Knowledge Graph Service Registration

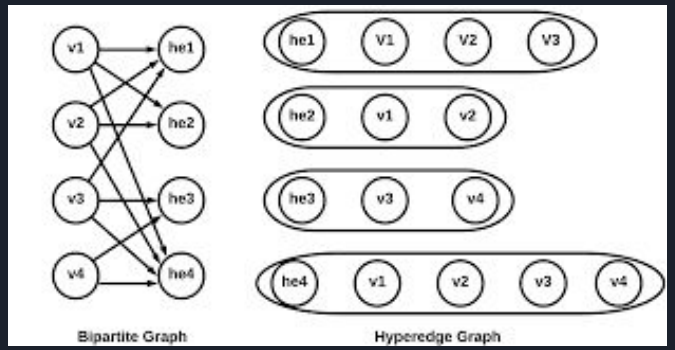
By Daniel Olsen and Daniel Weedon
CS 4624 Multimedia, Hypertext, and Information Access
Professor Edward A. Fox
Virginia Tech, Blacksburg VA, 24061
April 22, 2021

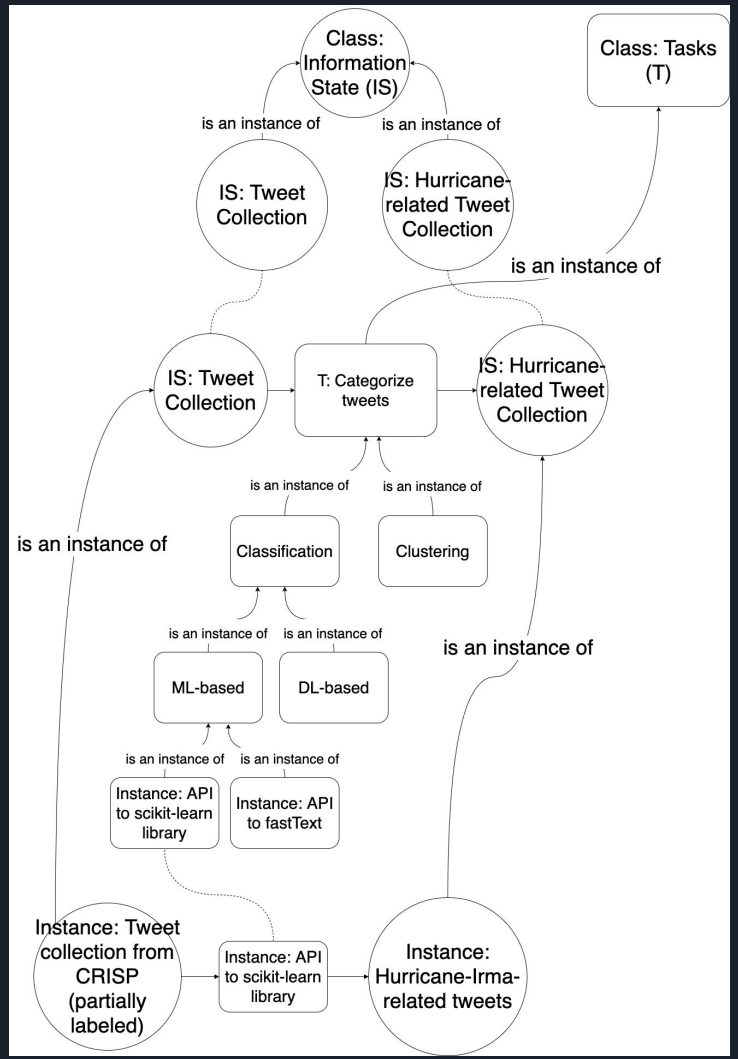
Outline

- What is a Knowledge Graph?
- Project Description
- Knowledge Graph Backend
- Python API
- Backend Challenges
- Front-end Visualization
- Front-end Goal
- Front-end Challenges
- Acknowledgements/References

What is a Knowledge Graph?

- Graph based data model
- Visualizes entities and their connects to each other
- Can use hyperedges
  - Many entities connected with 1 edge
  - Not 1 to 1





N. Imam, S. M. S. Hasan, and R. Kannan, “A Scalable Parallel Hypergraph Generator (HyGen),” http://www.mlgworkshop.org/, 2020. [Online]. Available: http://www.mlgworkshop.org/2020/papers/MLG2020_paper_5.pdf. [Accessed: 28-Apr-2021].

Project Description

This project was a ***two-component*** project with the goal of building a Knowledge Graph database that represents how user information goals are connected to one another.

