A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

Progetto di

Models and Algorithms for Data Visualization

Corso di Laurea in Ingegneria Informatica e Robotica

Curriculum Data Science – A.A. 2024-2025

Dipartimento di Ingegneria

docente

Prof. Giuseppe Liotta

MarvelNet

Explore the connections between heroes and their movie appearances

363433 **Gian Marco Ferri** gianmarco.ferri@studenti.unipg.it

Contents

1 Introduction **2**

2 Requirements Gathering **4**
2.1 End-users 4
2.2 Data Modeling 5
2.2.1 Data Extraction 5
2.2.2 Data Cleaning and Transformation 5
2.2.3 Definition of Dataset Types and Attribute Types 5
2.3 Task Modeling 6
2.3.1 Input Tasks 6
2.3.2 Output Tasks 6

3 Design **7**
3.1 Visualization Design 7
3.1.1 Marks, Channels and Semantics 7
3.1.2 Visual Encoding Idioms 8
3.2 Interaction Design 8
3.2.1 Interaction Operations 8
3.2.2 Interaction Paradigm 8
3.3 Architectural and Technological Choices 9
3.3.1 System Type 9
3.3.2 Technologies 9
3.4 Algorithm Engineering 9
3.4.1 Force-directed Layout Algorithm 9
3.4.2 Drag Behavior 10
3.4.3 Zoom and Pan Algorithms 10

4 Realization **12**
4.1 User Interface 12
4.2 Visual Aids and Live System 13

1 Introduction

The MarvelNet project is an interactive data visualization system designed to explore the connections between heroes of the Marvel Universe and their appearances in movies within the Marvel Universe through a dynamic graph visualization, where users can filter, zoom, drag nodes and acquire details from them.
The main purpose of this visualization system is to provide an engaging and insightful platform for users to understand and analyze the complex web of relationships within the Marvel Universe.

The data that will be visualized includes a list of Marvel characters who appear in various movies, a list of movies that feature these heroes, and relationships between heroes and the movies they appear in, represented as links between nodes.
The dataset is structured as an XML file that contains nodes for heroes and movies, and edges that define the connections between them. Each node has attributes such as id, type (hero or movie), and name, while edges have source and target attributes that indicate the connections.

The main reasons to visualize these data are to provide Marvel enthusiasts, researchers, and new Marvel fans with an intuitive tool to explore and understand the connections within the Marvel Universe and allow them to discover interesting patterns and relationships among heroes and movies, such as which heroes frequently appear together and the centrality of certain characters in the Marvel narrative.
For these exact reasons, this project offers an interactive platform where users can dynamically explore the data, adjust visualization parameters, and filter the information to suit their interests and inquiries.

The following sections are organized as follows:

- **Requirements Gathering**: This section describes the dataset in detail, the end-users of the visualization system, the data modeling processes, and the task modeling approach. It outlines how the data are extracted, cleaned and transformed, as well as the needs and tasks of the end-users.
- **Design**: This section covers the design phase of the project, including visualization design, interaction design, architectural and technological choices, and algorithm engineering. Describes the choices made for visual encoding, interaction operations, system architecture, and the algorithms used to implement visualization.
- **Realization**: This section provides a detailed description of the system user interface, describing its components and offering visual examples of the system in

action.

2 Requirements Gathering

Description of the Dataset

The dataset used in the MarvelNet project is an XML file named `marvel.xml`, taken from the 2019 Graph Drawing Contest and available here. It encapsulates information about characters (heroes) and movies within the Marvel Universe, along with the relationships between them. The dataset is structured into nodes and edges:

- **Nodes**: Represent individual entities, either heroes or movies. Each node has attributes such as `id`, `type`, and `name`.
- **Edges**: Represent connections between nodes, indicating which heroes appear in which movies. Each edge has `source` and `target` attributes that define the relationship between two nodes.

Example Structure of the Dataset

```
<graphml>
  <key attr.name="type" attr.type="string" for="node" id="type" />
  <key attr.name="name" attr.type="string" for="node" id="name" />
  <graph edgedefault="directed" id="">
    <!-- Nodes representing heroes -->
    <node id="h0">
      <data key="type">hero</data>
      <data key="name">Ant-Man</data>
    </node>
    <!-- Nodes representing movies -->
    <node id="m0">
      <data key="type">movie</data>
      <data key="name">Ant-Man</data>
    </node>
    <!-- Edges representing connections -->
    <edge source="h0" target="m0" />
  </graph>
</graphml>
```

This XML structure provides a clear and organized way to represent the relationships within the Marvel Universe, making it suitable for visualization.

