

Greek Mythology Explorer – Final Report

Natural Language Processing and Information Retrieval

-

Final Group Project

Elyesa Duru - 50251140

Hamidreza Rahimian - 50251141

Tim Leute - 50251142

Dilara Alkanalka - 50251145

Content

[1 Introduction 3](#_Toc200350721)

[1.1 Motivation 3](#_Toc200350722)

[1.2 Project goals and use case 3](#_Toc200350723)

[2 Data Acquisition and Graph Modelling 4](#_Toc200350724)

[2.1 Data source: Wikidata and SPARQL 4](#_Toc200350725)

[2.2 Graph construction: Nodes, edges and relationships 4](#_Toc200350726)

[3 System Architecture 4](#_Toc200350727)

[3.1 Technology stack overview 4](#_Toc200350728)

[3.2 Web integration and interface 4](#_Toc200350729)

[3.3 Migration from early prototypes 4](#_Toc200350730)

[4 Core Features 4](#_Toc200350731)

[4.1 Interactive graph exploration 4](#_Toc200350732)

[4.2 Shortest path queries 4](#_Toc200350733)

[4.3 Character similarity matching via vector comparison 4](#_Toc200350734)

[5 Evaluation 4](#_Toc200350735)

[5.1 Evaluation setup and methodology 4](#_Toc200350736)

[5.2 Relevance metrics: Precision@K, MAP, NDCG 4](#_Toc200350737)

[5.3 Results and Interpretation 4](#_Toc200350738)

[6 Conclusion and Outlook 4](#_Toc200350739)

7 References ………………………………………………………………………………………………..4

# Introduction

## Motivation

Greek mythology encompasses a vast network of characters—such as gods, deities, titans, and heroes—connected by complex familial and social relationships. Understanding these connections can be challenging when relying solely on traditional text-based resources or websites, the information is often spread out and must be gathered and connected by the reader. Given the relational structure of mythological characters, using a graph data to represent and explore the information is a natural and effective approach by modeling the characters as nodes and relationship as edges. nodes and relationship. This data structure makes it possible to explore connections interactively and visualize the mythological universe in a way that highlights structure and complexity. Our project leverages these strengths of graph-based modeling to offer a more accessible and intuitive way to study and navigate Greek mythology.

## Project goals and use case

The primary goal of this project is to develop an interactive and informative platform for exploring Greek mythological characters and their relationships through graph-based data representation. By leveraging structured data and graph analysis, the tool enables users to intuitively search for characters, visualize connections such as family ties or rivalries, and better understand the intricate narratives of Greek Mythology. Additional features such as shortest path queries and a preference-based character matching quiz make the platform both informative and engaging. The project serves educational and exploratory purposes by simplifying access to complex mythological networks.

# Data Acquisition and Graph Modelling

## Data source: Wikidata and SPARQL

Wikidata is a structured knowledge base where entities (items) are assigned unique identifiers starting with "Q" (e.g., Zeus is represented by Q34201), and relationships (properties) are denoted by identifiers starting with "P" (e.g., "father" is represented by P22). These identifiers enable precise and language-independent querying across domains such as mythology[[1]](#footnote-1) (Pastor-Sánchez et al., 2021).

To extract relevant data, the Wikidata Query Service was utilized, which supports SPARQL—a powerful query language for RDF (Resource Description Framework) data. The query was formulated to target mythological characters and their interrelations, resulting in the retrieval of a dataset containing entities labeled with names, types (such as god, deity, titan), and their corresponding relationships (e.g., mother, spouse, killed by). The query results were exported as a CSV file, serving as the raw input for the graph model.

## Graph construction: Nodes, edges and relationships

After obtaining the raw dataset from Wikidata, the next step involved modeling the data as a graph to enable structured exploration and analysis. In the constructed graph, each mythological character is represented as a node, while each relationship between characters—such as "mother," "father," "sibling," "spouse," or "killed by"—is modeled as a directed edge connecting two nodes.

