Precog Recruitment Task
A-T-L-A-S ATLAS!

Aryan Maskara

February 5, 2025

For simplicity, the player starting the game is Player 1, and the other player is Player 2.

1 Tasks Attempted

The following tasks were attempted:

- Task 1 - Graph Analysis
- Task 2 - Community Detection
- Paper Reading

The bonus task (Link Prediction) was not attempted because I did not have enough time to explore the Node2Vec and PyTorch libraries and learn how to implement the specifics necessary for link prediction. While reviewing the research paper provided for reading helped me understand the Node2Vec details, I still did not have sufficient time to fully grasp and complete the requirement.

2 Creating the Datasets

The implementation for this part has been done in `scrape_dataset.py`

For me, the official list of countries is the same as the **member nations of the United Nations**. The countries dataset has been taken from here, since this dataset uses country names that are more **commonly known to the general public**, which will be used while playing ATLAS.

The cities list has been taken from here. Only the top 500 cities have been taken.

The countries + cities list is a combination of the first 2 datasets.

The data was saved to respective CSV files after scraping the websites for the names of the countries and cities.

3 Creating the Graphs

3.1 Original Graphs

The graphs have been created using the NetworkX library, as mentioned in the task document. If the **last letter of country A was same as the first letter of country B**, then a directed edge from A to B was created. This was implemented in `create_original_graphs.py`.

3.2 Simplified Graphs

Since while saying a country, the player goes from the **first letter to the last letter**, the player does not need to have links (edges) between pairs of nodes. Instead, they can have links between a pair of characters. For example, for Romania, an edge from R to A will be added. Furthermore, since there may be many countries starting and ending with the same pair of letters, **each edge will be weighted**, where the weight denotes the number of countries / cities the player can say to go from one letter to another.

Thus, the simplified graphs have only 26 nodes and multiple edges, depending on the dataset. This was implemented in `create_simplified_graphs.py`.

4 Task 1 - Graph Analysis

4.1 Indegrees and Outdegrees

The first thought one could have is to see where the game could come to an end. Only nodes with **indegree count** $>$ **outdegree count** can serve as an end to the game. This is because if the indegree count $\leq$ outdegree count is satisfied, there can be a one-one or a one-many mapping which allows the player to escape that letter. The idea of this insight is to potentially develop a good strategy by having information on where to trap your opponent.

4.2 Closed Subgraphs of $\leq$ 3 nodes

4.2.1 Countries Only Graph

Running `graph-analysis.py` for closed subgraphs, we see that for countries only graph, **Player 1 can trap Player 2 in the letter A**.

Further analysis shows that **being trapped in he letters O, A and R is of actual importance**. Player 1 can say San Marino to trap Player 2. Player 2 says Oman, while Player 1 says Niger, keeping him in the subgraph. Player 2 can only say Russia / Romania / Rwanda, getting to A. There are 11 countries in A, 9 of which end in A and 2 end in N (Afghanistan and Azerbaijan). If Player 2 tries to go to N, Player 1 can stop them by saying Namibia, Nicaragua, North Korea or Nigeria, bringing them back to A. In the end, when it is Player 2's turn, they will run out of countries and Player 1 wins.

4.2.2 Cities Only Graph

Running `graph-analysis.py` for closed subgraphs, we see that for the cities only graph, **Player 1 can trap Player 2 in the letter I by saying Siliguri**. Now whatever Player 2 says, Player 1 can trap them back at **letter I**.

- Istanbul meets with Lubumashi
- Incheon / Ibadan meet with Nairobi / New Taipei
- Indore meets with Ekurhuleni
- Izmir meets with Rawalpindi
- Islamabad meets with Delhi

There are no other cities starting with I, hence Player 2 is stuck at I, and Player 1 wins.

4.2.3 Countries + Cities Graph

Running `graph-analysis.py` for closed subgraphs, we see that for the countries + cities graph, **Player 1 can only trap Player 2 in the letters A, I, U**. Analysis on this would be difficult however, since there may be some countries or cities that do not get used in the subgraph, hence changing the outcome of the game. For example, if Player 1 was at I, and Uganda had not been used, Player 1 lost one word that they could have said, which could mean that instead of Player 1, Player 2 wins this game.

