HACKHPC

@ADMI26

JUNE 22-26, 2026

The Hacking Tribunal!

Future Event Prediction

Team Introduction

Jonathan Wembolua
Graph Analytics & Prediction Lead
Computer Science, *WSSU*

Madison Conley
AI Research & Insight Lead
Computer Science, *MVSU*

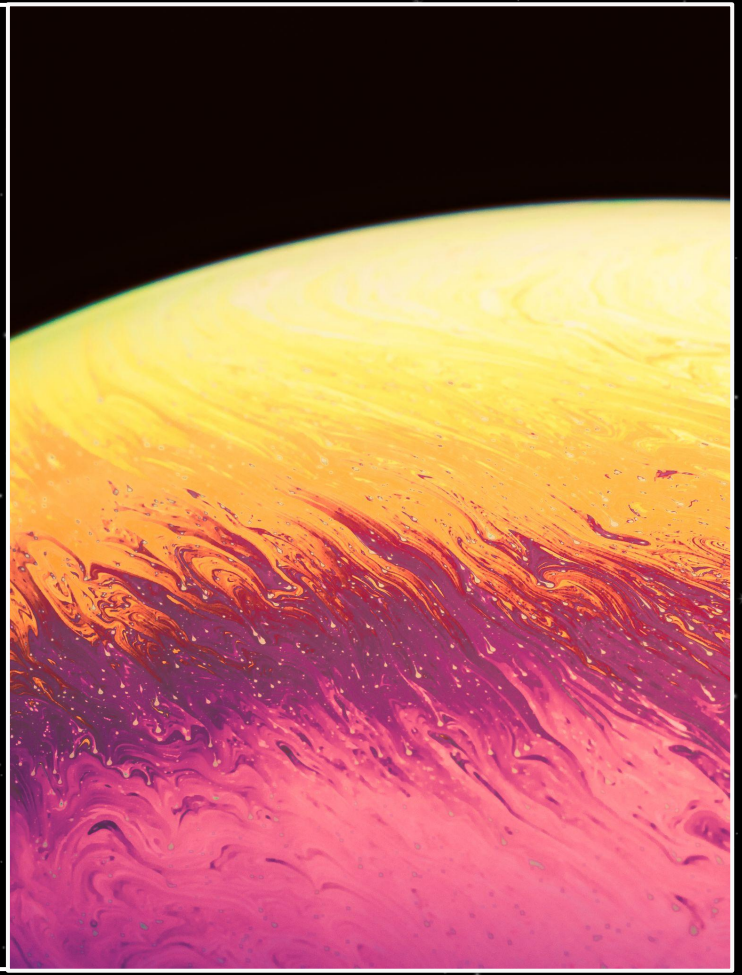
Nadia Rapheal
Technical Program Manager & Integration Lead
Computer Science, *WSSU*

Josiah Johnson
Frontend, Visualization & Presentation Lead
Computer Science, *WSSU*

Usir Jackson
Data & Knowledge Graph Lead
Computer Science, *Claflin*

Research Question:

How can historical data and innovation trends in robotics be analyzed to accurately forecast future technological developments in the field?



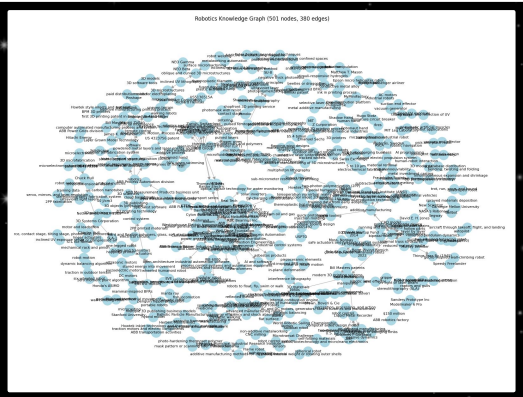
Usir Jackson
Data & Knowledge Graph Lead
Computer Science, Claflin

