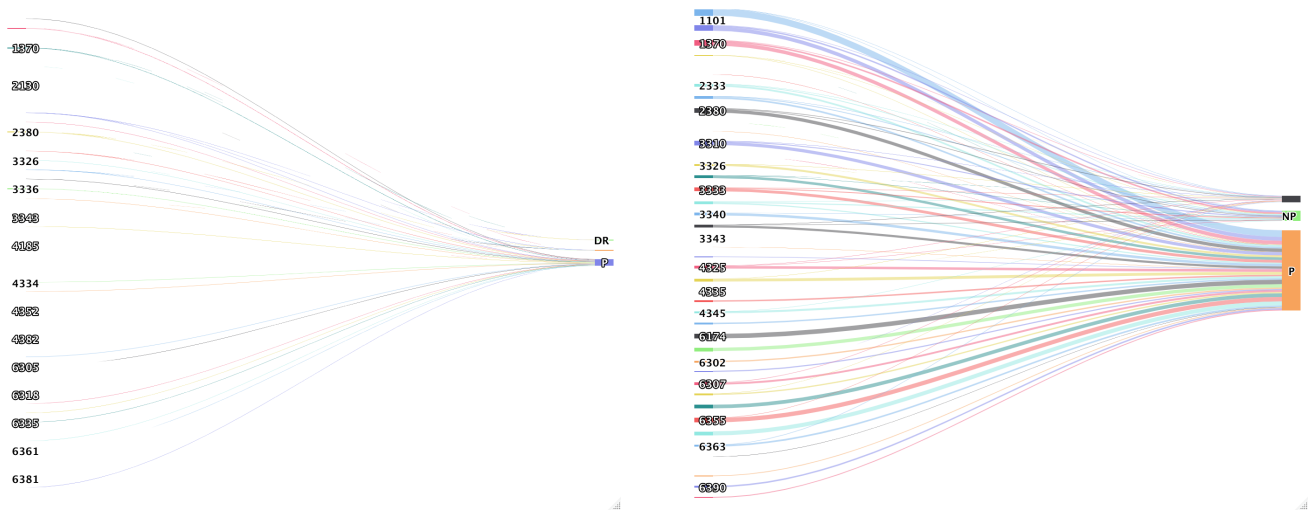


Figure 1: Spring and Fall 2016 grade statistics

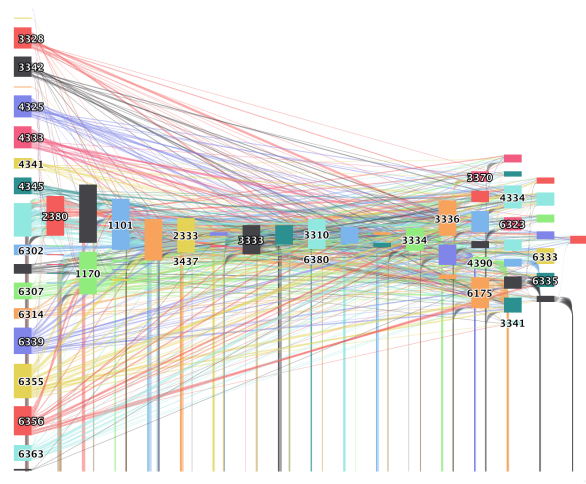


Figure 2: Courses flow for 2016

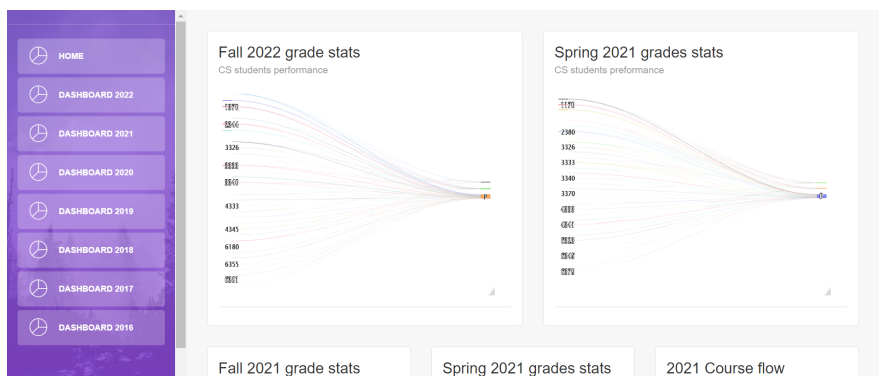


Figure 3: Our webpage

# Anime Recommendation

Kolade Alabi \*

Kofi Nketia †

## Abstract

Our objective for this project was to recommend a list of anime titles given various inputs. We considered two primary scenarios: one where user data is unavailable, and another in which it is. The recommendations were made based on efficient similarity calculations. Given the name of an anime, the "model" locates that anime and returns the top 10 most similar anime. Given a user's name, the model finds the most similar user, determines the most liked anime from both users, and then returns a list of anime exclusive to the new user.

## 1 Acknowledgement

Many of the methods and functions applied in this project were referenced from 4a previous project[1] from an Intro to Data Science course in the Spring semester of 2022

## 2 Methodology

While the primary contribution of this project is the recommendation system, the steps that we took to replace empty values in our dataset, transform categorical variables, and various other efforts to preprocess our data were equally if not more important. This section describes these efforts in detail.

### 2.1 Preprocessing: Anime Dataset

The Anime dataset presented a couple of problems for us. We first needed to deal with the empty values in the dataset. These were represented as the string "Unknown", and our first step was to replace these with proper NaN values. With that done, we opted for a simple forward-fill of values in the "Scored", "Aired", and "Duration" columns. Before filling the "Aired" and "Duration" columns, however, more select processing was necessary.

The "Aired" feature of anime titles represented the time an anime first aired to the public. These times were presented in multiple forms, such as "12/1/2005" or "December 1, 2005". By looping through each value and selecting only the last 4 characters (representing the year), this column was transformed into an integer representing how old a given anime is in years. For the "Duration" column was slightly more difficult to process, but we used a similar approach to obtain the duration of each episode in minutes.

\*Department of Computer Science, University of Texas Rio Grande Valley, kolade.alabi01@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, kofinketia01@utrgv.edu

