Finding Important Features on the Wikispeedia Game - Final Report

Riccardo De Vidi[1], **Michela Schibuola**[2], **Lorenzo Serafini**[3]

Department of Information Engineering, University of Padua
Via Gradenigo 6, 35131 Padova, Italy
{riccardo.devidi.1[1], michela.schibuola[2], lorenzo.serafini.1[3]}@studenti.unipd.it

Introduction

Wikispeedia [1] is an online game played on a 4,600-article version of Wikipedia, that was once available at schools-wikipedia.org. Each article is classified under certain categories based on its content and is connected to the others with links. Given two articles, the player's goal is to start from the first one and reach the second by following the links between them. The game has been developed from the Data Science Lab, a research group in the School of Computer and Communication Sciences at EPFL in Lausanne, Switzerland [2].

Data information

SNAP Stanford [3] provides the Wikispeedia data [4][5].

- `articles.tsv`: list of 4,604 articles;
- `categories.tsv`: list of articles in which, for each of them, its category is specified;
- `links.tsv`: list of 119,882 links, represented as articles pairs;
- `paths_finished.tsv`: list of 51,318 players' finished paths, represented as lists of articles;
- `paths_unfinished.tsv`: list of 24,875 players' unfinished paths, represented as lists of articles;
- `shortest-path-distance-matrix.txt`: matrix in which each row represents an article and it contains the distances from it to all articles, an underscore "_" is used to indicate that the target cannot be reached from the source.

The dataset also contains the plain text content of each article and the full HTML package of the Wikipedia version used by Wikispeedia.

Paths formulation

`paths_finished.tsv` and `path_unfinished.tsv` contain lists of paths done by users who played the game. Each row contains information about the user and the time of execution of the game session, which will not be used in our analysis. Each user's path is represented as a list of articles separated by a semicolon. The user has the ability to come back to the previous article by "back click"; this information is represented with a "<" in the path. In our analysis, we will consider only the effective paths, by removing the articles wrongly visited. At the end of the line there is a rating given by the user, which will not be used too.

Categories formulation

The categories are characterized by a hierarchical structure, many articles have more than one category, instead some articles have no category. The categories structure is not given. The hierarchy is given by a list of categories starting from the root to a subcategory, for example the `10th_century` article has the category `subject.History.General_History`, and the `Almond` article appears twice in the list because it has the categories `subject.Everyday_life.Food_and_agriculture` and `subject.Science.Biology.Plants`.

Objective of Study

The structure of articles and links over which the game is played can be seen as a directed graph, where each article is a node, and each link between two articles is an edge. From `shortest-path-distance-matrix.txt` we know that the longest shortest path has length 9, and so the diameter of the graph is 9.

To begin, we check if it is possible to reach each article from every other article and if the graph can be intended as undirected (i.e. if there exists a link that from a first article brings to a second article, then there exists a link that from the second article brings to the first one).

Continuing, we compare the paths followed by the players with the optimal ones and analyze them in relation to random generated paths. We determine which articles occur the most in the shortest paths, which may be the article that would help with the resolution of most of the games. Moreover, we compare the lengths of the shortest paths in the graph with the ones of random generated graphs to understand if they are a unique characteristic of the game graph or not chosen a random graph generation method.

To conclude, using the given classification of the articles, we compare this grouping with the ones that can be performed by different clustering algorithms. This will help to understand whether the connection between the articles is given by their categories or by other aspects.

Analysis

Connectivity and undirected graph

To verify whether each article can be reached by any other article, it is necessary to check whether the directed graph is strongly connected. From `shortest-path-distance-matrix.txt` we find that some entries are labeled with "_": this means that some articles cannot be reached by some others.

Additionally, instead of checking if the graph can be considered undirected, it is simpler to check if it is true the contrary: if the list of links contains a pair of articles but not the same pair swapped, the graph cannot be considered undirected. It is immediate to verify that in the dataset there is the edge `(10th_century, Algeria)` but not the edge `(Algeria, 10th_century)`, while for example there are the edges `(10th_century, 11th_century)` and `(11th_century, 10th_century)`, which means that the graph cannot be considered undirected.

Player paths analysis

Given a user's path, in all the shortest paths with the same source and target, we count

- the number of common edges with respect to the user's path;
- the number of edges that the user used before performing an error with respect to the shortest path;
- the number of edges that the user used to reach the target article after having performed his last error with respect to the shortest path,

and we keep the highest value among all the shortest paths for all the metrics. Moreover, for each player's path we compute the length difference between the performed path and the optimal one.