The graph was built using NetworkX, a Python library designed for the creation, manipulation, and analysis of complex networks. This representation allows us to treat mythology as a relational graph, enabling effective traversal, visualization, and analysis of inter-character relationships. The final graph consists of 529 nodes and 1358 edges, reflecting the rich and interconnected nature of Greek mythology.

# System Architecture

## Technology stack overview

For this project, we had two development phases: Phase One and Phase Two.

In Phase One, we attempted to implement the project using local HTML, JavaScript, and some Python programs to visualize the data, create nodes and relationships, and run everything locally due to the large volume of data. However, this approach was not the most sensible or effective way to present the data, especially since the project involved around 600 nodes and more than 2,000 relationships. Therefore, we moved the Neo4J , Sandbox , HTML serves , API and URLs for data sharing in second phase to provide a smoother and better result for our search engine.

## Web integration and interface

To improve performance and user experience, we decided to switch to online services to speed up the project and provide a smoother user interface.

We then moved from using local CSV files to an online database sandbox. In the next step, we transitioned to using the graph database Neo4j. Neo4j is a graph database that allows us to easily create and manage nodes and relationships using its query language called **Cypher**.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1 : Neo4J interface

Using Neo4j enabled us to run the entire script online instead of on our local machines. We linked the online database with Neo4j, defined the nodes and relationships there, and then exported the data from Neo4j to our online website via a URL and API.

This transition significantly boosted our project’s performance. By running the project in the cloud instead of locally, it became much faster, more reliable, and more user-friendly.

## Migration from early prototypes

We no longer use Python, CSV files, or local HTML files. Instead, we switched to online tools like Neo4j (a graph database) , Databases like Sandbox and real HTML with online database services. This lets us visualize, edit, and run our JavaScript scripts directly in the cloud, making everything faster, smoother, and more reliable.

# Core Features

## Interactive graph exploration

The coding was challenging because we used Cypher, a lightweight query language for Neo4j. It works like basic logic, asking the database to find nodes, relationships, and how they’re connected. Each node represents a character with attributes like ID, name, and type. Relationships (e.g., father, mother, sibling) link the nodes and are labeled by type. We also assigned different colors to node types to make the graph easier to read. To simplify things further, we added a feature to display selected groups of nodes like 25, 100, or all nodes. Finally, we implemented a search function that finds a specific character and all related nodes and relationships.

A screenshot of a computer code

AI-generated content may be incorrect.

Figure 2: example of coding in Neo4J , Cypher Language

## Shortest path queries

**The shortest path is easy to understand.** It takes two nodes (characters) and searches through all related nodes to find the shortest connection between them. If no direct match is found, it continues searching through connected nodes. Once found, the path is visualized on the graph and also exported as a file showing which characters are connected and the type of relationship between them.  
We use **breadth-first search (BFS)** under the hood, as implemented by Neo4j’s built-in shortestPath() function. This algorithm explores all neighboring nodes level by level, starting from both source and target nodes simultaneously (bidirectional BFS), until it finds the shortest path in terms of the number of relationships (hops). It does not take relationship weights into account and is ideal for finding the simplest connection in an unweighted graph.

