A-T-L-A-S Atlas!

Nisarg Suthar, Microsoft, India.

Tasks Completed

1. Dataset Creation ✓
2. Graph Analysis ✓
3. Community Detection ✓
4. Bonus
   (a) Link Prediction using Node2Vec ✓
   (b) Link Prediction using GNN ✓
5. Paper Reading Task ✓

I. INTRODUCTION

ATLAS is a classic word game where players name geographical locations. Each name must start with the last letter of the previous one (e.g., Madagascar → Russia → Armenia). The game ends when a player cannot provide a valid, previously unused location. This project analyzes the game using complex network theory. It models countries and cities as nodes in a directed graph, where an edge exists if one location's name ends with the letter that the next location's name begins with. Additionally, we dive deep into the Node2Vec [4] algorithm for graph representations.

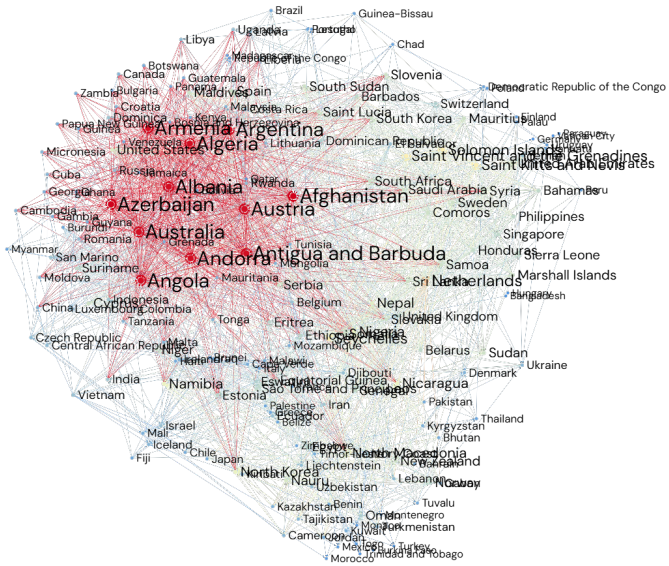

FIG. 1: The ATLAS game graph for countries. [Made using Gephi [1]]

II. DATASET CREATION

The following 3 datasets were created using web scraping with the latest information available as of date on the respective sources.

- Countries Dataset (Names of all 195 UN recognised countries)
- Cities Dataset (Names of the top 500 cities by population globally as of Dec 10, 2025)
- Combined Dataset (The union of the above 2 lists)

The information regarding the country names was obtained from Wikipedia [9] with no changes made to the official names recorded for the countries in the UN list of recognised countries.
The list of cities was obtained from the Geonames dataset on Huwise DataHub. [3]
Further filtering and data formatting can be referred from the code on my GitHub repository [7] under the data folder.

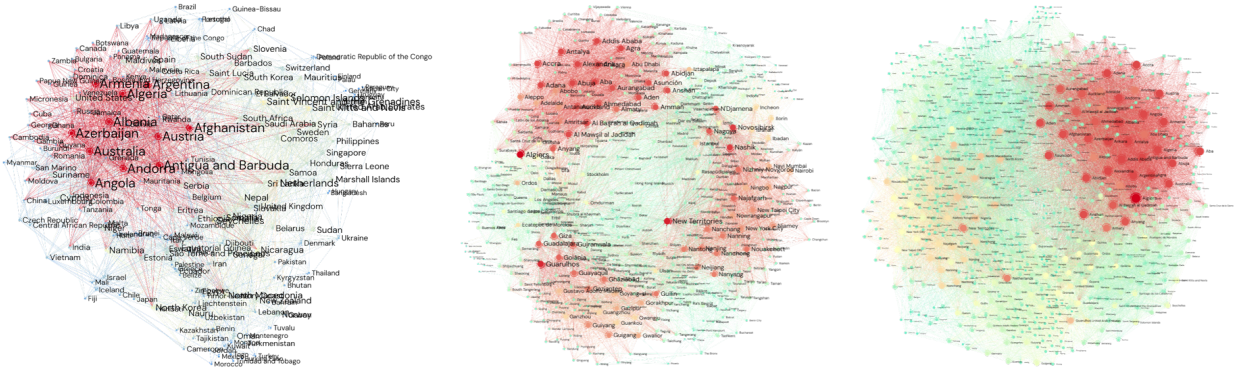

FIG. 2: Graph visualization for the three datasets. From left to right: Countries, Cities, Combined. [Made using Gephi [1]]

III. TASK: GRAPH ANALYSIS

All of the graph analysis has been done by leveraging the NetworkX [5] python library.

