Data Collection

1. Data Sources and Collection Process

The dataset for the project is manually gathered using the available set of public API services the game developers provide, documentation for which is referenced here: [1].

The strategy is based on a snowballing search given a first player ID, which is used as a seed. Using an 'open' queue and a 'closed' list, we inspect all of the player's in-game characters, usually 3, and pick their latest available game. From those game IDs, we gather usage data for that match and add all participants that have not been processed before to the open list. Then, this process is repeated for as long as desired, controlled by a 'radius' parameter. We are also able to restrict the search to a date range.
Below is a diagram showing more detailed functionality of the search script.

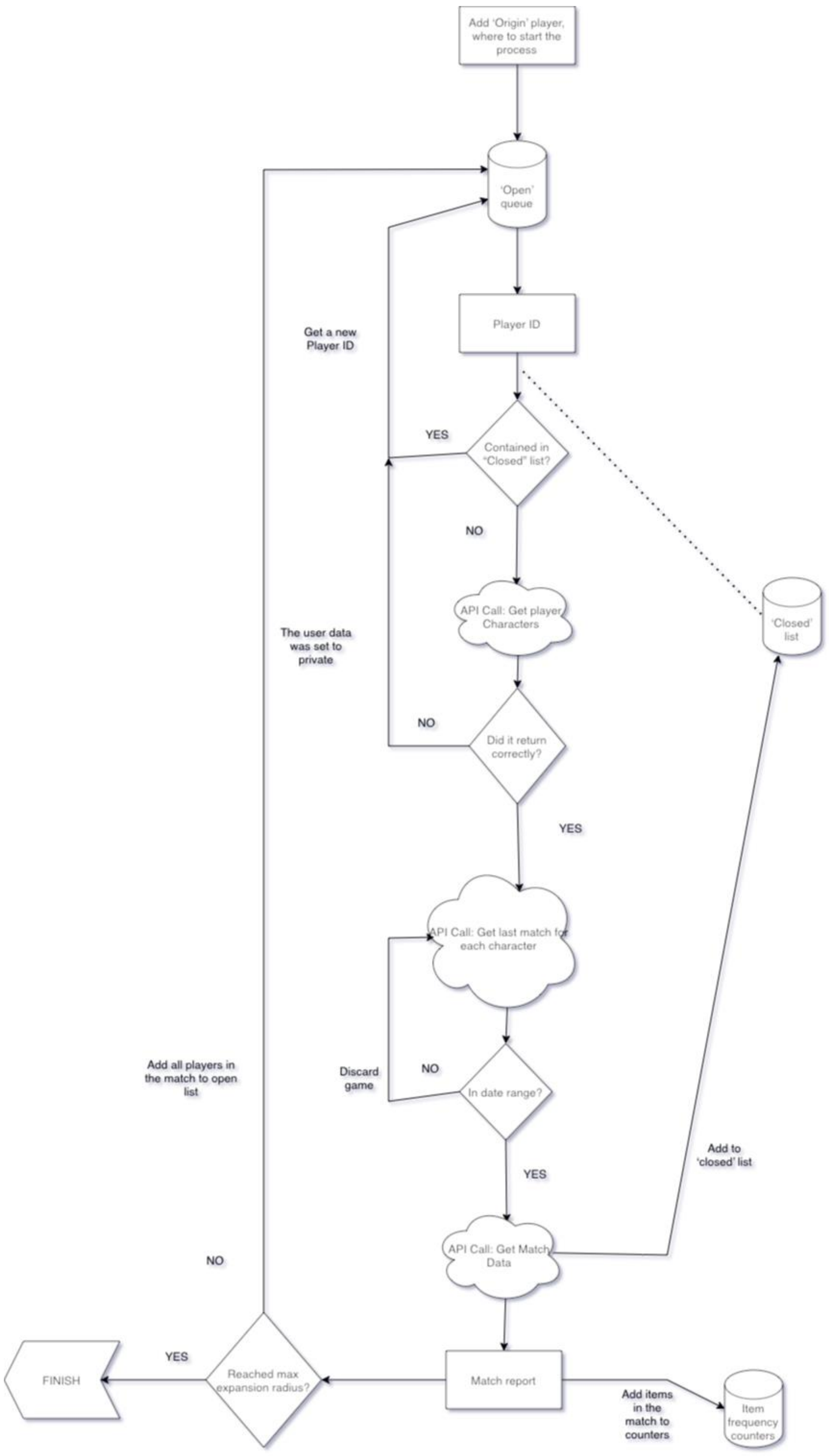
Add 'Origin' player, where to start the process
'Open' queue
Player ID
Get a new Player ID
YES
Contained in "Closed" list?
NO
API Call: Get player Characters
'Closed' list
The user data was set to private
NO
Did it return correctly?
YES
API Call: Get last match for each character
Add all players in the match to open list
Discard game
NO
In date range?
YES
Add to 'closed' list
NO
API Call: Get Match Data
YES
FINISH
Reached max expansion radius?
Match report
Add items in the match to counters
Item frequency counters

Fig 1. Flow diagram for data collection script.

To develop this gathering script, python was chosen due to its strong set of pre-existing libraries as well as ease of development. In particular, the 'json', 'requests' and 'pickle' modules were a big draw for manipulating API responses, performing queries and storing gathering results respectively.

1.1. Scalability

Unfortunately this search approach is not really scalable due to the exponentially growing set of users contained in the open list. However, the number of matches studied with a parameter of 'radius' = 2 was 1049 matches and 565 unique item instances, which is essentially the entire corpus of items contained in the game.

Studying a bigger set of matches would only improve the popularity numbers gathered, and not alter the number of instances of our dataset, therefore we believe that performance is not such a big issue, since the popularity values will converge around certain values anyway, meaning that having searches with a bigger 'radius' parameter leads to diminishing returns when it comes to having more accurate popularity metrics and runtime.

1.2. API Usage

The endpoints used in our search script do not require authentication from users but they do require, as usual, a developer key.

The API provided unfortunately constrains the performance of our search script in some ways. If we recall, three API calls were used to get Player information and their game Characters, their match history and finally the match information or 'Post Game Carnage Report' (PGCR).
The service offers some concatenation options that we make use of in the first use, getting player information and their characters in a single request, however it hinders runtime because it does not provide a way to query several players at once. The same issue appears when querying a player character's latest games, and up to three separate calls may be needed. Overall, for a single player who has three characters which is generally standard, 7 API calls are required: 1 to fetch player characters, 1 for each of the three characters to fetch their latest game and 1 for each of those latest games.

Given all this, network rate generally is a bottleneck in searches.

2. Personally identifiable information

Commenting on privacy concerns of this search strategy, no personally identifiable information is recorded or accessed at any moment in time. When obtaining player information, no emails, names or phone numbers are contained in the returned information. Only tangential information linked to a person is accessed (Username, platform), but even so it is never stored in the dataset and is only used for logging purposes.

3. Bias

Since the search strategy relies on picking at maximum 3 games from a given user, all from different player characters we essentially get a random match from users. This is important because users tend to 'stick to their guns' and use their preferred items more than others. If the search strategy used more than this limited set of matches, our ‘origin’ block could be biasing the dataset towards the weapons the origin user was using in that extended set of games.

Finally, following up on point 1.1, since the dataset does not contain users and instead consists of items, no human bias is introduced.

Item bias is however introduced, but this is desirable since we are measuring popularity, and in this case having the set of data biased towards a certain item acts as a proxy for the global popularity of the item.