PageRank

Nicola Cortinovis, Marta Lucas, Luca Pernice

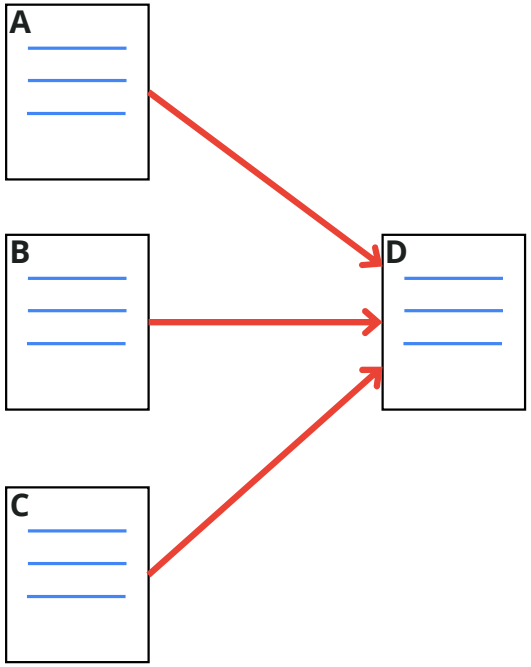
Introduction

PageRank: an algorithm developed by Google to “objectively” measure the importance of web pages.

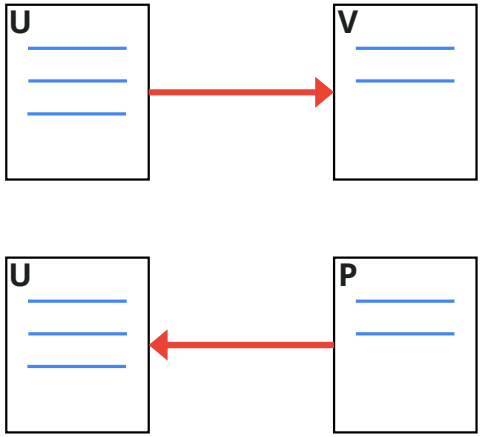
Motivations: help search engines rank results and supports smarter and more robust navigation through the web.

Content

PageRank Intuition



Pages → Nodes
Links → Edges



$$F_U = V$$
$$B_U = P$$
$$N_U = |F_U|$$

$$R(u) = c \sum_{v \in B(u)} \frac{R(v)}{N_v}$$

More resistant to search engine manipulation

More aligned to common sense notation of importance

Power Iteration

Let A be a square, stochastic, positive matrix and R a generic stochastic vector. The *Perron-Frobenius theorem* hypothesis is satisfied by A, therefore there is a **unique** eigenvector associated with the dominant eigenvalue

We can apply the Power Iteration algorithm to find an approximation of the dominant eigenvector, which is the limiting (stationary) distribution of the Markov chain described by A and correspond to the PageRank vector

$$R = AR$$

$$A = V\Lambda V^{-1} \quad \longrightarrow \quad A^k = V\Lambda^k V^{-1} \qquad R_0 = Vg \quad \longrightarrow \quad A^k R_0 = V\Lambda^k V^{-1} Vg$$

$$A^k R_0 = V\Lambda^k g = \lambda_1^k \sum_{i=1}^{n} v_i \left(\frac{\lambda_i}{\lambda_1}\right)^k g_i \xrightarrow{k\to\infty} \lambda_1^k v_1 g_1 = R_k$$

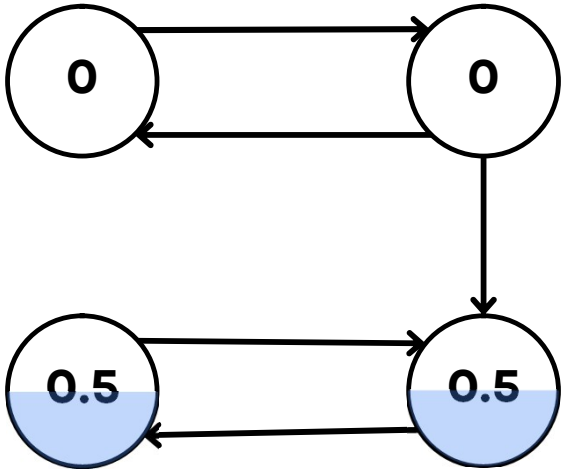
We iterate until a maximum value for k is reached or when the difference between two iterations is below a specific threshold

$$||R_k - R_{k-1}||_1 < \varepsilon$$

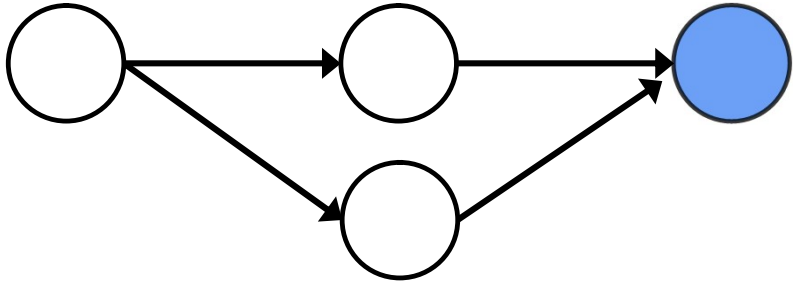
Structural Issues

No guarantee of Perron-Frobenius hypothesis

Bottom Strongly Connected Components will absorb all the probability mass (scores) in the long run leaving the other pages with a 0 rank