This Knowledge Graph is connected to a workflow management system that allows developers to register their services and visualize the graph.

Knowledge Graph Backend

The knowledge graph database was built using Grakn

Grakn is a tool used to build knowledge graphs through Entity-Relationship (ER) models and can utilize:

- Type hierarchies
- Hyper-relations (hyper-edges)

Python API

A Python API was made using Flask 1.1.2 and is used to communicate with the database. A client can use this API to to access the knowledge graph and manipulate data within through numerous operations:

- Create and remove files (nodes)
- Create and remove tasks (edges)
- Search for files/tasks/both based on multiple attributes
- Find the path between 2 nodes

Backend Challenges

- Learning Grakn (and Grakl)
- Developing an efficient schema
- Learning Flask
- API Remake

Front-end Visualization

- HyperNetX library
- Representation
  - Nodes = Files
  - Edges = Service/Task
  - Combined = Workflow

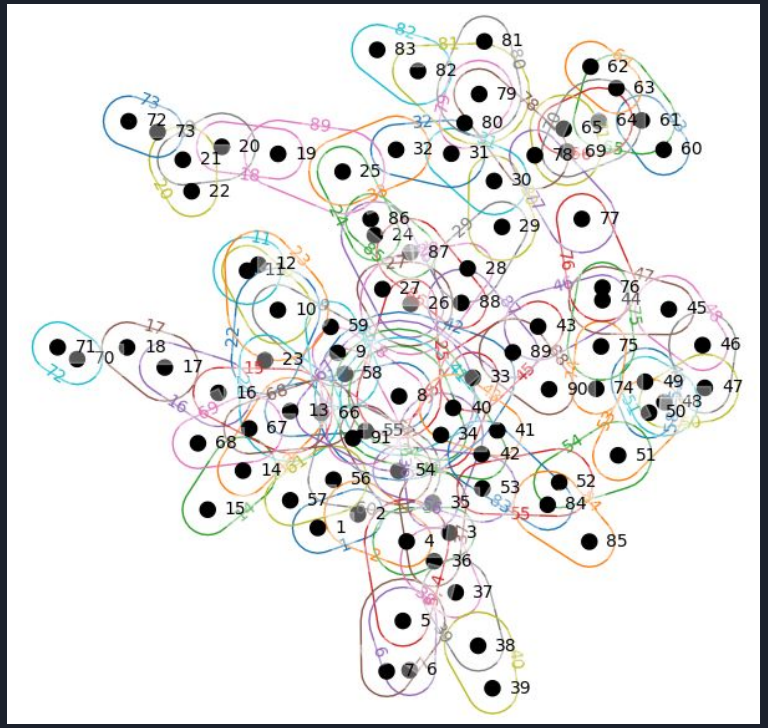


Figure 1: Entire “Rubber-band” hypergraph w/sample data

fileId	name
1	Cleantext from HBase (RDD of tweets)
2	Cleaned Lemmatized Tweets
3	Set of Events Classes
4	Tweet Training Data
5	Word Feature Representation
6	Predicted Class of the Given Tweet
7	Associated Probabilities of Each Class

Figure 2: Node data

taskId	name	inputId	outputId
1	lemmatize	1	2
2	Manually Classify Tweets	2	4
3	Manually Classify Tweets	3	4
4	Feature Selection	4	5
5	Feature Selection	4	5
6	Tweet Classification	5	6
7	Tweet Classification	5	7

Figure 3: Part of Edges data

Front-end Goals

This part of the project is meant to give developers a way to view the knowledge graph and provide them the ability to add their containerized service to the knowledge graph.

- Visualize the Knowledge Graph
- Connect to backend via Python API
- Integrate with prior work
- ~~Service Registration form~~ (Another VT student's project)

Front-end Challenges

- Attempting to use prior work
- Learning React
- Researching visualization methods
- Finding the right library for hypergraphs
- Learning HyperNetX library
- Getting Python and React to work together

Acknowledgements/References

Acknowledgements

- Prashant Chandrasekar

References/Resources

- https://vtechworks.lib.vt.edu/handle/10919/19081
- http://hdl.handle.net/10919/98239
- https://grakn.ai/grakn-core
- https://github.com/graknlabs
- https://vtechworks.lib.vt.edu/handle/10919/101526
- https://git.cs.vt.edu/cs-5604-fall-2020/fe/team-fe-repo/-/tree/knowledgeGraph
- https://github.com/pnnl/HyperNetX