Number of nodes in Country Network = 195
Number of nodes in City Network = 496
Number of nodes in Combined Network = 691

Number of edges in Country Network = 2035
Number of edges in Country Network = 8895
Number of edges in Country Network = 19666

A. Sure Shot Wins

Anyone who says Norway after Yemen wins!

1. Based on Cut Nodes

Cut vertices disconnect the graph. If there is a cycle from the cut vertex to itself, then the cycle can be used as a sure shot win.

For example in the country graph, Yemen and Oman are cut vertices. I have prepared a list of 3-length, 4-length, 5-length cycles that bring you back to Oman or Yemen.

Yemen → Nauru → Uruguay
Oman → Nepal → Lesotho
and so on,

Refer the `code/analysis/winning_paths.txt` for more paths.

B. Network Level Properties

1. Degree Distribution

I was looking for scale free distribution to see if the network is resilient to node deletions. However, i could not see any significant scale-free behaviour proving that the node deletions are significant for the graph structure. This makes sense as the game prohibits repeating a country which translates to node deletion in the graph. The plots can be seen in under the `code/analysis/network_properties` folder in the repo.

2. Mean Degree

Mean Degree for Country Network: 10.435
Mean Degree for City Network: 17.933
Mean Degree for Combined Network: 28.501

Mean degree is higher for the combined graph reflecting availablility of more options to go from a place in the combined network. Also, cities mean degree is higher than that of country meaning there are more options available at each node. This means the game becomes easier if you have the knowledge of all the places. Mean In-degree and Out-degree values are similar.

3. Density

Density of Country Network: 0.0537
Density of City Network: 0.0362
Density of Combined Network: 0.0413

The density of country network is highest, meaning the nodes are well connected than expected number of links randomly. The lower number density in the other graphs signify there are lower number of node pairs not being an edge due to the ATLAS game constraints. So game in the country is less constraint relative to the number of nodes.

4. Assortativity

Degree assortativity of Country Network: -0.286
Degree assortativity of City Network: -0.133
Degree assortativity of Combined Network: 0.0103

Assortativity is relatively more in the combined graph, meaning higher degree nodes prefer to attach to other higher degree nodes. The other 2 networks have negative assortativity meaning that higher degree nodes lead to lower degree nodes and vice versa. This is important as we want to be on a higher degree node and force the opponent to have lesser options in order to win.

5. Transitivity

Transitivity of Country Network: 0.310
Transitivity of City Network: 0.107
Transitivity of Combined Network: 0.168

Higher transitivity means that if A can lead to B, and B can lead to C, then A can often lead directly to C. The game has redundant paths. It can help to strategise and create odd length cycles which are a sure shot win. There is higher chance to do this in the country graph.

6. Avg Clustering Coefficient

Avg Clustering Coefficient of Country Network: 0.180
Avg Clustering Coefficient of City Network: 0.061
Avg Clustering Coefficient of Combined Network: 0.0841

Measures how tightly a node's neighbors connect with each other. High clustering around a country means the countries that are reachable from it tend to be inter-reachable. There is a safe zone where nodes can lead to one another instead of going down into a single trajectory. This means that the country graph is safer to play in the beginning.

C. Edge Level Properties

1. Betweenness

Higher edge betwenness means there is high chance that the game play is going to visit either of the 2 nodes. Travelling via a high betweenness edge means going thorugh a structural chokepoint which can force the opponent to a dead end sooner.

Examples of high betweenness edges for country graph:
('Nauru', 'United Kingdom') 0.08051174865607857
('New Zealand', 'Dominican Republic') 0.05613551015612871
('Afghanistan', 'Netherlands') 0.04540561672004976

The highest edge betweenness for the edeges is more in the country graph, meaning that the country graph provides more impact on steering an opponent to a death trap.

