Core Features in Depth

---

1. Researcher Discovery

What It Does

Users can search and explore researchers by:

- Fields of study
- Topics of interest
- Publications authored
- Visualize their collaborations and connections.

Implementation

1. **Backend:**
    - Use a **knowledge graph database** (Neo4j or Dgraph).
    - Define nodes for Researcher, Publication, and Field of Study.
    - Relationships:
        - HAS_FIELD: Connect researcher to their fields of study.
        - AUTHORED: Connect researcher to their publications.
        - COLLABORATED_WITH: Connect researchers who co-authored or worked on the same projects.
2. **Frontend:**
    - Build a **search bar** to filter researchers by field, topic, or keyword.
    - Display a **list of researchers** with their profile details.
    - Include a graph visualization (use **Cytoscape.js** or **D3.js**) showing their network (collaborators, publications).
3. **Graph Query Example (Neo4j - Cypher):**

cypher

Copy code

```
MATCH (r:Researcher)-[:AUTHORED]->(p:Publication)
```

```
WHERE r.field_of_study = "Artificial Intelligence"

RETURN r, p
```

User Flow

1. User enters a query in the search bar (e.g., "Artificial Intelligence").
2. Results show a list of researchers in that field, along with:
   - Profile
   - Publications
   - Network of collaborators (visualized).
3. User clicks on a researcher to view detailed relationships.

---

2. Knowledge Graph Visualization

What It Does

Displays connections between:

- Researchers
- Publications
- Datasets
- Research outcomes

Implementation

1. **Backend:**
   - Use a graph query engine (Neo4j, Dgraph) to retrieve relationships dynamically.
   - For large-scale data, use paginated graph queries to prevent overloading.
2. **Frontend:**
   - Visualization library:
     - **Cytoscape.js**: Interactive graph with nodes and edges.
     - **D3.js**: For more customized visualizations.

    - Node types:
        - **Researchers**: Circular nodes.
        - **Publications**: Rectangular nodes.
        - **Datasets**: Hexagonal nodes.
    - Edge types:
        - **Dashed lines** for indirect relationships.
        - **Solid lines** for direct relationships.

3. **Graph Query Example:**

cypher

Copy code

```
MATCH (r:Researcher)-[:AUTHORED]->(p:Publication)-[:USES_DATASET]->(d:Dataset)

RETURN r, p, d
```

User Flow

1. User views a researcher profile.
2. A graph visualization appears, showing:
    - Collaborators
    - Publications
    - Linked datasets.
3. User clicks on a node to expand relationships dynamically.

---

3. Publication Search

What It Does

Allows users to:

- Search publications by keywords, topics, or authors.
- View **AI-generated summaries** for better understanding.

Implementation

1. **Backend:**
    - Index publications using a **search engine** (Elasticsearch or Solr).
    - Integrate AI (e.g., OpenAI GPT or Hugging Face models) for text summarization.
    - Add metadata to each publication:
        - Title, abstract, keywords, author(s), publication date, etc.
2. **Frontend:**
    - Search bar with filters:
        - Keywords
        - Authors
        - Date range
    - Display:
        - Title
        - Abstract (with AI-generated summary).
    - Provide a "View Full Details" button.
3. **Summarization Example (OpenAI API):**
    - Send the publication's abstract to GPT for summarization:

python

Copy code

```python
def summarize_publication(abstract):
  response = openai.Completion.create(
    engine="text-davinci-003",
    prompt=f"Summarize the following abstract: {abstract}",
    max_tokens=150
  )
  return response['choices'][0]['text']
```

User Flow

1. User searches for publications using keywords (e.g., "machine learning").
2. Results display publications with:
   - Title
   - AI-generated summaries.
3. User clicks on a publication to view full details and linked datasets.

---

4. Collaborator Recommendation

What It Does

AI-powered recommendations suggest collaborators based on:

- Similar research areas.
- Co-authorship patterns.
- Shared datasets or research outcomes.

Implementation

1. **Backend:**
   - Use graph-based algorithms (e.g., **PageRank**, **Node2Vec**) to rank potential collaborators.
   - Query the knowledge graph for:
     - Researchers with similar fields.
     - Co-authorship patterns.
     - Researchers using the same datasets.
2. **Frontend:**
   - Create a "Recommended Collaborators" section on each researcher’s profile.
   - Show:
     - Name
     - Field of study

        - Reason for recommendation (e.g., “Shared dataset: XYZ”).

3. **Graph Query Example:**

cypher

Copy code

```
MATCH (r:Researcher)-[:AUTHORED]->(p:Publication)<-[:AUTHORED]-(r2:Researcher)
WHERE r.field_of_study = r2.field_of_study
RETURN r2, COUNT(p) AS shared_publications
ORDER BY shared_publications DESC
```

User Flow

1. User views a researcher profile.
2. Recommendations appear with reasons for the suggestion.
3. User clicks on a recommendation to view the new collaborator’s profile.

---

5. Dataset Linkage

What It Does

Links datasets to publications and researchers for better discovery.

Implementation

1. **Backend:**
    - Create a Dataset node in the knowledge graph.
    - Link datasets to:
        - Publications (via USES_DATASET).
        - Researchers (via CREATED_BY).
2. **Frontend:**
    - Display linked datasets on publication and researcher profiles.
    - Add a "Download Dataset" or "View Details" button for datasets.
3. **Graph Query Example:**

cypher

Copy code

```
MATCH (p:Publication)-[:USES_DATASET]->(d:Dataset)

WHERE p.title = "Deep Learning for Genomics"

RETURN d
```

User Flow

1. User views a publication.
2. Linked datasets appear with details (e.g., dataset name, description).
3. User clicks to view or download the dataset.

---

6. Explainable AI

What It Does

Explains why a collaborator, publication, or dataset is recommended.

Implementation

1. **Backend:**
    - Capture the reasoning path from the knowledge graph.
    - Example: "Recommended Dr. X because they collaborated with Dr. Y on Topic Z."
2. **Frontend:**
    - Display a tooltip or side panel with the explanation.
    - Use a step-by-step reasoning tree for clarity.
3. **Graph Query Example:**

cypher

Copy code

```
MATCH (r:Researcher)-[:COLLABORATED_WITH]->(r2:Researcher)

WHERE r.name = "Dr. A"
```

```
RETURN r2, COLLECT(r2.collaboration_topics) AS topics
```

User Flow

1. User views a recommendation.
2. A tooltip explains the reasoning (e.g., shared dataset, co-authorship).
3. User clicks to explore the relationship further.

User Flow (End-to-End)

1. **Homepage:**
   - Search bar for researchers or publications.
   - Featured researchers and datasets.
2. **Researcher Discovery:**
   - Search results show researchers.
   - User clicks on a researcher to view their profile and connections.
3. **Knowledge Graph Visualization:**
   - Graph shows relationships (collaborators, publications, datasets).
   - User clicks on a node to expand relationships.
4. **Publication Search:**
   - Search results display AI-generated summaries.
   - User views full details of a publication and linked datasets.
5. **Collaborator Recommendation:**
   - Recommendations appear on researcher profiles.
   - Explanations are provided for each suggestion.
6. **Dataset Linkage:**
   - Publications and researcher profiles show linked datasets.
   - User downloads or views datasets.