4.3 Reachability Matrix

The reachability matrix is a heatmap that denotes the shortest path between any pair of nodes. This helps visualise some forceful paths, if they exist, in the 3 graphs created. For example, in the **countries graph**, via the heatmap, one can visualize that if **Player 1 lands at Q, they have to say a country that ends with R.** Similarly, if Player 1 lands at R, they have to say a country that ends with A.

Although there are no forceful paths in **cities graph**, there are some nodes in which the **options to go to other nodes become very limited.** For example, from Y the player can only go to E, G, I, N and U. On the other hand, there are some nodes from which you have a lot of options. For example, from S the player can go to A, E, G, H, I, L, M, N, O, R, T, U, Y and Z.

Since more edges get added to the **cities and countries graph**, it is a very dense graph, where all the pair of nodes **either being separated by a distance of 4**, or being unreachable from one another.

4.4 Correlation between Degree Centrality and Winning

Intuitively, it makes sense to always land on the node which is the most connected, so that you can maybe trap the other player into a cycle or into a tough situation.

I simulated 1,000 random games to see whether there is a correlation between **degree centrality** and **winning the game**. Let **winning centrality** be the average of the degree centralities of the person winning the game, while **losing centrality** is the average of the degree centralities of the person losing the game.

I plotted **winning centrality − losing centrality** when Player 1 wins and when Player 2 wins. I also found the correlation between the difference in centrality and the results, which came out to be close to 0 (**-0.01 to -0.11** to be exact). Thus, although we intuitively think that higher centrality equates to winning, that is not the case, at least when the game is played completely randomly.

4.5 Frequency of Ending Nodes

From Section 4.1 we know the possible nodes where a player can lose. Obviously not all of these nodes are equally likely to be the nodes where a player loses, but is there something else that is hidden in it - a fact that although theoretically we consider a node to be losing but practically there are minimal instances of the losing player ending on the node?

In this insight, I plan to simulate 10,000 games and plot a graph on the node due to which the losing player lost the game.

4.5.1 Countries Only

In the countries only plot, **more than 3 games out of every 4 end on the letter A**. The rest of the distribution is mainly distributed among the letters O, Y and R. Thus, although there are 8 letters because of which a player can lose, the root cause of it is usually the letter A, followed by O, Y and R. This may suggest that if one plays cautiously, they can win by trapping their opponent in the letter A.

4.5.2 Cities Only

In the cities only plot, **more than half the games end on the letter E**. Furthermore, the rest of the distribution is mainly distributed on I, U and O, with the graph showing a tiny spike for R. This actually gives an insight on cities − **a lot of the cities have their ending letter as a E, I, O or U, but very few cities actually start from these letters.** This is also confirmed by looking at the indegrees and outdegrees from Section 4.1.

4.5.3 Countries + Cities

In the countries + cities plot, **about 4000 games end on E, while 3500 end on O**, which comprises of approximately 3 out of every 4 games. After that, we see a huge drop in the frequency distribution among Y, I and R. Furthermore, a tiny portion of the games end at N and U.

One would think that there would be some minor differences in the frequency at which the losing player ends at the nodes, but the graphs depict that the losing nodes are heavily

categorized into 2-3 groups, which may give way to varying strategies that the players would want to try out in a game

4.6 K-core decomposition

The K-core decomposition of a graph is a maximal subgraph where every vertex is connected to at least K other vertices (or, has a degree of at least K). Denser graphs are more interconnected, and hence the nodes should have a higher K-core value.

Since this is a two-player win/loss game (where one player is trying to maximise the outcome, while the other is trying to minimise it), it gives ideas for complex strategies that break through the interconnectivity of the graph multiple times, while ensuring that the player is safe.

For the **countries only** graph, most of the letters have a **k-core value of 7**, which shows that although the graph is connected, it is much simpler to understand and be used in a way that benefits the player.

The idea is confirmed by denser graphs that have more edges. Most of the letters in the **cities only** graph have a **k-core value 12**, a significant increase. This is because the number of edges in this graph has also increased by 2.5 times when compared to the countries-only graph.