It is unclear how to treat **Dangling nodes** which would make the matrix not stochastic causing loss of probability mass since their contribution is not redistributed. But such pages might still be important!



$$\underbrace{\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}}_{R_0} = \underbrace{\begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0 \end{pmatrix}}_{R_1}$$

PageRank and Google Matrix

Enforce the matrix to be stochastic by assuming a uniform distribution over dangling nodes

$$\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

Update the formula by introducing a **jump vector** E and a **damping coefficient** α. These allow to escape BSCC. Now we can use **power iteration** on the **Google matrix** *G*.

$$R = \alpha(AR) + (1-\alpha)E \qquad E = \begin{pmatrix} \frac{1}{n} \\ \vdots \\ \frac{1}{n} \end{pmatrix} \qquad R_{k+1} = GR_k \quad G = \alpha A + (1-\alpha)E\mathbf{1}^T$$

Dataset

Wikipedia network of top categories

A web graph of Wikipedia hyperlinks collected in **September 2011**.

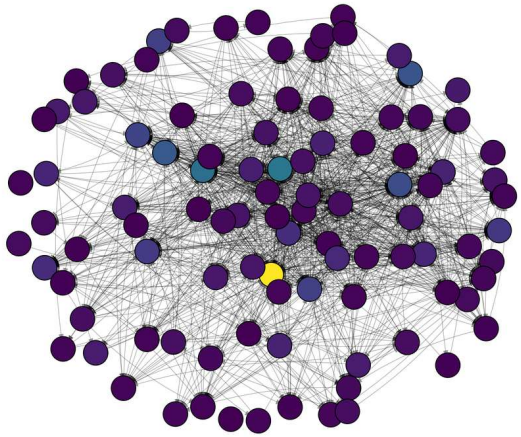
In addition to the graph, the page names and categories of the articles are provided.

Note: Nodes can be assigned to multiple categories

Source: Stanford Network Analysis Project, Jure Leskovec (https://snap.stanford.edu/data/wiki-topcats.html)

Dataset statistics

Number of nodes → 1.791.489
Number of edges → 28.511.807
Diameter → 9
Number of categories → 17.364



Implementation details

Our first *naive* implementation of PageRank used adjacency lists for the page-links dependencies. The scores get saved to a .csv file that can be loaded to avoiding recomputation.

While this worked, Python does **not** allow for parallelization, and iterating over large Python lists and dictionaries incurs significant overhead and poor cache efficiency, slowing down the score computation.

```
(IR) C:\Users\marta\OneDrive\Desktop\InfoRetrieval>python Pagerank.py
2025-07-22 22:22:31,106 - INFO - All files needed are present in data.
2025-07-22 22:22:31,107 - INFO - Loading graph edges...
2025-07-22 22:23:10,091 - INFO - Loading page names...
2025-07-22 22:23:11,535 - INFO - Loading categories...
2025-07-22 22:23:16,129 - INFO - Loaded graph with 1791489 nodes and 28508141 edges
2025-07-22 22:23:16,129 - INFO - Computing PageRank scores...
2025-07-22 22:23:16,130 - INFO - Computing PageRank using power iteration and adjacency lists...
2025-07-22 22:32:27,464 - INFO - Total PageRank score sum: 1.000000 (should be close to 1.0)
2025-07-22 22:32:27,464 - INFO - PageRank computation completed in 551.33 seconds
2025-07-22 22:32:27,465 - INFO - Converged: True, Iterations: 37
2025-07-22 22:32:37,512 - INFO - Results saved to results/wiki_pagerank_results.csv
```

