Kilter Board Beta and Route Generation

Kailen Richards

May, 2025

Abstract

Rock climbing is a sport of problem solving, especially in bouldering. The Kilter board is a standardized training board specifically for bouldering, and for many climbers, knowing where to put their hands and feet is a struggle at first: this tool will read the routes and give the optimal path for the climber to attempt. In addition, based on the given routes, the tool will generate similar routes based on a given grade that a climber selects.

Introduction

Rock climbing is a sport where understanding a route and where each part of your body needs to be is essential. In climbing terms, this is called 'beta,' and when a climber is stuck or lost on what to do on a climb, they typically ask, 'What is the beta for this climb?' and often not long after, the lost climber gains a new understanding of their sport. Unfortunately, it takes many years and practice to understand a route and read the beta for it, which presents a challenge for new climbers. I wanted to make a tool not only for new climbers but also for experienced climbers. The goal of this project is to utilize data from the kilter board to create an optimal path for each climb and, using that optimal path and the climb itself, generate new routes with their own unique sequences. I hope to use my high-level knowledge of climbing from coaching and route-setting (creating climbing routes) to create an algorithm that creates equitable sequences that are accessible and efficient.

My github repo

Related Work

Upon the initial literature search, models have been made for another standardized climbing board, the Moonboard. However, their models focus more on classifying the routes created by climbers, generating new climbs, and predicting their grades. For their move

sequencing, they utilized the standardization of the board and its holds to use the distance and difficulty of each hold to place each route in the beam search algorithm[1]. In terms of route generation, they utilize an LSTM that takes the route sequence predicted by the beam search to generate a new route[1]. For their grading system, they utilized the Moonboard's climbs with their labled grades and the generated climbs into their LSTM to grade the climb's difficulty[1].

The search found that one of the most significant issues with the grading of climbs and sequencing falls upon the fact that climbing is a male-dominated sport. Why might that be an issue? For starters, in the United States, the difference in height between biological males and females is six inches. At the same time, to the eye of the beholder, climbing makes a significant difference in the difficulty of a route. Keeping this in the scope of standardized training boards, where the boards are the same in every gym, reduces the subjectivity of the grading scales. However, in the case of the Moonboard, which was the first board to introduce numerous grade benchmarks for their boards, those benchmark grades vary for everyone.[2].

The Kilter board app allows users to see what other climbers have voted in the majority about the grade of the climb. Compared with the previous work seen with the Moon board, grades are important for marking progress but are less important when looking at new routes. Predicting the grade for a new route is inherently subjective; it varies for everyone, no matter the skill. Grades for newly generated climbs should be loosely labeled but within range of a specific difficulty; grades are merely a guide, not a certain truth.

Datasets

The dataset used was acquired from the Climb Dex Github repository which also utilizes the Board Lib Github repository, combined they allowed me to access Kilter's API to access the Sqlite3 database.

The database contains 24 tables, with the central table being the 'climbs' table, which had around 271,502 unique climbs. The 'climbs' table included 20 columns, with the central focus of the approach being on the 'frames' column, which contained a sequence of 'PxxxxRxx' pairs, each with unique identifiers subsequent to each letter. The 'P' references the id in the 'Holes' table, which contains the hold's name and corresponding X and Y coordinates. The 'R' references the table 'Placement roles' that contains the corresponding position and role of the hold in the specific climb. This vital information about the dataset allowed easy queries to create a new column for each climb to create the sequences necessary to generate new climbs.

Methodology

After gaining access to the SQLite database that contained the Kilter board data, the data needed to be transformed in a way that the modified beam search could read; through the initial data exploration, I was able to gain an understanding of the schema of the database to effectively query the database. Within the 'db utils', I used Python to query the database and parse through the frames column for each climb instance; the script creates a new column in the locally saved database titled 'holds on.' The script utilizes the 'P, R' pairs representing each hold in the climb to retrieve data from the tables that they reference. Each section of the (P, R) pairs respectfully retrieves the essential x and y coordinates of the hold and its unique 'id', and the hold role is organized into numbers (12-start, 13-middle, 14-finish, and 15-foot). After creating the 'holds on' column, each climb will be passed into the 'sequence generator.'

In order to correctly generate the sequences, each hold of each climb had to be processed and assigned the proper role. For each climb, the holds are filtered based on how they are being used: Start and finish can be used as hands or feet, All holds can be used as feet, and hands cannot use holds with the role of foot (15). The script computes the hold's distances from each other using the X and Y coordinates associated with each hold; from there, the hand and foot reachability matrix is computed. For the proper computations, precise measurement of hold distances on an actual Kilter board located at Rock Spot Climbing in Brookline, MA. Unlike the Moonboard, not every hold on the kilter is equidistant in a perfectly square grid; in some rows, there are unfilled spots. After measuring the T-nuts, which allow the climbing holds to be secured to the wooden wall along the X-axis, each hold is 8 inches from the next hold directly to the left and right of it and 8.5 inches from each hold above it. With the proper conversion to the data's X and Y plane, I measured hold reachability using another climber whose height and wingspan match the national average (5 foot 8.5 inches). I tested while he had stable positioning and how many holds he could reach directly vertically and horizontally with his arms and feet. After some testing with the sequence generator and observing many climbers, I found multiplying the X distance by 3.5 for arms and 3.2 for feet to be the most accurate representation of how a climber can reach. When generating the hold reachability matrix, I wanted to ensure that the climber cannot use a hold that is too high unless it is the start hold. From there, to place the climb on a measurable grid full of the available holds, it creates a grid based on the Y coordinates.