The **countries and cities** graph consists of about 100 more edges than the cities-only graph, which can be shown by the **maximum k-core value increasing to 15**.
Thus, for bigger graphs (with more edges), the strategy needs to become more resilient so as to be able to break into the k-core values of the nodes of the graph multiple times, while ensuring that the player can still reach to safety.

5 Task 2 - Community Detection

In this task, I implemented `Louvain` and `Infomap` algorithms to find clusters and communities of the Atlas graph created by countries of the world.

I assessed the quality of the community detection algorithms with the following metrics. All the metrics have been taken from the documentation of NetworkX library.

- Modularity
- Coverage
- Conductance

5.1 Louvain Algorithm

The Louvain algorithm maximized modularity by maximizing edges within communities and minimizing edges in between communities.

5.1.1 Results

The results of the Louvain algorithm are as follows:

- **Louvain Modularity:** 0.453
- **Louvain Coverage:** 0.711
- **Louvain Conductance:** 0.146

These results suggest that the community structure detected by the Louvain algorithm indicates a clear separation of different communities.

5.1.2 Communities

The detected communities are as follows:

- **Community 1:** India, Indonesia, Bangladesh, DR Congo, Thailand, Italy, Iraq, Poland, Mali, Malawi, Chad, Burundi, Haiti, Dominican Republic, Hungary, Israel, Denmark, Finland, Ireland, Djibouti, Fiji, Brunei, Iceland, Kiribati, Dominica
- **Community 2:** China, Brazil, Russia, Tanzania, Kenya, Colombia, Uganda, Algeria, Argentina, Afghanistan, Canada, Angola, Malaysia, Ghana, Venezuela, Australia, Zambia, Romania, Guatemala, Cambodia, Guinea, Rwanda, Bolivia, Tunisia, Cuba, Papua New Guinea, Portugal, Azerbaijan, Austria, Libya, Bulgaria, Liberia, Mauritania, Costa Rica, Panama, Croatia, Georgia, Mongolia, Bosnia and Herzegovina, Qatar, Moldova, Armenia, Lithuania, Jamaica, Albania, Gambia, Botswana, Latvia, Guyana, Luxembourg, Malta, Micronesia, Grenada, Tonga, Antigua and Barbuda, Andorra
- **Community 3:** United States, Philippines, South Africa, South Korea, Sudan, Spain, Saudi Arabia, Syria, Sri Lanka, Somalia, Senegal, Netherlands, South Sudan, United Arab Emirates, Honduras, Sweden, Belarus, Switzerland, Sierra Leone, Laos, Serbia, Singapore, Slovakia, Slovenia, Cyprus, Mauritius, Comoros, Solomon Islands, Suriname, Maldives, Bahamas, Barbados, Sao Tome & Principe, Samoa, Saint Lucia, Seychelles, St. Vincent & Grenadines, Saint Kitts & Nevis, Marshall Islands, San Marino
- **Community 4:** Pakistan, Nigeria, Japan, Iran, Germany, Yemen, Uzbekistan, Nepal, Cameroon, Niger, North Korea, Burkina Faso, Kazakhstan, Benin, Jordan, Tajikistan, Turkmenistan, Kyrgyzstan, Paraguay, Nicaragua, Lebanon, Norway, Oman, New Zealand, Namibia, Gabon, Lesotho, North Macedonia, Bahrain, Bhutan, Liechtenstein, Nauru
- **Community 5:** Ethiopia, Egypt, Turkey, France, Ukraine, Mozambique, Côte d'Ivoire, Chile, Ecuador, Zimbabwe, Czech Republic, Greece, Togo, El Salvador, Congo, Central African Republic, Kuwait, Eritrea, Equatorial Guinea, Trinidad and Tobago, Timor-Leste, Estonia, Eswatini, Cabo Verde, Belize, Tuvalu
- **Community 6:** Mexico, Vietnam, United Kingdom, Myanmar, Morocco, Peru, Madagascar, Belgium, Uruguay, Guinea-Bissau, Montenegro, Vanuatu, Monaco, Palau

On closer look, most of these communities are related by only 1-2 letters. For example **Community 2** mainly has countries that either start with A or end with A. Similarly, **Community 3** has countries that either start with S or end with S. **Community 4** follows the same suit with the letters N and Y (because Yemen is the only country that starts with Y).