Central articles usage by players

We compute the betweenness centrality of each node and count how many times the 100 nodes with the highest betweenness centrality are used by the users in their paths.

Shortest paths lengths distribution

We verify whether the out-degrees and the in-degrees of the graph nodes explain the distribution of the distances between all the articles.

Comparison with random paths

To analyze the users' paths, we compared them with paths generated randomly from the same starting node to the same destination node. The main goal was to determine, under different parameters, whether there are situations where the pathway is easy to find (i.e., it is frequently identified by the random paths) or not. Additionally, this approach provides insights into the probability of finding a path with a short length.

Clustering

We compared two greedy methods: Clauset-Newman-Moore greedy algorithm and the Louvain community detection algorithm. We wanted to understand if it was possible to infer the category of the articles by performing clustering on the graph and consider the articles in the same cluster to have the same category. All but six articles have categories defined in a hierarchical structure. All the categorized articles have "`subject`" as their first category in the structure so we focused on the successive ones. Unfortunately some articles have only one category after "`subject`" like the article "`Algebra`" that has categories: "`subject.Mathematics`". Due to this fact we focused only on the very first category after "`subject`".

Experiments

Architecture for the experiments

All the experiments have been performed on consumer laptops, e.g. HP15s (CPU: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.20 GHz; RAM 16GB; Windows 11).

Repository

The link to the code of our repository is
https://github.com/michelaschibuola/wikispeedia.git .

Player paths analysis

Each player's path starts from a source node, terminates on a final node and has a target node that is equal to the final node in the case of finished paths and different in the case of unfinished paths. We store all the ordered couples composed by source node and target node in a set, and we compute the shortest path between source and target for all the couples with the function `networkx.shortest_paths_generator()` [6]. The result obtained for a couple will be used to analyze all the players' paths with the same start and target nodes. Continuing, for each player's path we compute the previously reported metrics iterating over the optimal paths and the player ones, we store them in a dataset (with one row of metrics for each player path) and we study their relationship exploiting the object `seaborn.PairGrid` [7].

Central articles usage by players

We compute the betweenness centrality for all the nodes with the function `nx_parallel.betweenness_centrality()` [8], we count how many times the 100 nodes with the highest betweenness centrality are used by the users in their paths, and we represent the obtained information with a bar plot to see how they compare (`matplotlib.pyplot.hist()` [9]).

Shortest paths lengths distribution

We generate 50 random graphs exploiting the function `networkx.directed_configuration_model()` and we compute the distribution of the distances between all the nodes in each of them. We then exploit the mean

and the standard deviation of each distance (i.e. we estimate the mean and the standard deviation of 50 values for each distance) to estimate the z-score.

Comparison with random paths

Before executing the random walks, we analyzed the distribution of the path lengths for the completed paths. Based on this analysis, we implemented the random walk algorithm by starting from the source node and moving to one of its neighbors, selected randomly, until reaching the destination node or exceeding a maximum number of steps. For each path, the random walk was executed a fixed number of times.

We applied this technique first to a fixed number of random paths selected from the completed paths, then to the paths where users needed more than 50 steps to reach the destination, and finally to a fixed number of random paths selected from the unfinished paths. At the end, we compared the lengths of the users' solutions with those of the random walk solutions to evaluate the quality of the users' answers.

Clustering

The Clauset-Newman-Moore greedy algorithm and the Louvain community detection algorithm were implemented by the NetworkX functions: `greedy_modularity_communities` and `louvain_communities`. A first issue, as mentioned, is that there are six articles that have no category: `'Directdebit'`, `'Donation'`, `'Friend_Directdebit'`,`'Pikachu'`, `'Sponsorship_Directdebit'`, `'Wowpurchase'`. An easy fix for this is to define an addition category: `'NoTag'` which all of these six articles will be consider part of. The second issue is that some articles have more than one category, since they are reported more than once in the file `categories.tsv`.To analyze the clustering reported in output by these two methods, we wanted to understand for each cluster the number of articles that belong to each category. We decided that an article with multiple categories would be counted in each of its categories.

Results

Player paths analysis

The relationships between the metrics are reported in Figure 1 as histograms. The diagonal of the grid contains the densities of the metrics evaluated on finished and unfinished paths, the lower triangular map contains the relationships between the metrics evaluated on finished paths, and the upper triangular map contains the relationships between the metrics evaluated on unfinished paths.