Optimization

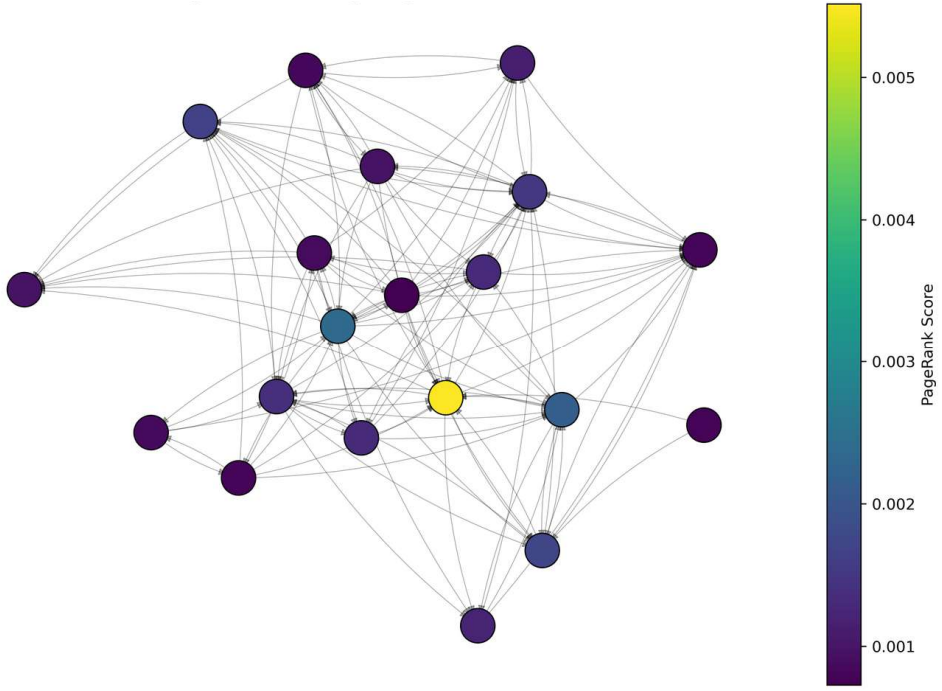
We optimized runtime by using the *Python-GraphBLAS* library **BLAS** (Basic Linear Algebra Subprograms)

The library allows for parallelization of matrix multiplication, leveraging some algebraic properties and reducing runtime significantly.

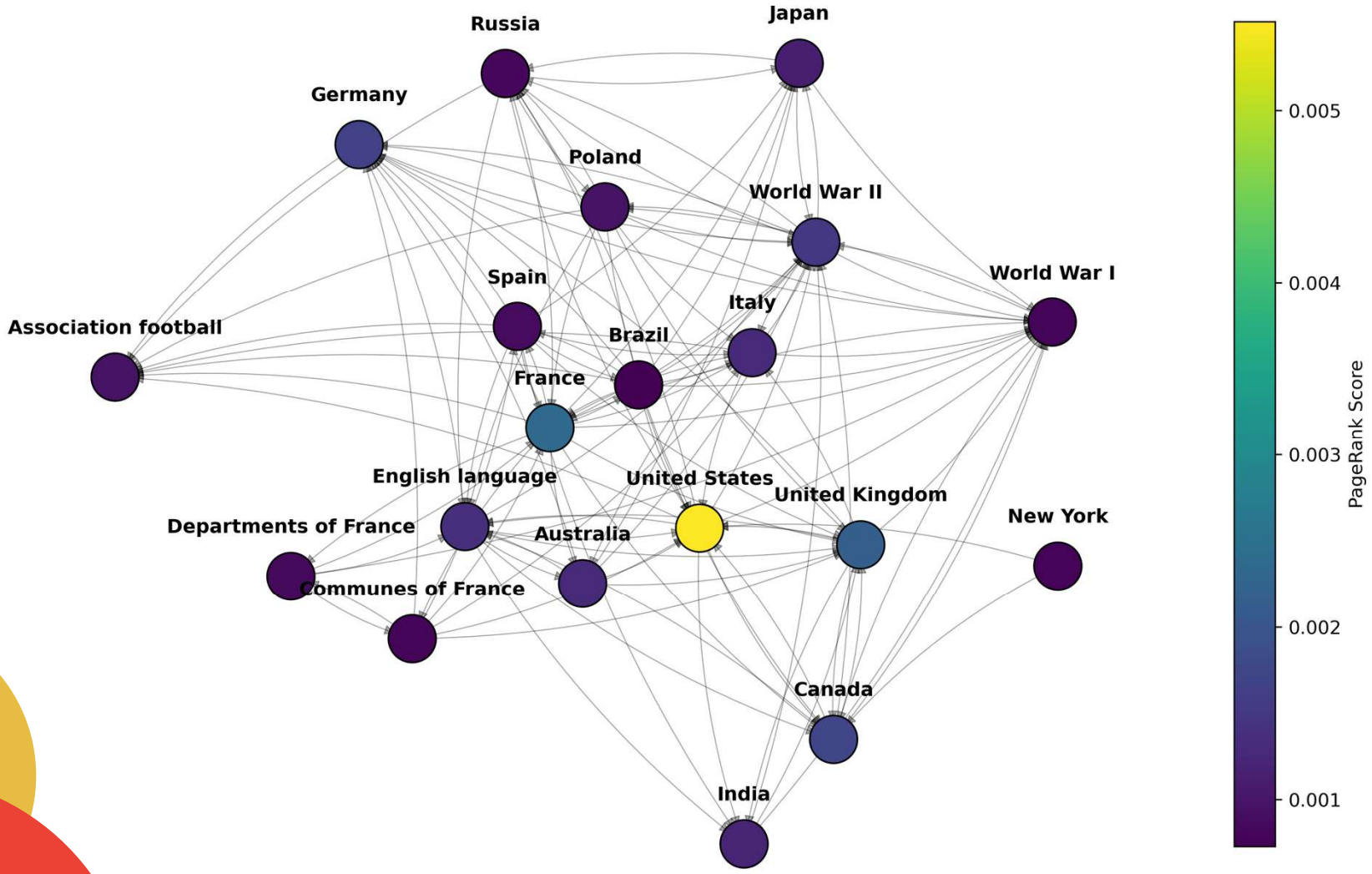
```
(IR) C:\Users\marta\OneDrive\Desktop\InfoRetrieval>python PageRank.py
2025-07-22 22:12:19,228 - INFO - All files needed are present in data.
2025-07-22 22:12:19,229 - INFO - Loading graph edges...
2025-07-22 22:13:03,536 - INFO - Loading page names...
2025-07-22 22:13:05,174 - INFO - Loading categories...
2025-07-22 22:13:10,719 - INFO - Loaded graph with 1791489 nodes and 28508141 edges
2025-07-22 22:13:10,720 - INFO - Computing PageRank scores...
2025-07-22 22:13:10,720 - INFO - Computing PageRank using matrix method leveraging GraphBLAS...
2025-07-22 22:13:21,339 - INFO - Done creating the matrix of shape: (1791489, 1791489)
2025-07-22 22:13:35,369 - INFO - PageRank completed in 14.03s
2025-07-22 22:13:35,370 - INFO - Converged: True, Iterations: 37
2025-07-22 22:13:35,802 - INFO - Total PageRank score sum: 1.000000 (should be close to 1.0)
2025-07-22 22:13:44,493 - INFO - Results saved to results/wiki_pagerank_results.csv
```