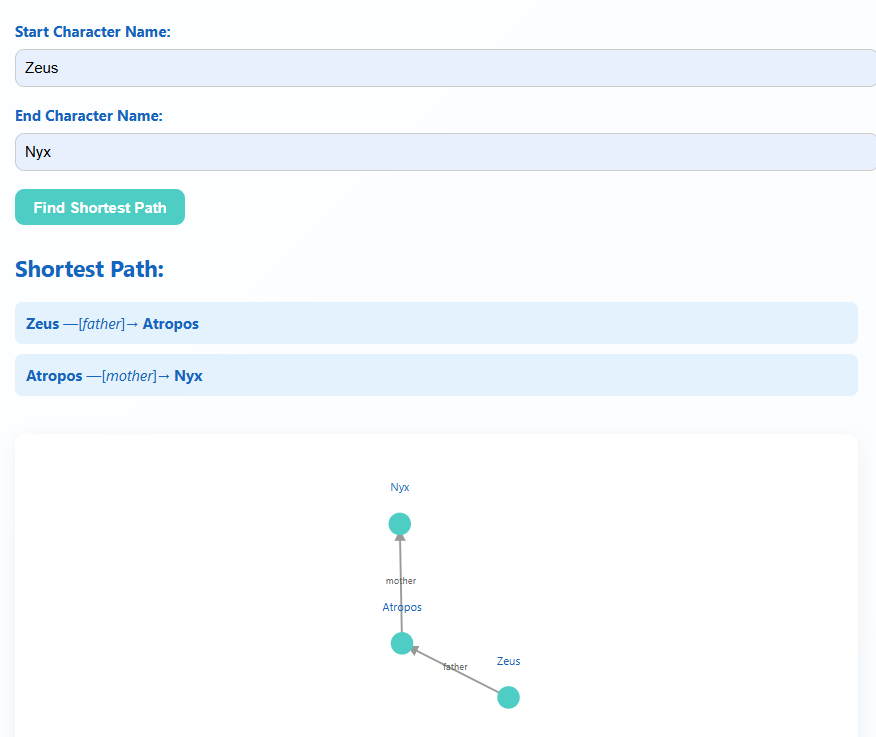


Figure 3: Example of Shortest Path connection

## Character similarity matching via vector comparison

Our platform includes a character similarity feature that recommends mythological figures based on user-defined preferences. Users adjust sliders for traits like:  
children, kills, spouses, popularity, influence, siblings.

These inputs are transformed into a numeric user vector. Each character is also represented as a vector, based on relationship data from Neo4j. We calculate similarity using Euclidean distance—lower values indicate stronger matches.

Figure 4: Interface with sliders to define character preferences

The matching logic is implemented in JavaScript:

javascript

KopierenBearbeiten

const userVec = sliderConfigs.map((\_, i) => +document.getElementById(`slider${i}`).value);

function euclideanDistance(a, b) {

return Math.sqrt(a.reduce((sum, v, i) => sum + (v - b[i]) \*\* 2, 0));

}

Each character's profile is built from graph relationships:

javascript

KopierenBearbeiten

const charVec = [

inc.filter(e => ["father", "mother"].includes(e.type)).length,

inc.filter(e => e.type === "killed by").length,

out.filter(e => e.type === "spouse").length,

inc.length, out.length,

out.filter(e => e.type === "sibling").length

];

The results are ranked and explained, showing both match score and traits, helping users explore mythology in a personalized way.

# Evaluation

Note on Sources  
The theoretical foundations and evaluation methodology in this section are based on materials from the course lecture (*Lecture 04, 2025, Chung-Ang University*) as well as the following standard IR textbooks: *Introduction to Information Retrieval* by Manning, Raghavan, and Schütze (2008), and *Modern Information Retrieval* by Baeza-Yates and Ribeiro-Neto (1999, 2011).

## Evaluation setup and methodology

To evaluate the effectiveness of our character matching system, we follow standard IR evaluation methodology. Since direct user satisfaction is hard to measure, we use relevance as a proxy.[[2]](#footnote-2) A result is considered relevant if the suggested character meaningfully matches the user-defined profile.

Our setup includes:

* A set of **test queries** based on user-defined slider inputs.
* **Manual relevance judgements** for the top-k results per query, using binary and graded scales.[[3]](#footnote-3)
* Evaluation using standard IR metrics.

This approach aligns with the classic IR evaluation framework: a document collection, a set of information needs, and relevance assessments.[[4]](#footnote-4)