D. Node Level Properties

I have analysed various node level properties and centrality [6][8] measures like Out degree advantage (Out Degree - In degree), Eigen Vector centrality, PageRank, HITS, Trophic Levels, and betweenness centrality. The intent is to send the opponent to a node which has lesser number of options which means lower out degree, higher eigen centrality, higher page rank, higher HITS authority, low HITS hub value, lower trophic level and higher betweenness centrality.

1. Custom Parity Property

I have come up with a custom node level property called parity which is a randomized algorithm to find the difference between the number of odd length and even length terminating paths starting at a node. An odd length path is a sure sign of win for anyone who starts that trajectory. More the number of odd length terminating paths starting at a node, higher the chance of winning.

The code generates a sample of terminating paths that start from a node and calculates the difference in an efficient way. This approach tackles the problem of huge computation times required for listing all the paths from a node.

Listing 1: Randomized Node Parity

```
1
2  def sample_maximal_paths(G, source, num_samples=1000):
3      sampled_paths = []
4
5      for _ in range(num_samples):
6          path = [source]
7          visited = {source}
8          current = source
9
10         while True:
11             neighbors = [n for n in G.successors(current) if n not in
                   visited]
12             if not neighbors:
13                 sampled_paths.append(path.copy())
14                 break
15
16             # Randomly choose next node
17             next_node = random.choice(neighbors)
18             path.append(next_node)
19             visited.add(next_node)
20             current = next_node
21
22     return sampled_paths
23
24 def source_parity_adv(G, source, **kwargs):
25     sample_paths = sample_maximal_paths(G, source, **kwargs)
26     sample_path_lengths = []
27     with ThreadPoolExecutor() as executor:
```

```
28          sample_path_lengths = list(executor.map(lambda path: 1 if len(
                path)%2 == 1 else -1, sample_paths))
29      return sum(sample_path_lengths)
```

E. Playing the GAME OF ATLAS

I have implemented the ATLAS game to be played with the computer along with helpful tournaments among strategies based on the above observations. I have plotted a heatmap to show the head-to-head strategy winnings. Also since there is some assymetry between the strategy that starts the game, I have kept all the $n \times n$ values where the i^{th} row and j^{th} columns means that the i^{th} strategy played the first move against the j^{th} strategy.