Looking at the diagonal elements of the grid, we notice that the distribution of the differences between the players' and the optimal paths can be seen as invariant from the conclusion of the game, while all the other metrics have marginals that are not similar.

It is interesting to note that the optimal length of finished paths has a mean which is smaller than the one of unfinished ones, and that the player length distribution of the unfinished paths has its maximum in 0. From this information, we can conclude that in general the players prefer to play simpler games and that when the game seems harder to play, the users tend to quit it immediately.

The histogram `Beg equal edges` vs `End equal edges`, `Finished = True` in a first analysis seems symmetric. We consider this result interesting, since at first glance in our opinion there is no correlation between these two metrics, but we do not investigate it further. We limit ourselves to observe that this symmetry is reflected in all histograms with `Beg equal edges` and `End equal edges` , `Complete = True` .

Also the histogram `Player length` vs `Difference`, `Finished = True` shows an interesting correlation between the metrics: the length difference seems to be distributed as a linear function of the player path length plus some noise. An equivalent characterization can be given to the histogram `Player length` vs `Difference`, `Finished = False`. Other histograms whose invariance from the hue is interesting are `Optimal length` vs `Player length` and `Optimal length` vs `Difference`.

To conclude, it is possible to study how optimal in terms of equal edges and length are the finished players paths analyzing the histograms `Player length` vs `Optimal length`, `Finished = True` and `Player length` vs `Tot equal edges`, `Finished = True`. From the first one, we observe that there is not a direct correlation between the optimal length and the player one, but we notice that for the majority of games players tend to perform zero to two errors, and that an important part of games that could be finished in one step takes much longer. The latter suggests that players do not scan the entire Wikipedia page before clicking on a link. From the second one, we observe that the players have never taken more than six correct links in a path and that the paths of length three, in general, have more correct edges than the longer ones.

Central articles usage by players

Looking at the histogram reported in Figure 2 we observe that the 100 nodes with the highest betweenness centrality are often used by the users more frequently with respect to the optimal paths. It can be observed, applying the previous analysis to the whole set of articles, that almost all the articles are used more frequently by the users with respect to the optimal paths. Restricting our analysis to the finished paths values, it is observed that the nodes with higher betwenness centrality tend to be more used by the players with respect to the other graph nodes.

Shortest paths lengths distribution

From the scores reported in Table 1 we can conclude that the outdegrees and indegrees of the graph nodes do not explain the distribution of the distances between all the articles since between f and $E[X_F]$ there is an error bigger than three standard deviations for all the distances.