2.1 End-users

The end-users of the MarvelNet visualization system include:

- **Marvel Fans**: Enthusiasts who are deeply interested in the Marvel Universe and want to explore the connections between their favorite heroes and movies.
- **New Marvel Fans**: People who have just started watching Marvel movies and want to use this system to understand more about the characters and their relationships.
- **Researchers**: Academics who are studying the interconnectedness of characters in the Marvel Universe, possibly for sociocultural studies or media analysis.

2.2 Data Modeling

2.2.1 Data Extraction

- **Process**: The XML dataset is loaded using D3.js, a JavaScript library to producing dynamic, interactive data visualizations in web browsers. D3.js provides functions to parse XML data, making it easy to extract the necessary information.
- **Implementation**: The data is extracted using D3's `d3.xml` function, which loads the XML file and parses it into a format that can be manipulated programmatically.

2.2.2 Data Cleaning and Transformation

- **Process**: Once the data are extracted, it undergoes a cleaning and transformation process to ensure that it is in a suitable format for visualization.
- **Implementation**:
  - **Parsing Nodes**: Each node in the XML file is parsed to extract attributes such as `id`, `type`, and `name`.
  - **Parsing Edges**: Each edge is parsed to extract the `source` and `target` attributes, defining the connections between nodes.
  - **Transformation**: The parsed data are transformed into a JSON format that D3.js can use to create the visualization.

2.2.3 Definition of Dataset Types and Attribute Types

- **Dataset Type**:
  - **Network** → items (nodes), links (edges) [cardinality: 52 nodes, 152 edges].
- **Attribute types for items**:
  - `id`: A unique identifier for each node, categorical.

  - `type`: Specifies whether the node is a hero or a movie, categorical.
  - `name`: The name of the hero or movie, categorical.

- **Attribute types for links**:
  - `source`: The `id` of the source node (hero).
  - `target`: The `id` of the target node (movie).

2.3 Task Modeling

The tasks are formulated to define user interactions with the visualization system. Each task is represented as an {action, target} pair.

2.3.1 Input Tasks

1. Explore the network of connections to see which heroes appear in which movies.
2. Identify the number of connections for each hero and movie.
3. Locate a specific hero or movie on the network.
4. Enjoy exploring the Marvel Universe network.

2.3.2 Output Tasks

1. Explore the network of connections to see which heroes appear in which movies:
   - **Action**: Explore.
   - **Target**: Network topology.
2. Identify the number of connections for each hero and movie.
   - **Action**: Identify.
   - **Target**: Number of connections (degree) of each node.
3. Locate a specific hero or movie on the network.
   - **Action**: Locate.
   - **Target**: Specific node (hero or movie).
4. Enjoy exploring the Marvel Universe network.
   - **Action**: Enjoy.
   - **Target**: Network.

3 Design

3.1 Visualization Design

3.1.1 Marks, Channels and Semantics

The visualization uses the following marks and channels to represent the data:

Marks

- **Items (Nodes)**: Circle.
- **Links (Edges)**: Connection (segment).

Channels

- **Items (Nodes)**:
  - **Circle Hue**: Categorical attribute `id`.
  - **Circle Size**: Quantitative attribute `degree`.
- **Links (Edges)**:
  - **Position**: Proximity to represent neighboring nodes.

Semantics

- **Nodes**:
  - Each node represents either a hero or a movie. The type of node is indicated by its hue, blue for heroes and green for movies.
  - The size of the node can be dynamic based on the number of connections (degree). This helps to visually emphasize nodes with more connections.
- **Edges**:
  - Each edge represents a connection between a hero and a movie, indicating that the hero appears in that movie.

3.1.2 Visual Encoding Idioms

The visual encoding idiom utilized in the MarvelNet visualization system is the **unconstrained straight-line layout**. This selection is motivated by several factors:

- **Simplicity**: It is easy to implement and understand, making it accessible to users with varying levels of expertise.
- **Clarity**: Provides a clear and direct representation of connections without the need for complex curves or additional visual elements.
- **Scalability**: The layout can handle a large number of nodes and edges without becoming overly cluttered, maintaining readability even in dense networks.