5.2 Infomap Algorithm

The Infomap algorithm is based on the idea of information flow. Atlas is a game where once you reach somewhere, you cannot go back (directed edges). This aligns well with the Infomap algorithm which is based on Discrete Time Markov Chains and Random Walks. It is particularly effective for detecting small, well-defined communities.

5.2.1 Results

The results of the Infomap algorithm are as follows:

- **Infomap Modularity:** 0.436
- **Infomap Coverage:** 0.178
- **Infomap Conductance:** 0.242

These results suggest that the Infomap algorithm has detected communities with a lower modularity and coverage compared to the Louvain algorithm. The higher conductance value indicates that the communities detected by Infomap have more inter-community edges, which is an indication of a less clear-cut separation between groups.

5.2.2 Communities

The detected communities are as follows:

- **Community 1:** India, China, United States, Indonesia, Pakistan, Nigeria, Brazil, Bangladesh, Russia, Ethiopia, Mexico, Japan, Tanzania, Niger, Burkina Faso, Czech Republic, Tajikistan, Papua New Guinea, Portugal, Greece, Hungary, Belarus, Mauritania, Costa Rica, Croatia, Armenia, Lithuania, Jamaica, Albania, Botswana, Guinea-Bissau, Trinidad and Tobago, Bhutan, Suriname, Malta, Maldives, Micronesia, Cabo Verde, Brunei, Belize, Bahamas, Iceland, Vanuatu, Barbados, Sao Tome & Principe, Samoa, Saint Lucia, Kiribati, Seychelles, Grenada, Tonga, St. Vincent & Grenadines, Antigua and Barbuda, Andorra, Dominica, Saint Kitts & Nevis, Tuvalu
- **Community 2:** Egypt, Philippines, DR Congo, Vietnam, Turkey, Germany, Thailand, United Kingdom, New Zealand, Eritrea, Mongolia, Bosnia and Herzegovina, Qatar, Moldova, Bahrain, Solomon Islands, Liechtenstein, San Marino
- **Community 3:** Iran, Laos, Turkmenistan, Kyrgyzstan, Paraguay, Nicaragua, Bulgaria, Serbia, El Salvador, Congo, Denmark, Singapore, Lebanon, Finland, Oman, Timor-Leste, Estonia, Cyprus
- **Community 4:** France, South Africa, Italy, Uruguay, Nauru

- **Community 5:** Kenya, Myanmar, Colombia, South Korea, Sudan, Uganda, Spain, Algeria, Iraq, Argentina, Afghanistan, Yemen, Canada, Poland, Morocco, Angola, Ukraine, Uzbekistan, Malaysia, Mozambique, Ghana, Peru, Saudi Arabia, Madagascar, Côte d'Ivoire, Nepal, Cameroon, Sweden, Israel, Georgia, Gambia, Lesotho, Equatorial Guinea, Mauritius, Eswatini, Djibouti, Fiji, Comoros, Guyana
- **Community 6:** Venezuela, Australia, North Korea, Marshall Islands
- **Community 7:** Syria, Sri Lanka, Malawi, Zambia, Kazakhstan, Chad, Chile, Romania, Somalia, Senegal, Guatemala, Netherlands, Ecuador, Cambodia, Zimbabwe, Guinea, Benin, Rwanda, Burundi, Bolivia, Tunisia, South Sudan, Haiti, Belgium, Dominican Republic, United Arab Emirates, Cuba, Honduras, North Macedonia, Monaco
- **Community 8:** Mali, Ireland, Kuwait, Panama, Latvia
- **Community 9:** Jordan, Azerbaijan, Switzerland, Sierra Leone, Libya, Liberia, Norway, Slovakia, Central African Republic, Namibia, Gabon, Slovenia, Luxembourg, Montenegro, Palau
- **Community 10:** Togo, Austria

As it can be seen, the communities make somewhat less sense than the communities detected by the Louvain algorithm. For example, Morocco and Monaco, although having the same indegree and outdegree, are separated into different communities. Furthermore, countries ending with O are also separated in different communities, which is different from Oman.

That being said, there is some similarity in the communities. For example, Kenya, Colombia etc end with A and are placed in the same community. This suggests that the community detection method was not up to the mark, but still better than a randomized community detection algorithm.