Our final preprocessing step was to transform each categorical variable into some numeric representation. By way of encoding, we created new columns for the categorical variables of "Genres", "Source", "Type", and "Rating", placing a 0 or 1 for each row depending on whether an anime contained that category.

### 2.2 Preprocessing: Users Dataset

We also made use of a dataset of more than 300,000 users with their ratings of various anime. For the purposes of demonstrating our model, we selected the first 1000 users, considering only an anime and the user's rating of that anime. In the future, we hope to find ways to make use of the other available feature as well as the full user dataset.

A new column was created for each anime a user rated, and a value was placed for a user's rating of that anime. This was used to create an entirely numerical representation of our data that would be primed and ready for a similarity calculation.

### 2.3 Similarity Calculation

The two methods of similarity calculation used were Cosine Similarity and Euclidean Distance. Both are accurate and highly versatile measurements, but in practice, they work differently. The Cosine Similarity calculation would compare each anime to the others and give a score regardless of the magnitude of various column values. This could be useful if no normalization of data is desired as extremely large outlier values should not sway results too greatly. However, in cases where higher values should correlate more with others (like in the case of users rating an anime very high versus very low), the Euclidean Distance metric is preferred.

Using the scikit-learn library in Python, we applied both these matrices to make 2 cosine similarity matrices (one for anime and one for users) as well as 2 euclidean distance matrices (for the same groups). These matrices are created by comparing the vectors of each anime/user's measurements against all of the others. In the cosine similarity matrix, high values represent small angles between two vectors. In the euclidean distance matrix, low values represent small distances between the heads of two vectors.

```
[1.          0.70270627 0.69935987
 [0.70270627 1.          0.4162361
 [0.69935987 0.4162361  1.]
```

### 3 Our Results

Given that both sections of our recommendation system are intelligent calculations, rather than machine-learning models, accuracy metrics are not easily obtainable. However, we could obtain a broad understanding of how "well" our model performs based on previous experience watching selections of anime and a basic judging of how similar the recommendations are.