## Relevance metrics: Precision@K, MAP, NDCG

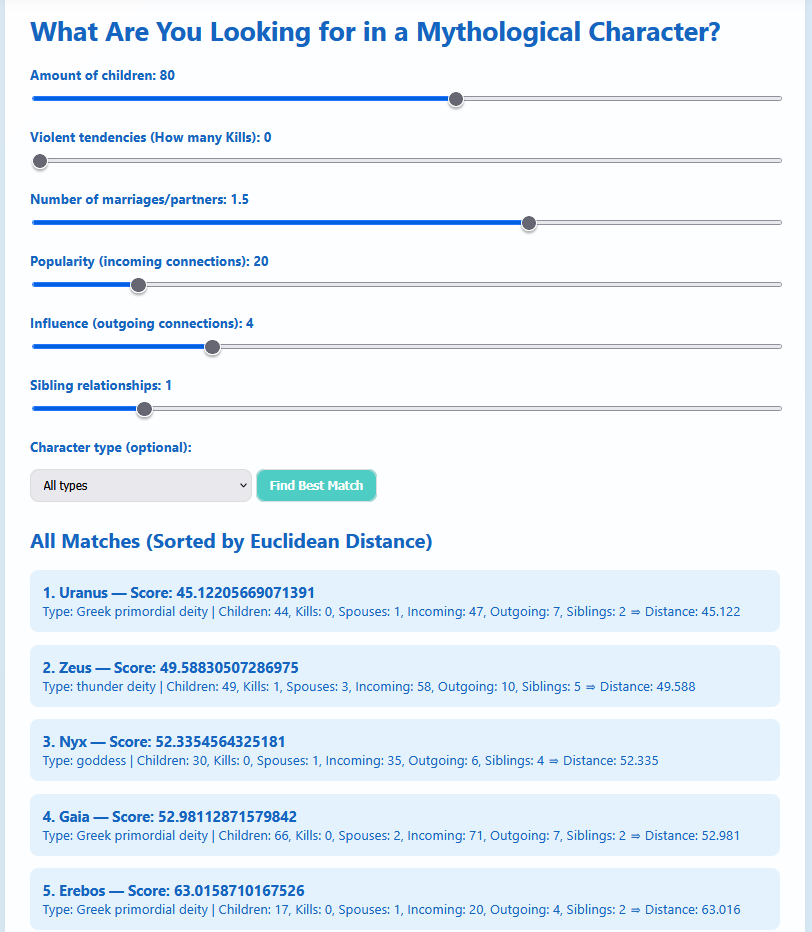
We apply three widely used IR metrics:

* **Precision@K**: Measures the proportion of relevant results in the top K[[5]](#footnote-5). It reflects how well the system ranks relevant characters at the top.
* **Mean Average Precision (MAP)**: Averages the precision at each relevant result position, then across all queries.[[6]](#footnote-6) It rewards systems that rank relevant results higher.
* **Normalized Discounted Cumulative Gain (NDCG)**: Accounts for graded relevance and discounts lower-ranked results logarithmically.[[7]](#footnote-7) It is especially useful when not all relevant results are equally important.

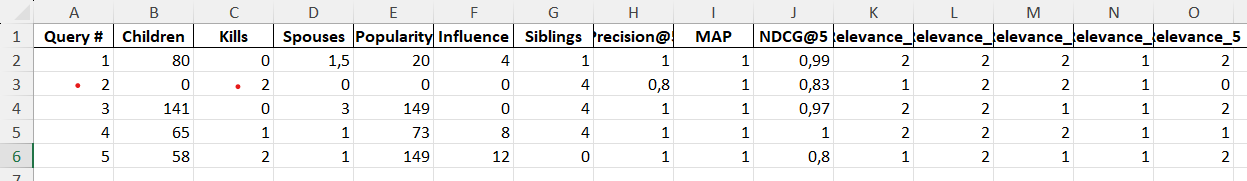
These metrics allow us to compare different configurations of our matching algorithm and assess ranking quality in a structured way.

## Results and Interpretation

To evaluate the effectiveness of our character matching system, we conducted five test queries using different slider configurations. Each query returned a ranked list of mythological characters based on Euclidean distance to the input profile. As an example, the screenshot below shows one such query and its top results:



To assess the quality of these results, we manually assigned graded relevance scores (0 = not relevant, 1 = somewhat relevant, 2 = highly relevant) to the top 5 matches for each query. Based on these judgments, we computed three standard IR metrics: **Precision@5**, **Mean Average Precision (MAP)**, and **Normalized Discounted Cumulative Gain (NDCG@5)**. The following table summarizes the results:



* **Key Observations:**
* **Precision@5** was **1.0** for 4 out of 5 queries, indicating that most top-ranked results were relevant.
* **MAP** reached the maximum value of **1.0**, showing that relevant characters were consistently ranked at the top across all queries.
* **NDCG@5** values ranged from **0.80 to 1.00**, reflecting that highly relevant characters were generally placed in higher positions, with minor variations in ranking quality.