3.2 Interaction Design

3.2.1 Interaction Operations

- **Selection**: Clicking on a node displays detailed information about the hero or movie, such as the name, type, and number of connections.
- **Exploration (Panning)**: Users can pan across the graph by dragging.
- **Zooming**: Users can zoom in and out of the graph using mouse scroll.
- **Abstract Zooming**: When users hover their mouse over a node, the connected edges and related nodes are highlighted, and the names of the affected nodes are shown.
- **Encoding**: A toggle allows users to enable or disable dynamic sizing of nodes based on their degree (number of connections).
- **Filtering**: Users can use checkboxes to filter the nodes displayed in the graph, such as showing only heroes or only movies.

3.2.2 Interaction Paradigm

The interaction paradigm used in the MarvelNet visualization system is the **Full View**, which follows the mantra of Ben Shneiderman: 'Overview first, zoom and filter, then details on demand.'

3.3 Architectural and Technological Choices

3.3.1 System Type

The system being designed is a **web-based interactive visualization system**. It is accessible via a web browser and provides a dynamic and interactive user experience.

3.3.2 Technologies

The technologies used in this project include:

- **HTML**: For structuring the web page.
- **CSS**: For styling the web page and visualization elements.
- **JavaScript**: For implementing the interactive functionality.
- **D3.js**: For creating and manipulating SVG elements and implementing the force-directed graph layout. D3.js is chosen for its powerful data visualization capabilities and flexibility.

3.4 Algorithm Engineering

This subsection describes the algorithms chosen or designed to implement the visualization and interaction idioms for the MarvelNet project.

3.4.1 Force-directed Layout Algorithm

The force-directed layout algorithm positions nodes dynamically based on their connections. It uses forces such as repulsion and attraction to arrange the nodes in a way that minimizes overlap and makes the network structure clear.

Implementation The force-directed layout algorithm is implemented using D3.js's `forceSimulation`. The simulation includes:

- `forceLink`: Defines the links between nodes with a specified distance.
- `forceManyBody`: Applies a repulsive force between nodes to prevent overlap.
- `forceCenter`: Centers the graph in the SVG container.

```
const simulation = d3.forceSimulation(graphData.nodes)
    .force('link', d3.forceLink(graphData.links).id(d => d.id).distance(100))
    .force('charge', d3.forceManyBody().strength(-200))
    .force('center', d3.forceCenter(width / 2, height / 2));
```

3.4.2 Drag Behavior

The drag behavior allows users to interactively manipulate node positions. It is implemented using D3.js's `drag` behavior.

Implementation

```
node.call(d3.drag()
    .on('start', dragstart)
    .on('drag', dragging)
    .on('end', dragend));

function dragstart(event, d) {
    if (!event.active) simulation.alphaTarget(0.3).restart();
    d.fx = d.x;
    d.fy = d.y;
}

function dragging(event, d) {
    d.fx = event.x;
    d.fy = event.y;
}

function dragend(event, d) {
    if (!event.active) simulation.alphaTarget(0);
    d.fx = null;
    d.fy = null;
}
```

3.4.3 Zoom and Pan Algorithms

Zoom and pan algorithms allow users to navigate the network by zooming in and out and panning across the graph. These interactions are implemented using D3.js's `zoom` behavior.

Implementation

```
const zoom = d3.zoom().on('zoom', function(event) {
    svg.attr('transform', event.transform);
});

d3.select('#graph').call(zoom);
```

By integrating these algorithms, the MarvelNet visualization system provides a dynamic and interactive experience, allowing users to explore the Marvel Universe in an engaging and insightful way.

4 Realization

4.1 User Interface

The user interface of the MarvelNet visualization system is designed to be intuitive and user-friendly. The following are the key components of the UI:

Graph Visualization

- **SVG Container**: The main area where the force-directed graph is rendered. It dynamically adjusts to the screen size and allows for zooming and panning.
- **Nodes**: Represent heroes and movies, color-coded for easy identification. Heroes are depicted in steel blue, while movies are depicted in green.
- **Links**: Lines connecting nodes, representing the appearances of heroes in movies.

Interactive Controls

- **Zoom and Pan**: Users can zoom in and out using the mouse wheel and pan across the graph by dragging.
- **Node Details**: Hovering over a node highlights its connections and displays a tooltip with details such as the name, type, and number of connections.
- **Filters**: Checkboxes to show or hide heroes and movies, allowing users to focus on specific parts of the network.
- **Dynamic Sizing**: A toggle to enable or disable dynamic node sizing based on the number of connections.

Information Display

- **Statistics Panel**: Displays general statistics such as the total number of connections, heroes, and movies.
- **Node Information Panel**: Shows detailed information about the selected node, including its name, type, and connections.