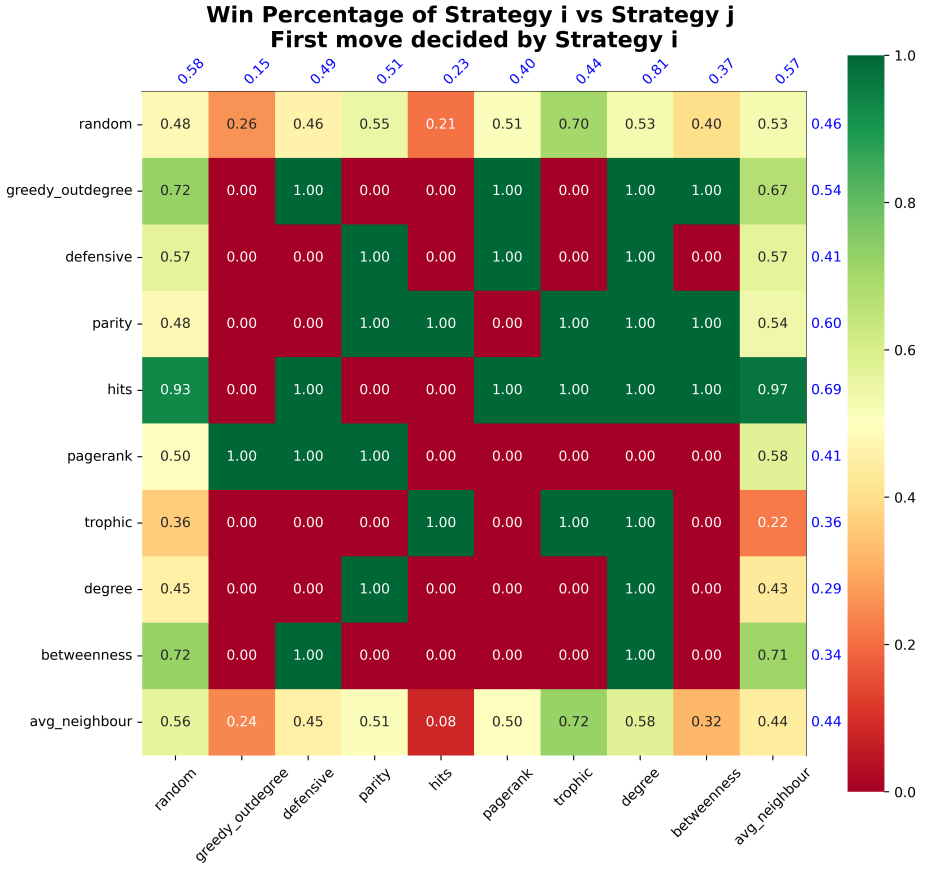

FIG. 3: The ATLAS game heatmap countries.

I observe that parity is a good approach on the cities and combined graph with lower clustering values signifying the existance of death trajectories opponents cannot escape. Whereas HITS advantage and greedy outdegree optimization also work well.

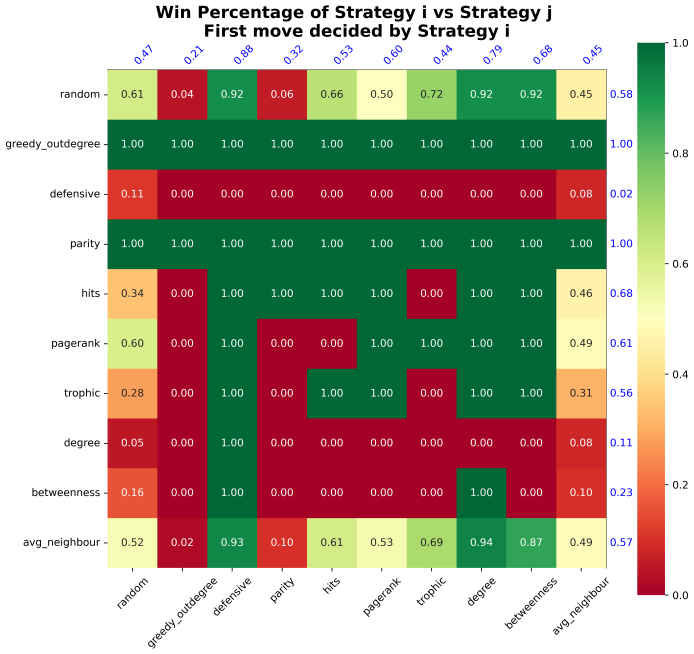

FIG. 4: The ATLAS game heatmap cities.

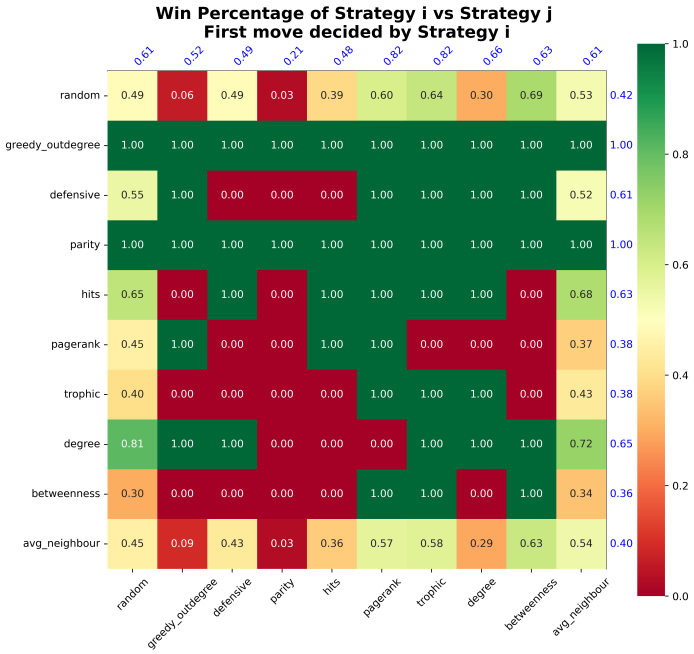

FIG. 5: The ATLAS game heatmap cities.