Milestones:
Biological Seed Dataset Creation

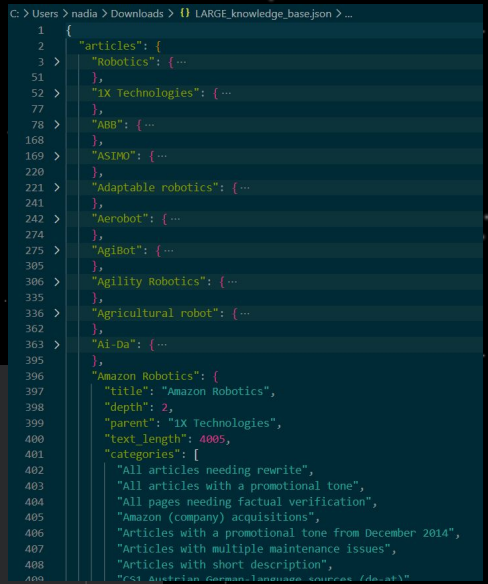
1. Select initial biological seed articles.
2. Extract Wikipedia article links and categories.
3. Generate structured JSON datasets.

Knowledge Graph Construction

1. Create nodes from biological concepts.
2. Create relationships between connected concepts.
3. Generate graph visualization output.



< visualization output



^ result of wiki crawl

v getting article text and links

```python
# wikipedia api helpers
def get_article_text(title: str) -> str:
    """Fetch the plain-text extract of a Wikipedia article."""
    url = (
        "https://en.wikipedia.org/w/api.php"
        "?action=query"
        "&prop=extracts"
        "&explaintext=1"
        "&format=json"
        f"&titles={requests.utils.quote(title)}"
    )
    try:
        r = requests.get(url, headers=HEADERS, timeout=15)
        r.raise_for_status()
        page = next(iter(r.json()["query"]["pages"].values()))
        return page.get("extract", "")
    except Exception as e:
        print(f"  ⚠  Text fetch failed for '{title}': {e}")
        return ""


def get_article_links(title: str) -> list:
    """Return Wikipedia article titles linked from the given article (main namespace only)."""
    url = (
        "https://en.wikipedia.org/w/api.php"
        "?action=query"
        "&prop=links"
        "&pllimit=max"
```

AI Resources:

Anthropic Claude Sonnet 4.6

Libraries:

- requests – Wikipedia API requests
- json – Knowledge base serialization
- time – Rate limiting during crawling
- re – Historical year extraction
- anthropic – Claude API integration
- networkx – Knowledge graph construction and graph analysis

Data Sources:

- Wikipedia API
- Internal article links
- Article categories

Platforms:

- Google Colab
- Anthropic API
- Wikipedia API

Nadia Rapheal
Technical Program Manager & Integration Lead
Computer Science, WSSU

Technical Program Manager & Integration Lead

Coordinate project milestones and team deliverables.
Integrate project components into a unified pipeline.
Validate data flow across all system modules.
Manage testing, debugging, and system integration.
Maintain project documentation and technical reports.
Prepare the final project for demonstration and delivery.

THE HACKING TRIBUNAL

5 - DAY HACKATHON GANTT CHART

Duration: 5 Days | Monday → Friday | TEAM: Nadia, Josiah, Usir, Jonathan, Madison

	ID	Task	Owner	Monday - 1	Tuesday - 2	Wednesday - 3	Thursday - 4	Friday - 5
1	1	Repo Setup + README init	Josiah, Nadia					
	2	Colab + API Config System	Nadia					
2	3	Robotics Seed Article Selection	Usir					
	4	WikiAPI Recursive Scraper	Usir					
	5	Token Parser: Concept-Relation	Nadia					
3	6	Knowledge Graph Node Schema Design	Usir					
	7	Edge Construction + Wiki Link Mapping	Usir					
	8	NetworkX Graph Storage + Serialization	Josiah					
4	9	Historical Snapshot Pipeline	Jonathan					
	10	Interactive Graph UI Base	Josiah					
5	11	LLM Engine Init	Madison					
	12	Temporal Evolution Analysis	Jonathan					
	13	Centrality Metrics Computation	Jonathan					
	14	Convergence Detection System	Nadia					
6	15	RAG Forecasting Engine	Madison					
	16	Trend Prediction Output	Madison					
7	17	UI Creation and Design	Josiah					
	18	UI Expansion (Time, Cluster, Score)	Josiah					
8	19	Dashboard and Website	Josiah, Nadia					
	20	Final README + Poster + Demo Prep	ALL TEAM					