Visualizations

For graphical purpouses, instead we leveraged the *NetworkX* library



PageRank Results



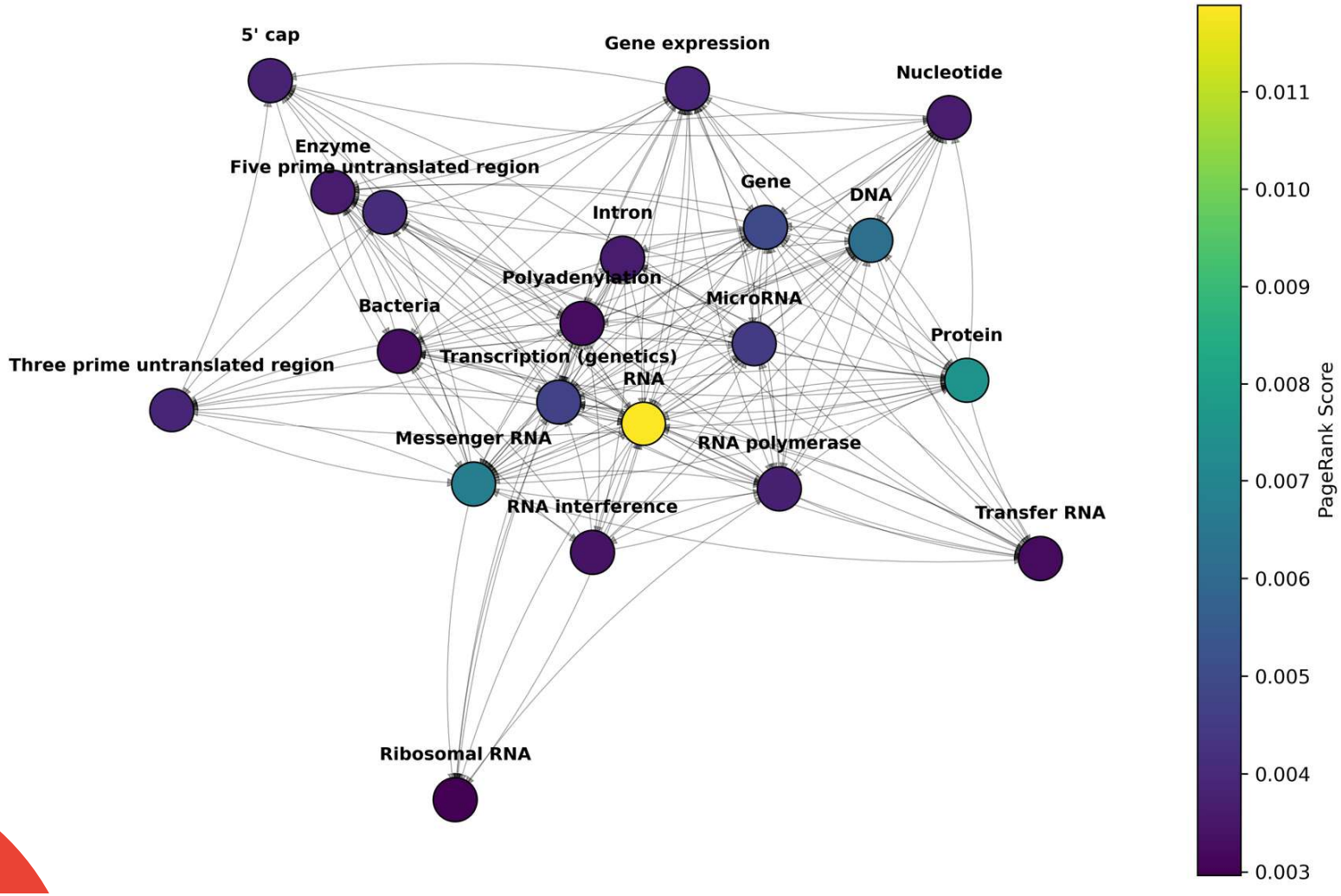
Topic specific pagerank

We can compute a topic specific PageRank by modifying the **Jump vector** to have positive uniform probabilities on only the nodes that are relevant to the topic and **0** on the rest