IV. TASK: COMMUNITY DETECTION

I performed community detection using 4 different methods:

1. Greedy Modularity Maximization
2. Lovain Community Finding Algo
3. Girvan-Newman Algo
4. K-Means clustering on node2vec vectors with p=1, q=0.1

The highest modularity was achieved with Louvain's algo.

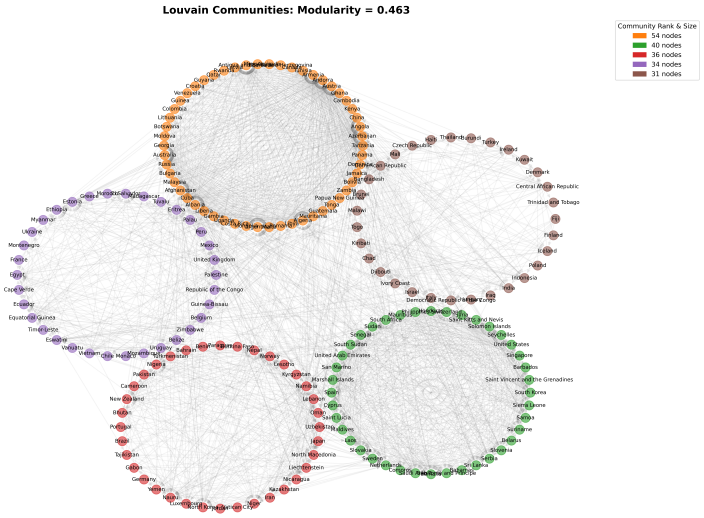

FIG. 6: Louvain Communities for Country Graph.

The communities obtained using greedy modularity and louvain's algo are not very interpretible. However, on a high level the countries that end with a common letter are in the same community.

The Girvan-Newman algo is an iterative algo removing the edge with highest betweenness centrality at every point. I have created an animation for the same. Please refer to the `code/community/animation.gif`. The communities here mostly have the countries that end with a common letter like 'a' or 's' with some other countries being a singleton node. I did 50 iterations and the highest modularity obtained was 0.423.

I also did K-Means clustering on learned node embeddings using node2vec. I found that the communities detected in this manner are more strict based on the ending letter of the country name. The odularity score for this approach is very less (0.088).

A community structure refers to a structure where the intra cluster linkage among the nodes is greater than the inter cluster linkage. Thei smeans that the game will stay inside a community more often than it shall move to a different community. One can remember all the names of the members of once community and continue to play the game for a long time. Other strategy that can be adopted is to force the opponent to a sparse community with very few nodes causing the opponent to run out of the options faster.

V. BONUS: LINK PREDICTION

Link prediction was done on the Country network for the ATLAS game by masking nodes at random such that the graph remains weakly connected. Negative edge samples were genrated at random and partitioned into non-itersecting train and test sets. The code for the dataset preparation and training can be found in the `code/bonus/link_pred.ipynb` notebook on the repo[7].

A. Using Node2Vec representations

The training dataset consisted of 1985 positive and 1985 negative edge samples.

The node2vec embeddings were obtained for the nodes by creating 500 walks of length 50 with p=0.5 and q=0.1. The dimension of the embeddings was set as 64.

The edge representations were created as a concatenation of the source node embedding with the target node embedding as the edges we try to predict are directed and we need to preserve the asymmetry in the edge embeddings for the source and target. Hence the edge embeddings had 128 dimensions.

A simple fully connected neural network model with 3 hidden layers was created with a dropout ratio of 0.2 on the second layer while training.

ReLU activation was used for intermediate layers while Sigmoid was used for the final layer to output probabilities.

The model was trained for 20 epochs with Adam optimizer with lr=0.001 using the BinaryCrossEntropy loss on Nvidia GeForce MX 450 GPU.

The training loss: 0.0035

The training prediction performance was as follows:

Confusion Matrix:

$$\begin{bmatrix} 1985 & 0 \\ 0 & 1985 \end{bmatrix}$$

TABLE I: Prediction Performance of Node2Vec powered NN on Train Set

Metric	**Score**
Accuracy	1
Precision	1
Recall	1
F1	1
AUC	1

It seems that the trained model had overfitted the training data. However, when the number of epochs or number of layers of the model ere reduced, a significant reduction in test set performance

was observed.