These results suggest that our system is effective at identifying and ranking mythologically similar characters based on user-defined traits. The combination of graph-based data modeling and vector similarity appears to yield meaningful and interpretable results.

# Conclusion and Outlook

This project successfully demonstrated the utility of graph-based data representation for exploring the intricate relationships among characters in Greek mythology. By transforming mythological entities and their connections into a structured network of nodes and edges, the developed platform allows users to intuitively navigate, visualize, and analyze complex mythological narratives. The integration of Neo4j and a web interface enabled interactive features such as character search, shortest path discovery, and preference-based character matching enhances user engagement and provides meaningful insights into the rich tapestry of mythological stories.

Furthermore, the character analysis feature was evaluated using standard information retrieval metrics such as Precision@K, MAP, and NDCG. The results demonstrated strong accuracy in ranking relevant mythological characters based on user preferences, confirming that the combination of graph-based modeling and similarity measures effectively supports personalized recommendations. This evaluation underscores the practical value of our approach and its potential for further refinement.

Looking forward, to further improve the platform, the dataset could be expanded to include additional mythologies or more comprehensive relations to broaden the scope of exploration. Incorporating advanced graph algorithms, such as community detection or centrality measures, may offer deeper analytical insights. Moreover, improving the user interface with more sophisticated visualization techniques and real-time interaction could increase accessibility and engagement for a wider audience.

# Team Work Distrubution

* Tasks were chosen based on personal interests
* Workload was balanced across the team
* Everyone contributed equally

| **Name** | **Main Tasks** |
| --- | --- |
| **Dilara** | Data Retrieval, Python Testing, Info Retrieval, PowerPoint, Report |
| **Elyesa** | Neo4j Character Analysis, Presentation, Report |
| **Tim** | Character Evaluation, User In, Report |
| **Hamidreza** | Neo4j Migration & Shortest Path, Information Retrieval, Report |

# References

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern information retrieval (1st ed.). Addison-Wesley.