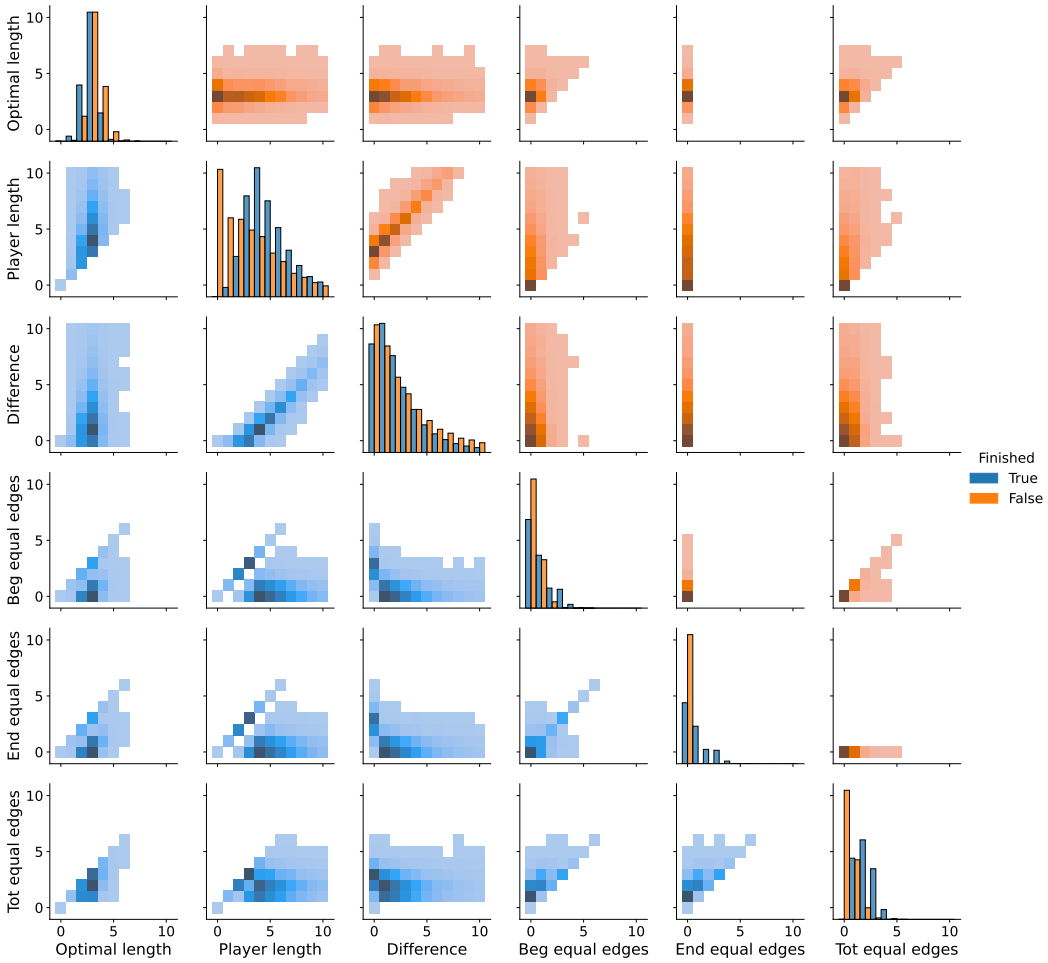

Figure 1: Correlations between paths metrics.

Length	z-score	f	$E[X_F]$	$\sigma[X_F]$
0	10.3209	$2.4762e-04$	$2.4424e-04$	$3.2713e-07$
1	32.6737	$6.4418e-03$	$6.1583e-03$	$8.6765e-06$
2	-29.7866	$1.6016e-01$	$1.8307e-01$	$7.6933e-04$
3	-34.4554	$5.2038e-01$	$6.0720e-01$	$2.5197e-03$
4	31.1411	$2.5930e-01$	$1.8736e-01$	$2.3101e-03$
5	25.5734	$4.4991e-02$	$1.4973e-02$	$1.1738e-03$
6	18.4228	$7.6480e-03$	$9.2249e-04$	$3.6506e-04$
7	8.5532	$7.8885e-04$	$5.0469e-05$	$8.6327e-05$
8	3.2886	$2.5170e-05$	$1.3478e-06$	$7.2440e-06$
9	-	$2.6892e-07$	0	0
≥ 10	-	0	0	0

Table 1: Lengths distribution z-scores; f Wikispeedia lengths density, $E[X_F]$ random graph distances probability mean, $\sigma[X_F]$ random graph distances probability standard deviation.

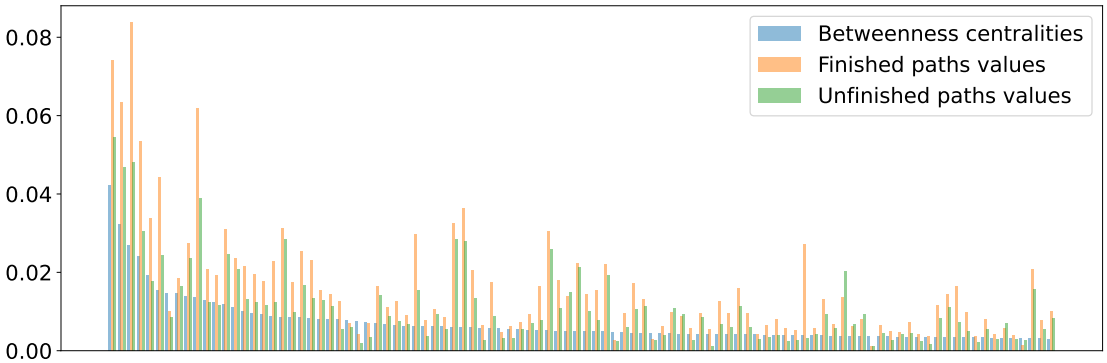

Figure 2: Betweenness centralities and articles usage in players paths; each set of three bars is associated to an article, the values are reported as percentages.

Comparison with random paths

The analysis of users' paths revealed that the average path length is approximately 7 steps, while the maximum length reaches 435 steps. We observed that paths with longer lengths were significantly fewer than shorter paths. Therefore, we decided to generate random paths with shorter lengths to ensure the execution time was not too high and to obtain a limited number of results that could provide more valuable information.