Additionally, each climb is passed through to generate the best sequence. To do that, I used metrics of hold quality, movement efficiency, limb alternations, body position, cross-prevention, and completion. When measuring quality, the move will be scored +10 for going to the finish, +5 for start, and +1 otherwise. Move efficiency rewards based on shorter moves by calculating the distance using dist = $\sqrt{(x_{\text{hold}} - x_{\text{prev\_hold}})^2 + (y_{\text{hold}} - y_{\text{prev\_hold}})^2}$ within the reachability matrix and computing (1/distance * 0.1). Additionally, it rewards

limb alteration by ensuring the current moved limb is not the previously moved limb. The algorithm also rewards stable body position and avoids crossing by calculating the center of gravity by averaging each limb's X and Y coordinates. It then determines if the body is within the support polygon made by the feet. To do this, it takes the range of the X coordinates of the feet and checks if the X value of the center of gravity is within that range. It rewards if it is and only slightly rewards if it is not. In order to penalize crossing, check the position of the hands in the sequence. If either hand is crossed, the move is penalized. Finally, the sequence is given a bonus for reaching the end of the climb, and each state is expanded until the end of the beam width or if an optimal sequence is found. After the modified beam search has been completed, the data is ready to be used to train the model.

The data pipeline for the model uses each move as a tuple of (hold, limb) and tokenizes them as token = (hold id * 4) + limb id + 1. The sequences are then padded to a fixed length of 50 moves. The limbs were mirrored for data augmentation; long sequences were split into shorter sequences and random jitter. As for the architecture of the model, there is the embedding layer that converts the tokens into dense vectors with a dimension of 128; then, the 2-layer LSTM captures the dependencies in the sequence with dropout. There is then a batch normalization layer before the final fully connected output layer, with a size of the vocabulary for the classification of each token. For a loss function, I used focal loss, which is designed to address significant class imbalance. For training, I used a fundamental 80/20 split for training and validation data, along with AdamW optimizer and OneCycleLR for dynamic adjustment of the learning rate, gradient clipping, and early stopping. For testing and generating new climbs, the model is saved for further use.

For the generation of new climbs, I first needed the layouts of all the walls; using the same database, I extracted the 'holds on' column and paired it with the generated sequences to create a combined JSON file to pass into the climb generator. The climb is selected randomly or by index to use the same starting holds, and then using the token encoding scheme created when training the model, the selected start holds are converted into their respective tokens. These tokens are then passed into the pre-trained model, which outputs the probabilities for the next hold/token. The temperature parameter controls the randomness of token selection, and the model uses top-p sampling to control the randomness of the output as well as to remove unlikely tokens. A token is then sampled from the distribution and appended to the sequence, which continues until the maximum length has been reached or a padding token is generated. Once the sequence is generated, it is decoded from the token back to the holds and limbs tuple, validated, and needed/missing holds are added, resulting in a complete climb.

Evaluation Results

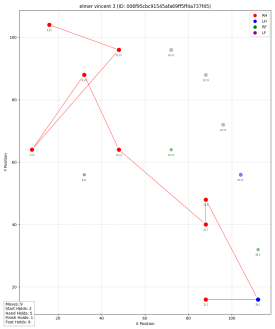

Figure 1: Figure 1

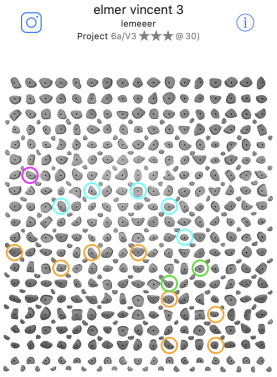

Figure 2: Figure 2

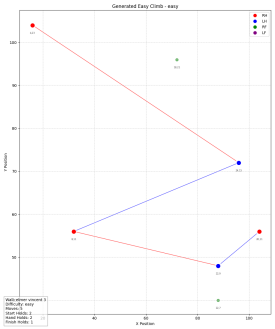

Figure 3: Figure 3

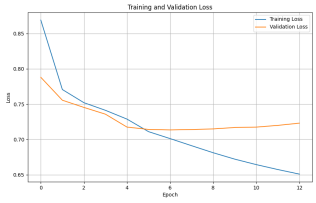

Figure 4: Figure 4

After looking at the visualized results, one can deem it successful at taking in the given climbs and sequences to produce a new climb. However, looking at Figure 1 compared to Figure 2, they show a successful sequence that finishes the climb to the untrained eye, but the sequence without the rules of the kilter board works. Unfortunately, the sequence uses holds only meant for feet as hands, but fortunately, the generated route is climbable and rather enjoyable. This error stems from the sequence generator and modified beam. It stems from how each hold is encoded and the preprocessing before the search. Figure 4 shows the training and validation loss of the model used to generate Figure 3, it performs pretty well with a validation loss below 0.75.

Conclusion

Overall, after utilizing the dataset and compiling an effective ETL pipeline to train an LSTM model, the results could be considered a minor victory, but I do not believe that this model currently lives up to its full potential. In the summer following this class, I plan to rework the sequence generator to produce more accurate sequences and retrain the LSTM to compare the results. Following the potential success of reworking the search

algorithm and producing multiple usable climbs, I plan to find a way to upload each climb onto the kilter board app to allow climbers to test and enjoy these climbs.

References

[1] Yi-Shiou Duh, Je-Rui Chang *Recurrent Neural Network for MoonBoard Climbing Route Classification and Generation.* Stanford University, 2020.

[2] LBR. *The Subjectivity of Rock Climbing Grades.* Long Beach Rising, 2024.