4.2 Visual Aids and Live System

The live system and the demonstration video are accessible through the following links:

- MarvelNet Live System
- Demonstration Video

This project can be found at the following link: Github link.

Here are some snapshots of the system in action:

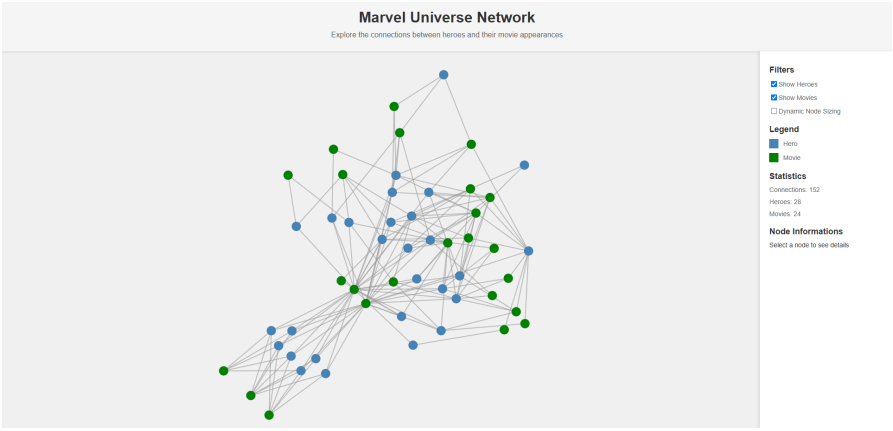

Figure 1: Overview of the MarvelNet user interface.

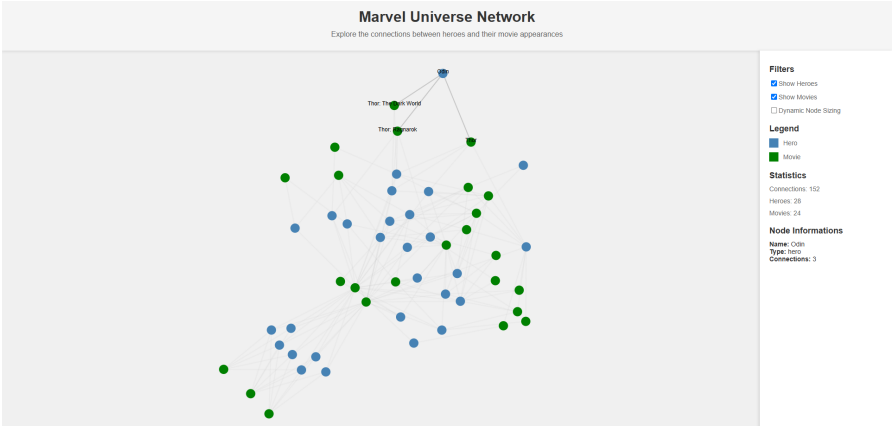

Figure 2: Detailed information displayed upon hovering over a node and clicking it.

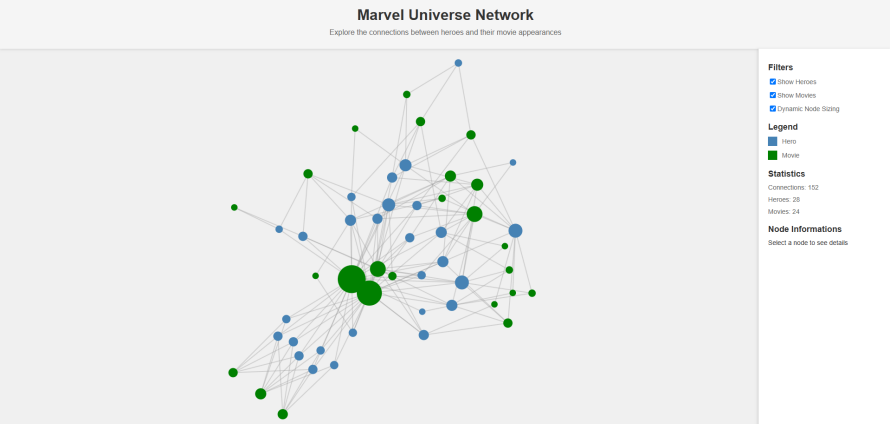

Figure 3: Dynamic sizing of nodes based on their degree.

These resources provide a comprehensive view of how the MarvelNet visualization system works and its interactive features.