The trained model was tested on a test set containing 50 positive and 50 negative edge labels.
The test prediction performance was as follows:

Confusion Matrix:

$$\begin{bmatrix} 50 & 0 \\ 1 & 49 \end{bmatrix}$$

TABLE II: Prediction Performance of Node2Vec powered NN on Test Set

Metric	**Score**
Accuracy	0.99
Precision	1
Recall	0.98
F1	0.99
AUC	0.996

B. Using Graph Neural Networks (GNN)

I was experimenting with GNNs [2] for the first time and have tried a simple encoder-decoder architecture for link-prediction.

Here the training data was generated by using RandomLink split with 25% f the edges kept in the test set.

The features for the nodes were kept simple. A concatenation of 26 length vectors for start node and end node was feeded into the encoder to create embeddings of size 64.

The encoder contains 2 SAGEConv layers and the decoder is a simple fully connected neural network with 2 layers.

ReLU is used for the first layer while Sigmoid is used for the second layer.

The features for the decoder are created using the encoder embeddings and concatenating the source and target node embeddings to preserve the asymmetry of a directed edge.

The model was trained for 50 epochs with the Adam Optimizer with lr=0.005 using the BinaryCrossEntropy loss on Nvidia GeForce MX 450 GPU.
The training loss was approximately 0.

Again it seems that the trained model had overfitted the training data. However, when the number of epochs or number of layers of the model ere reduced, a significant reduction in test set performance was observed.

There were 1527 positive and negative samples in the training set and the prediction performance on the training set was flawless.

Confusion Matrix:

$$\begin{bmatrix} 1527 & 0 \\ 0 & 1527 \end{bmatrix}$$

TABLE III: Prediction Performance of GNN on Train Set

Metric	**Score**
Accuracy	1
Precision	1
Recall	1
F1	1
AUC	1

The trained model was tested on a test set containing 508 positive and 508 negative edge labels. The test prediction performance was as follows:

Confusion Matrix:

$$\begin{bmatrix} 493 & 15 \\ 3 & 505 \end{bmatrix}$$

TABLE IV: Prediction Performance of GNN on Test Set

Metric	**Score**
Accuracy	0.9823
Precision	0.9712
Recall	0.9941
F1	0.9823
AUC	0.9981

VI. EXTRA BONUS TASK

I tried implementing an RL agent to play ATLAS. However due to time constraints the work could be brought to give promising results. I used Double DQN policy using a simple fully connected neural network. The reward policy was set such that the agent get maximum reward on winning, minimum reward on losing and an intermediate reward for getting the right successor country. However, the results were not so good and I am trying to adjust the reward policy and node features to help the agent learn better.

VII. PAPER READING

The paper presentation and video links have been submitted in the form. A copy of the presentation in PDF from is also kept in the GitHub repo [7].

---

[1] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 3, pages 361–362. AAAI Press, 2009.

[2] Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaž Stojanović, Kai Bendias, Felix Faber, Maciej Balcerek, and Jan-Eric Lenssen. PyG 2.0: Scalable learning on real world graphs. *arXiv preprint arXiv:2507.16991*, 2025.

[3] GeoNames. Geonames - all cities with a population > 1000. https://hub.huwise.com/explore/assets/geonames-all-cities-with-a-population-1000, 2025. [Online; accessed 17-December-2025].

[4] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery.

[5] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, pages 11–15. Institute of Electrical and Electronics Engineers (IEEE), 2008.

[6] Mark Newman. *Networks*. Oxford University Press, Oxford, 2018.

[7] Nisarg Suthar. Atlas. https://github.com/itsNisarg/Atlas, 2025. Accessed: 16-December-2025.

[8] Stefan Thurner, R. A. Hanel, and Peter Klimek. *Introduction to the Theory of Complex Systems*. Oxford University Press, Oxford, 2018.

[9] Wikipedia contributors. List of countries and dependencies by population. — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population, 2025. [Online; accessed 17-December-2025].