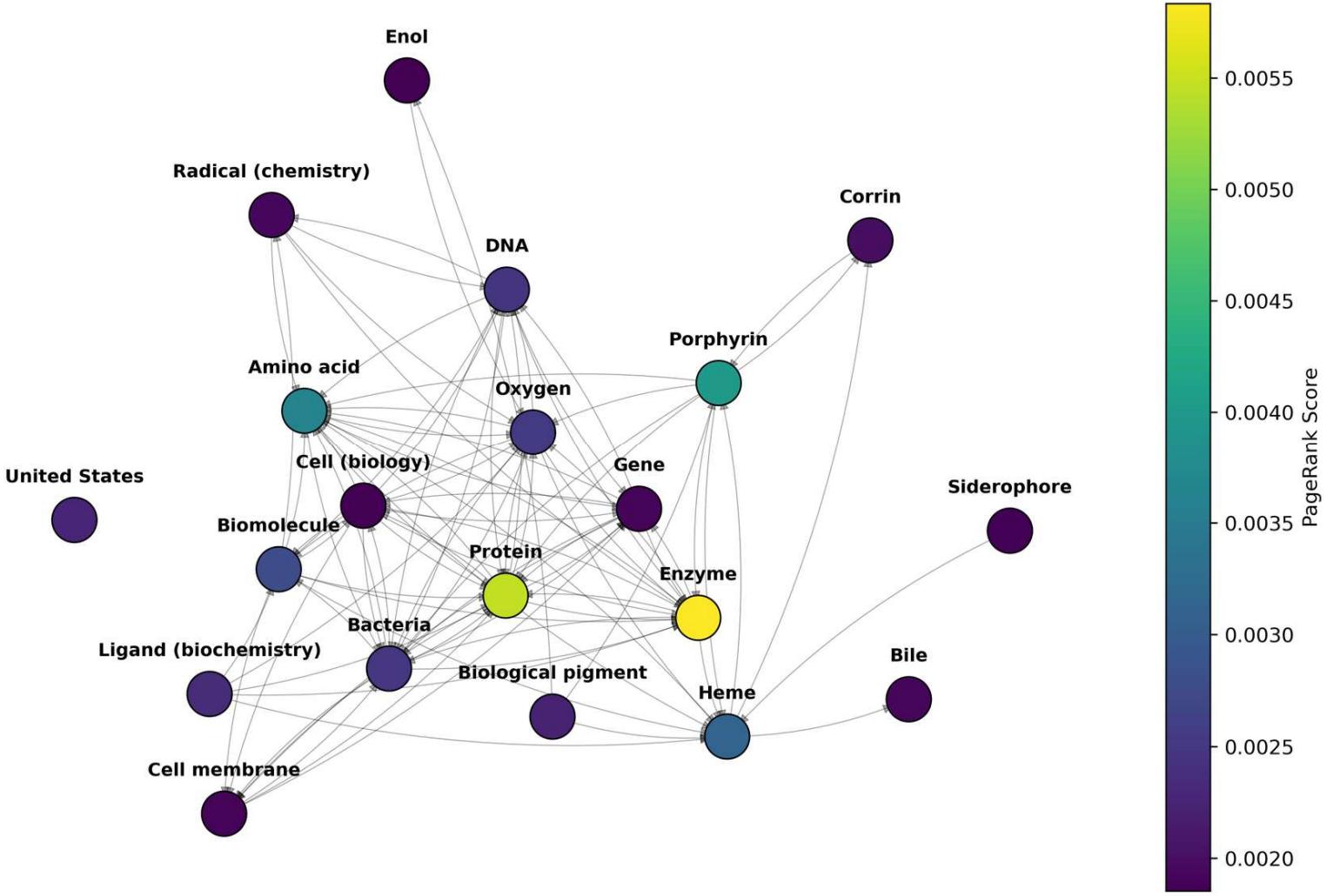
$$S = \{\text{nodes pertaining a certain topic}\} \qquad E_i = \begin{cases} \frac{1}{|S|} & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$$

This fundamentally changes the Markov chain and therefore the stationary distribution to which it converges by adding this **bias** towards topic nodes and consequentially to their close neighbors