Madison Conley
AI Research & Insight Lead
Computer Science, MVSU

Milestones:
AI Research & Insight

Generate AI predictions for robotics concepts.
Analyze historical predecessors of emerging technologies.
Forecast future technology trajectories using an LLM.
Evaluate technological impact across related fields.
Enrich the knowledge base with AI-generated insights.
Produce explainable predictions grounded in historical data.

```
msg = client.messages.create(
    model="claude-haiku-4-5-20251001",
    max_tokens=1000,
    messages=[{"role": "user", "content": pred_prompt}]
)

raw = msg.content[0].text.strip()

# DEBUG: Print what we got back
print(f"  [DEBUG] Predecessors response length: {len(raw)}")
if len(raw) > 0:
    print(f"  [DEBUG] First 100 chars: {raw[:100]}")
else:
    print(f"  [DEBUG] EMPTY RESPONSE")
    return None

predictions["predecessors"] = clean_and_parse_json(raw)
print(f"  ✓ Predecessors analyzed")
time.sleep(1)
```

< creating calls to anthropic

v predictions and context

```
def predict_node(node, G, kb, analysis_type="all", curated_set=None):
    """
    Generate predictions for a single node.
    analysis_type: "predecessors", "forward", "backward", or "all"
    curated_set: Set of curated forecast concepts (optional)
    """
    predictions = {}

    # Skip if node not in graph (no relationships)
    if node not in G:
        print(f"  ∅ Node not in graph (no relationships) — skipping")
        return None

    # Get context
    predecessors = get_predecessors(node, G, depth=2)
    successors = get_successors(node, G, depth=2)
    context = get_node_context(node, kb, G)

    # Add priority flag if in curated set
    if curated_set and node.lower() in curated_set:
        predictions["is_curated_forecast_priority"] = True

    try:
        # PREDECESSORS ANALYSIS
        if analysis_type in ["predecessors", "all"]:
            pred_prompt = PREDECESSOR_PROMPT.format(
                node=node,
```

```
# prediction prompt templates


PREDECESSOR_PROMPT = """\
You are a robotics historian analyzing concept evolution.

Given the emerging robotics concept: "{node}"

Historical predecessors that likely led to it: {predecessors}

Based on robotics history, what key technologies or concepts from the past
enabled or directly led to "{node}"? Be specific about HOW they connect.

Return ONLY valid JSON (no markdown, no preamble):

{{
  "direct_predecessors": ["concept 1", "concept 2"],
  "enabling_technology": "description of how they enabled this",
  "timeline_origin": "approximate decade when these predecessors were relevant"
}}
"""
```

^ prediction prompt

Programming Language:

Python

Libraries and Frameworks:

- Anthropic
- NetworkX
- JSON
- time (API rate limiting)
- collections (batch processing)

Data Produced:

- AI Generated historical predecessor predictions
- Enriched robotics knowledge base

Platforms:

- Google Colab
- Anthropic API

AI Resources:

- Anthropic Claude Haiku (LLM for historical analysis and future prediction)

Jonathan Wembolua
Graph Analytics & Prediction Lead
Computer Science, WSSU

Milestones:

Historical Evolution Analysis
Analyze historical growth trends across robotics technologies.
Calculate forecasting metrics from graph and temporal data.
Rank emerging concepts by forecast score and confidence.