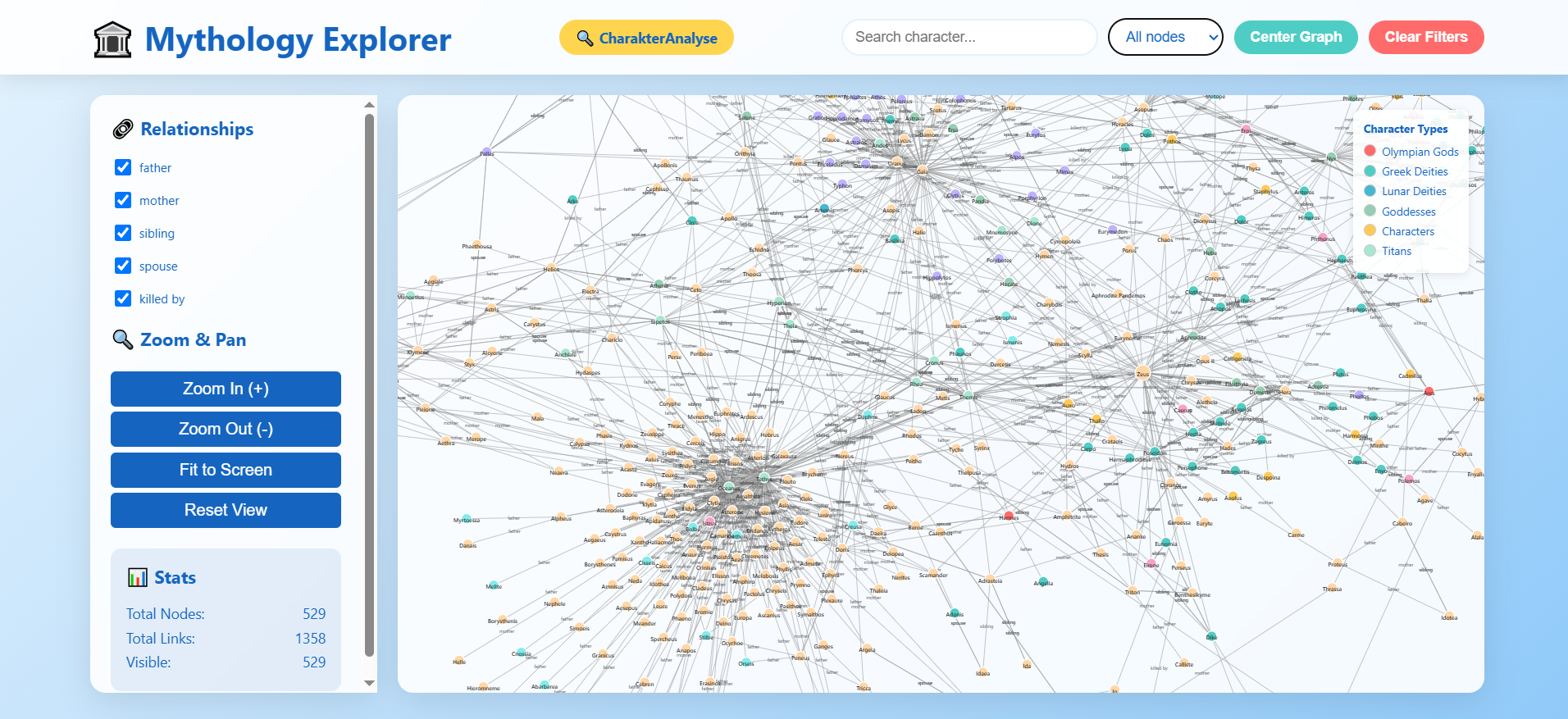
Baeza-Yates, R., & Ribeiro-Neto, B. (2011). Modern information retrieval: The concepts and technology behind search (2nd ed.). Addison-Wesley.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.

Pastor-Sánchez, J., Kontopoulos, E., Saorín, T., Bebis, T., & Darányi, S. (2021). Greek mythology as a knowledge graph: From Chaos to Zeus and beyond. Semantic Web Journal. Retrieved from <https://www.semantic-web-journal.net/system/files/swj2754.pdf>

Lecture 04. (2025). Information Retrieval and Web Search [Lecture slides]. Department of Computer Science and Engineering, Chung-Ang University.

Appendix 1 , Website overview:



1. Pastor-Sánchez, J., Kontopoulos, E., Saorín, T., Bebis, T., & Darányi, S. (2021). Greek Mythology as a Knowledge Graph: From Chaos to Zeus and Beyond. Semantic Web Journal. [↑](#footnote-ref-1)
2. Chung-Ang University. (2025). *Lecture 04: IR Evaluation Measures*. Natural Language Processing and Information Retrieval, p.3. [↑](#footnote-ref-2)
3. Chung-Ang University. (2025). *Lecture 04: IR Evaluation Measures*. Natural Language Processing and Information Retrieval, p.4, 17; Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. [↑](#footnote-ref-3)
4. Chung-Ang University. (2025). *Lecture 04: IR Evaluation Measures*. Natural Language Processing and Information Retrieval, p.4. [↑](#footnote-ref-4)
5. Chung-Ang University. (2025). *Lecture 04: IR Evaluation Measures*. Natural Language Processing and Information Retrieval, p.10; Baeza-Yates, R., & Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology behind Search* (2nd ed.). Addison-Wesley. (Original work published 1999) [↑](#footnote-ref-5)
6. Chung-Ang University. (2025). *Lecture 04: IR Evaluation Measures*. Natural Language Processing and Information Retrieval, p.13, 16. [↑](#footnote-ref-6)
7. Chung-Ang University. (2025). *Lecture 04: IR Evaluation Measures*. Natural Language Processing and Information Retrieval, p.21. [↑](#footnote-ref-7)