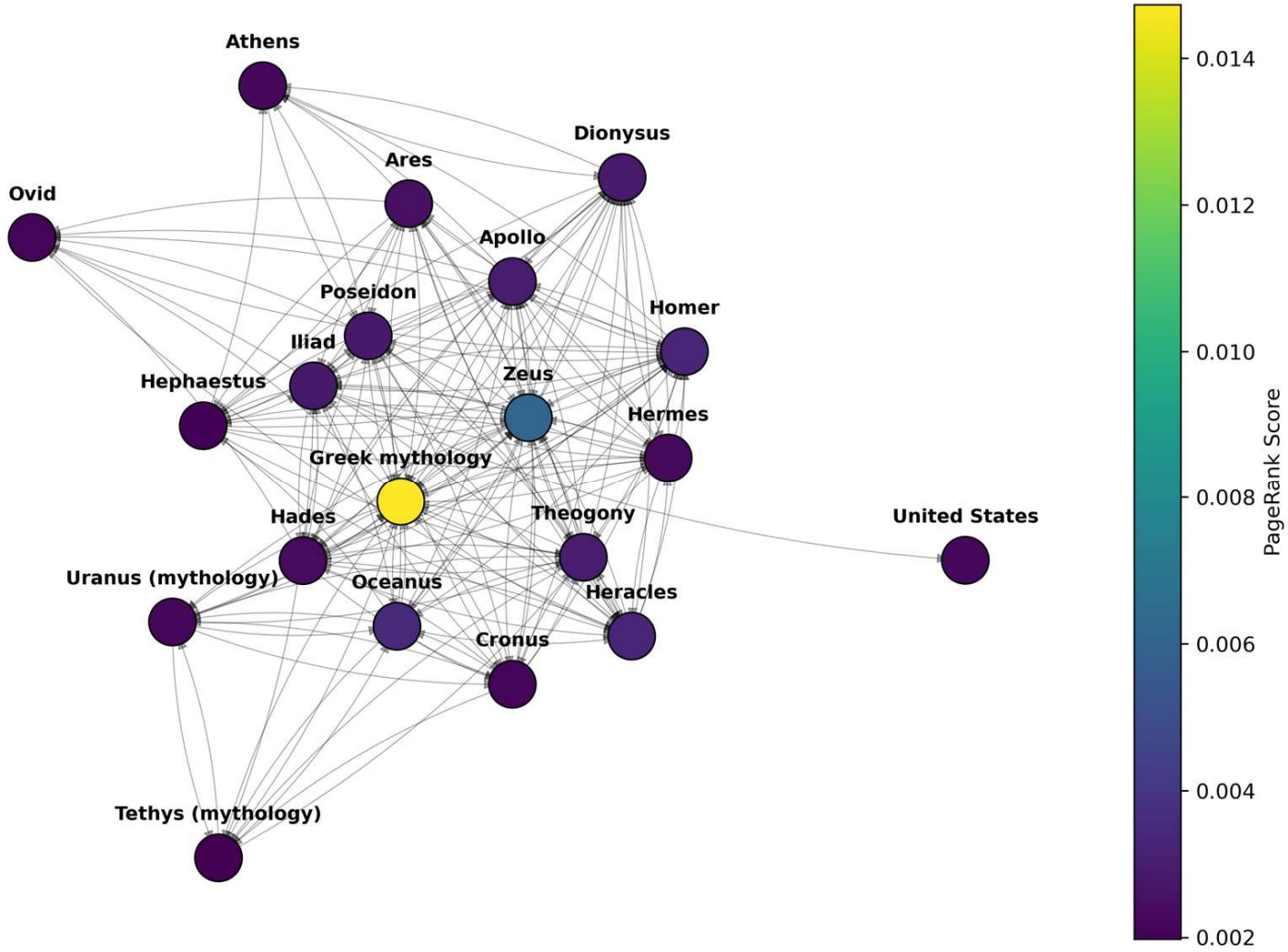
Topic Specific PageRank Results: RNA



Topic Specific PageRank Results: Biomolecules



Topic Specific PageRank Results: Greek Gods



Personalized PageRank

We can build on top of Topic Specific PageRank by introducing **Personalized PageRank**. The idea consists of a simple **linear combination** of different Pageranks, which can refer to different topics or represent different users' interests

$$\mathrm{P}_{\mathrm{new}} = \sum_{i=1}^{k} w_i \mathrm{P}_i \qquad \sum_{i=1}^{k} w_i = 1 \qquad w_i > 0 \ \forall i$$