Given the name of an anime, our model finds the index of the anime in each similarity matrix and sorts the list of all other anime by their similarities. It then returns a list of names of the top 10 most similar anime.

```
recommendAnimeE("One Piece")
✓ 0.6s
['Hunter x Hunter (2011)',
 'Naruto: Shippuuden',
 'Bleach',
 'Fairy Tail',
 'Dragon Ball Super',
 'Dragon Ball Z',
 'Naruto',
 'Hunter x Hunter',
 'Black Clover',
 'Dragon Ball Kai']
```

Given the name of a user, the model finds the most similar user (using the matrix) and obtains a list of their favorite anime. It compares this list to the original user's list of favorite anime and finds exclusive entries. It returns this final list as recommendations.

```
recommendUserC("0.0")
✓ 2.3s
['Toki wo Kakeru Shoujo',
 'Yu☆Gi☆Oh! Duel Monsters',
 'Gokusen',
 'Detective Conan',
 'Tonari no Yamada-kun',
 'Hotaru no Haka',
 'Tokimeki Tonight',
 'Fate/stay night',
 'Bleach',
 'Igano Kabamaru']
```

### 4 Conclusion

Large machine-learning models are not always necessary to draw insights and make predictions, as I hope you gathered from this project.

#### References

[1] O. P. Xavier Rios, Kolade Alabi. Gamers, 2022.

# An AI for 5ive Straight

Jose Martinez \*

Brandon Tiu †

Jose Amaro ‡

## Abstract

When Straight 5 is generalized into a K-in-a-row-game, where two players place  $p$  pieces on  $M*N$  board alternatively. The first player to get his pieces K-consecutive in a line horizontally, vertically, or diagonally wins. When both players get the lowest value cards this game becomes Connect6 where there is no winning strategy from a empty board. As well at fixed constants of  $K, p$  such that  $K - p \geq \max\{3, p\}$  at any given starting position it is PSPACE-complete when trying to determine whether the first player has a winning strategy. With this when both players both get lowest value cards in their opening hand this implies PSPACE-complete[1].

## 1 Introduction

Straight 5 is a game similar to that of connect 4. The way this game is played is there is a 10 by 10 board where players take turn alternating placing pieces on the board. Another constraint is that there are cards labeled from 0 to 99 and at the start of the game, 4 cards are drawn and randomly given to each of the players. A player during their turn can either draw a card or play a card. However, a player cannot do both during the same turn. When they play a card, they can place a pin on the board where the number is either equal or greater than the value on the card.

## 2 Our Results

We were able to create a game board as well as the numbers aligned with each of the index's. We are able to designate positions to the pins on the board so that we place them in the pin holes. The game now as it stands has a bug that does not allow the game to finish in the regular condition when either a player is left with 4 dead cards or one player has 5 pins either diagonally, horizontally, or vertically.

## 3 Conclusion

Had we been able to successfully implement the board game and then code the AI to play the game, the end result would have been an AI that would a higher chance of winning against a human opponent. With the AI, the way we planned to train it was through reinforcement learning. This would have included generating a starting

population of 100,000 to 150,000 species, running it for  $n$  generations, each generations would play against another member of the species, all of the losers would be killed off, the winners would then be cloned through the use of asexual reproduction, and then we would mutate the species with a mutation rate of about ten percent to derive the AI. Once we run for a certain amount of  $N$  generations we run a tournament style once they have gained a sense for their own style in winning and through this bracket we can determine the single best winning strategy or one of the best winning strategies the Ai within those constraints could create.

## References

- [1] M. Y. Hsieh and S.-C. Tsai. On the fairness and complexity of generalized k-in-a-row games. *Theoretical Computer Science*, 385(1):88–100, 2007.

\*Department of Computer Science, University of Texas Rio Grande Valley, jose.martinez56@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, brandon.tiu01@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, jose.amaro01@utrgv.edu

## **Author Index**

Alabi, Kolade, 13  
Alaniz, Robert M., 2  
Amaro, Jose, 15  
  
Elizondo, Simon, 4  
  
Grizzell, Elise, 11  
  
Hinojosa, Hector, 1  
Houghton, Nicholas, 11  
  
Knobel, Ryan, 6  
  
Martinez, Jose, 15  
  
Nketia, Kofi, 13  
Nurbek, Gaukhar, 11  
  
Panuelos, Mike, 6  
Peredo, Angel, 9  
Perez, Juan, 6  
Phillips-Lenz, Konrad, 9  
  
Ramirez, Miguel, 1  
Rodriguez, Andrew, 2  
Ruiz, Luis, 6  
  
Tiu, Brandon, 15  
  
Valdez, Kyara, 8