For the completed users' paths, we found that in some cases, the random walk produced shorter paths than the users' paths. In other cases, it failed to find any path between the source and destination nodes, and sometimes it generated longer paths. This behavior is shown in Figure 3. For paths longer than 50 steps, we ran the algorithm multiple times. We noticed that, for some games, the random paths produced a large number of short solutions. This suggests that users did not find an optimal solution, even though the probability of finding one was relatively high. However, for some paths, the random walk failed to find any solution. This behavior is shown in Figure 5.

For the random unfinished paths, as shown in Figure 4, the behavior was similar to that of the finished paths, indicating that the difficulty of the game was comparable for both types. To confirm this hypothesis, we computed whether the random walk could find at least one path of length less than 20 within a maximum of 200 trials for both finished and unfinished paths. The results showed that, for finished paths, in 45% of the cases (23,256/51,318), the random walk could not reach the destination node. For unfinished paths, this percentage increased to 66% (16,463/24,875).

In both cases, the random walks struggled to reach the destination node, but the results indicate that it was more challenging for unfinished paths.

Clustering

Even though there are articles with multiple categories let's note that: only 1 article has 3 categories, 505 articles have 2 categories and 4098 articles have 1 category. Moreover by inspection of the cumulative plots, that we report for both the algorithms in the "`images`" folder of our repository, we can see that for large clusters almost all the categories are present, i.e., a category wasn't significantly present in one of the clusters but was scattered among them. This means that the two algorithms weren't unable to categorize the articles only by the modularity optimization. (Greedy plot pt1: Figure 6). Another aspect to point out is that both of the methods, kept the isolated nodes in their own cluster, since they do not contribute to an increase in the modularity. In fact in the plots for Louvian clustering, the Louvian clusters from 11 to 22 are the same as the greedy clusters 13 to 24. Also they both reported the connected component of size 3: cluster 10 for Louvian and 12 for greedy (Greedy plot pt2: Figure7): '`Directdebit`', '`Friend_Directdebit`', '`Sponsorship_Directdebit`'.

References

[1] Robert West. *Wikispeedia*. https : / / dlab . epfl . ch / wikispeedia/play/. -.

[2] EPFL. *dlab*. https://dlab.epfl.ch/. -.

[3] Standford University. *Wikispeedia navigation paths*. https://snap.stanford.edu/data/wikispeedia.html. -.

[4] Robert West, Joelle Pineau, and Doina Precup. "Wikispeedia: An Online Game for Inferring Semantic Distances between Concepts". In: *21st International Joint Conference on Artificial Intelligence (IJCAI)* (2009). URL: %5Curl%7Bhttp://infolab.stanford.edu/~west1/pubs/West-Pineau-Precup_IJCAI-09.pdf%7D.

[5] Robert West and Jure Leskovec. "Human Wayfinding in Information Networks". In: *21st International World Wide Web Conference (WWW)* (2012). URL: %5Curl%7Bhttp : / / infolab . stanford . edu / ~west1 / pubs / West - Leskovec_WWW-12.pdf%7D.

[6] NetworkX developers. *NetworkX Network Analysis in Python*. https://networkx.org/. 2014-2024.

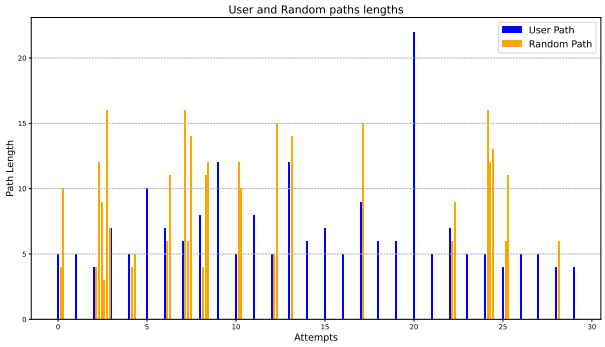

Figure 3: Comparison between some users' paths length (blue) from the finished paths with random generated paths length (yellow). Random paths generated with maximum 15 steps in 200 attempts.

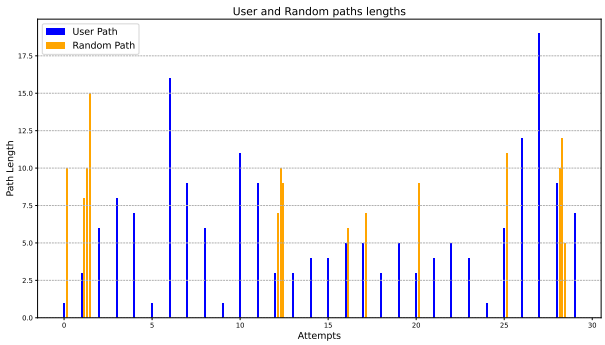

Figure 4: Comparison between some users' paths length (blue) from the unfinished paths with random generated paths length (yellow). Random paths generated with maximum 15 steps in 200 attempts.