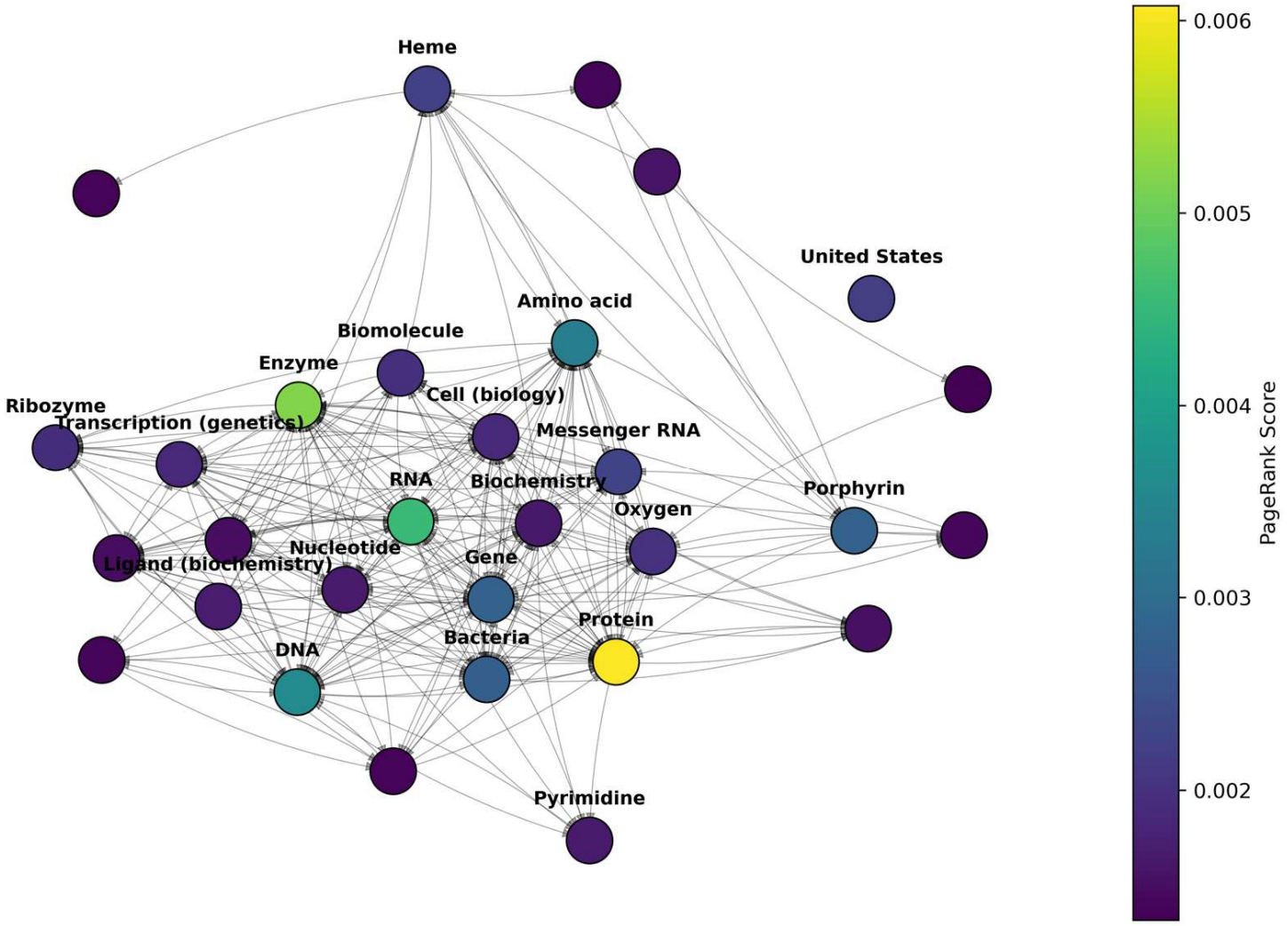
Personalized PageRank Results: RNA and Biomolecules