Convergence Detection System
Detect technologies converging across robotics subfields.
Analyze graph connectivity to identify innovation hotspots.
Generate prediction datasets for future trend analysis.

```python
# Calculate component scores
growth = year_growth_score(years_list)
recency = recency_score(years_list)
connectivity = connectivity_score(node, G)
subfield = emerging_subfield_score(node, kb, article_by_concept)

# PREDICTION FORMULA
# Emerging concept = high growth + high recency + low connectivity + high subfield presence
# Weight: growth and subfield are strongest signals
forecast_score = (
    (growth * 0.35) +           # rapid year mention growth
    (subfield * 0.35) +         # appearing in multiple subfields
    (recency * 0.20) +          # high recency
    ((1 - connectivity) * 0.10) # not yet fully integrated
)
```

^ prediction formula

```python
for pred in predictions:
    print(f"\n#{pred['rank']} — {pred['concept'].upper()}")
    print(f"  Confidence: {pred['confidence']}")
    print(f"  Timeline: {pred['timeline']}")
    print(f"  Forecast Score: {pred['forecast_score']}")
    print(f"  Summary: {pred['summary']}")

print("\n" + "=" * 80)
print("Ready for dashboard integration!")
```

```
================================================================================
ROBOTICS FORECASTING REPORT — 2026-2036
================================================================================

#1 — AUTONOMOUS AGRICULTURAL VEHICLES
  Confidence: High
  Timeline: 2026-2036
  Forecast Score: 0.993
  Summary: Autonomous agricultural vehicles shows rapidly growing men
```

< forecasting report output

Programming Language:

Python

Libraries:

- json – Loading and processing the robotics knowledge base
- networkx – Knowledge graph construction, graph traversal, and network analysis
- collections – Efficient aggregation of concept statistics and relationships
- statistics – Calculation of historical trend metrics and averages
- cytoscape – Compatibility with the visualizer

Forecasting Methods:

- Historical year growth analysis
- Recency scoring
- Graph connectivity analysis
- Confidence score generation
- Forecast ranking and prediction filtering

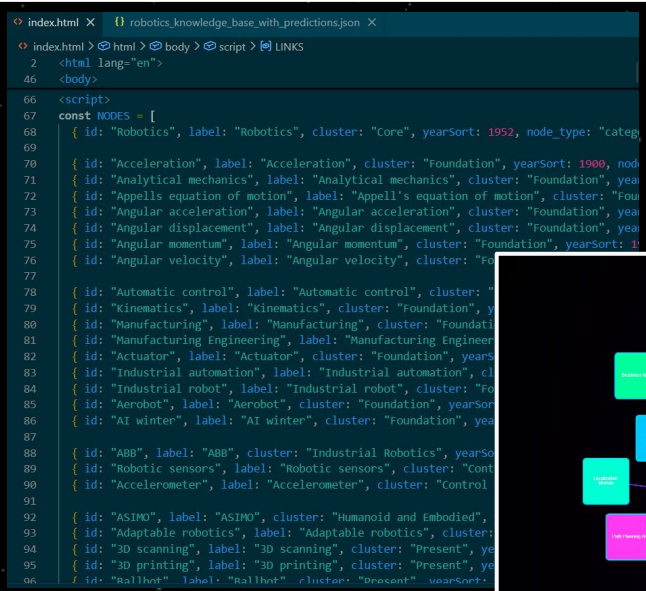
Platforms:

- Google Colab

Josiah Johnson
Frontend, Visualization & Presentation Lead
Computer Science, WSSU

Milestones:
Visualization and User Experience

Develop the interactive robotics knowledge graph interface.
Visualize concept relationships and prediction results.
Integrate forecasting data into dashboard components.
Design presentation graphics and project visuals.
Organize project documentation and deliverables.
Demonstrate project functionality during presentations.



< json nodes for the graph visualization

^ interface

Programming Languages:

- JavaScript
- Python

Libraries and Frameworks:

- React
- Cytoscape and Cytoscape-fcose
- React Router

Development Tools:

- VS Code
- Git
- GitHub

Platforms:

- Google Colab
- GitHub Pages

AI Resources:

- Copilot (code assistance, debugging)
- Claude (interface development, and implementation support)