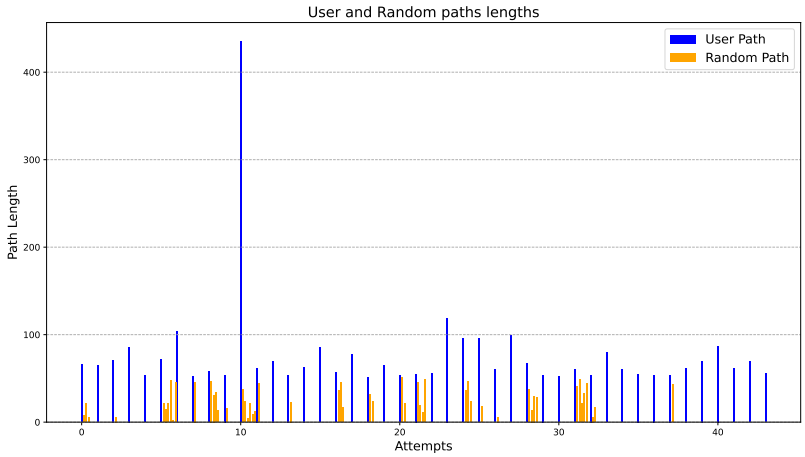

Figure 5: Comparison between the users' paths length (blue) from the finished paths in which they needed more than 50 steps with random generated paths length (yellow). Random paths generated with maximum 50 steps in 200 attempts.

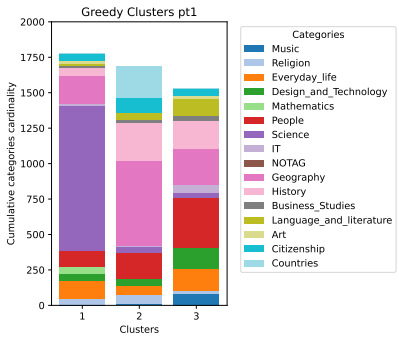

Figure 6: Cumulative categories cardinality in each greedy cluster pt1

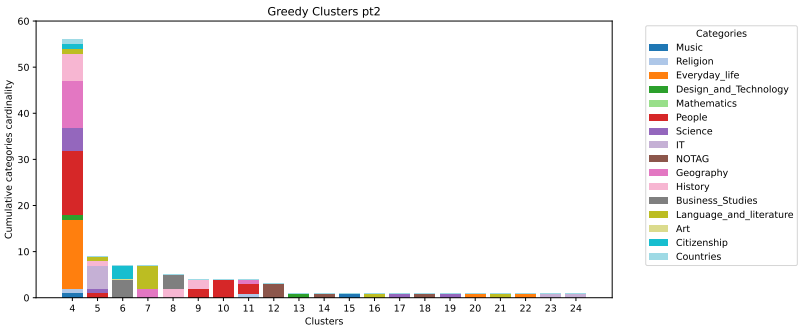

Figure 7: Cumulative categories cardinality in each greedy cluster pt2

[7] Seaborn developers. *Seaborn: statistical data visualization*. https://seaborn.pydata.org/. 2024.
[8] NetworkX developers. *nx_parallel*. https://github.com/networkx/nx-parallel. -.
[9] Matplotlib developers. *Matplotlib: Visualization with Python*. https://matplotlib.org/. 2024.

Contributions

During the process of searching the various datasets for an interesting project all the three of us proposed some ideas. We ended up selecting the project proposed by Lorenzo Serafini, that seemed to be the most engaging. Initially, Michela Schibuola stood out for analyzing the graph, finding out if it was connected and directed or undirected. Riccardo De Vidi focused on the analysis of the dataset concerning the paths done by the users. Lorenzo Serafini was concentrated on using also the category of the articles to analyze the graph, so he leaned toward clustering. Thus it was clear for us that we would divide the final work in the following way:

Riccardo De Vidi - 33.3%

Player paths analysis, central article usage by players, shortest paths length distribution.

Michela Schibuola - 33.3%

Connectivity and undirected graph, central article usage by players, random paths.

Lorenzo Serafini- 33.3%

Project idea, clustering.