Top 30 nodes

<u>Weights:</u>

0.3 RNA

0.7 Biomolecules



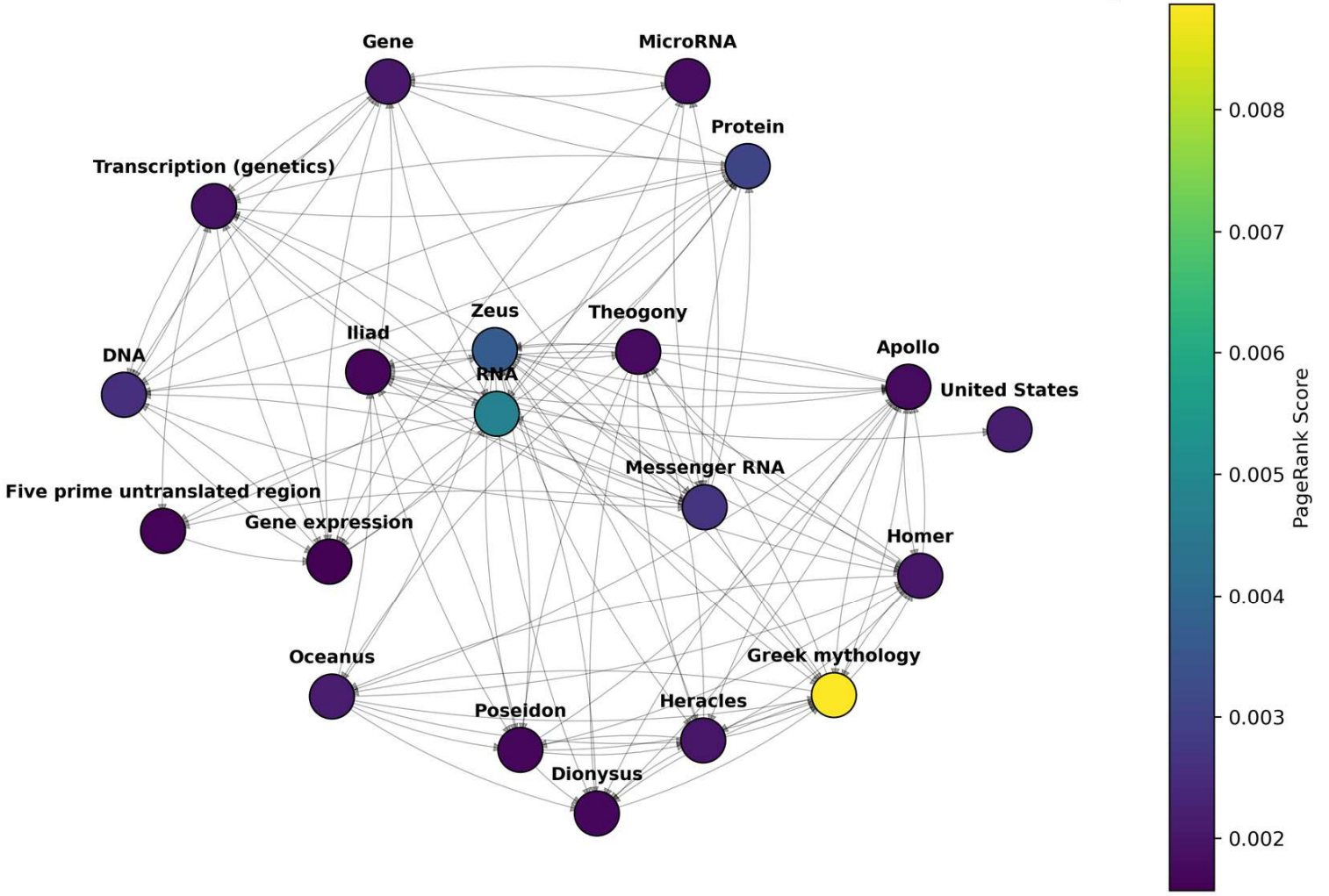
Personalized PageRank Results: RNA and Greek Gods

Top 20 nodes

<u>Weights:</u>

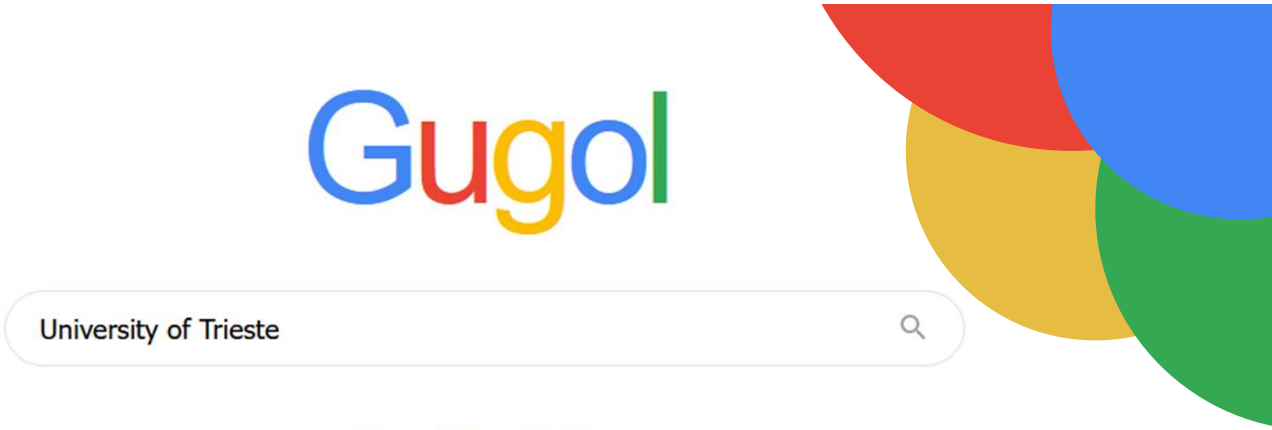
0.4 RNA

0.6 Greek Gods



Interface

We built a simple Interface using Fast API to perform queries on our Web Graph



PageRank today

PageRank remains a foundational concept, but it is just one of hundreds of **signals** used by Google.

With the rise of machine learning and natural language understanding, ranking now considers, among many other things, **user intent**, **content quality** and **context.**

Thank you for the attention!

References

Page, L., Brin, S., Motwani, R. and Winograd, T., The PageRank Citation Ranking: Bringing Order to the Web, Stanford InfoLab, 1999

Brin, S. and Page, L., The anatomy of a large-scale hypertextual Web search engine, Computer networks and ISDN systems, 30(1-7), pp.107-117, 1998

Jure Leskovec, Stanford Network Analysis project, https://snap.stanford.edu/data/wiki-topcats.html