Pycon . Tanzania · December 2022 · Zanzibar

PYTHON CONFERENCE TANZANIA 2022

PREDICTING FAKE NEWS USING GCN

SUZA - ZANZIBAR

Zephania Reuben

December 7, 2022

Meet Zephania Reuben [Nsoma]

- Software Developer - Beem Africa [2021 - 2022]
  - Python & Data Science
- Speaker/Facilitator - Python & Artificial Intelligence
  - PyCon 2019, 2020, 2021 by Python Community Tanzania
  - IndabaX Tanzania 2021 by Deep Learning Indaba
  - Data Science Training 2021, 2022 by Vema Academy
  - DevFestDar 2021,2022 by GDG Dar es Salaam
  - Teens in AI [Tanzania] 2021, 2022 by Ujuzi Forum
  - Advanced AI Training 2022 by AI4D Lab - Anglophone Africa
  - EnhanceMind AI Conference 2022 by CameLabs
- Organizing Committee
  - AI and Life 2019 by TeleSoftAI
  - UmojaHack 2020 by Zindi Africa
  - PyCon Tanzania 2021 by Python Community Tanzania
  - IndabaX Tanzania 2022 by Deep Learning Indaba
  - EnhanceMind AI Conference 2022 by CameLabs

Who is Zephania Now?

Aspiring AI Researcher & Python Programmer

Introduction

- Social media becomes the central way for people to obtain and utilise news, due to its rapidness and inexpensive value of data distribution.
- Though, such features of social media platforms also present it a root cause of fake news distribution, causing adverse consequences on both people and culture.
- Hence, detecting fake news has become a significant research interest for bringing feasible real time solutions to the problem.

Modeling Fake News Detector Using GCN

Outline

Observation

There are complex systems all around us:

- Society is a collection of 7+ billion individuals.
- Communication systems link electronic devices.
- Information and knowledge are organized and linked.
- Interactions between thousands of genes/proteins regulate life.
- Our thoughts are hidden in the connections between billions of neurons in our brain.

Observations

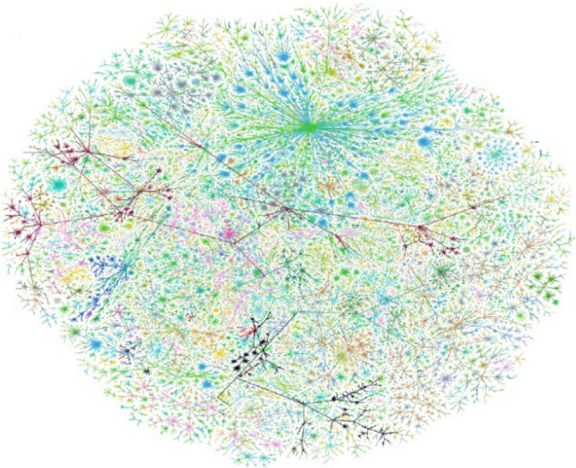
- What do these systems have in common?
- How can we represent them?

Observatiosn

There are complex systems all around us:

- Society is a collection of 7+ billion individuals.
- Communication systems link electronic devices.
- Information and knowledge are organized and linked.
- Interactions between thousands of genes/proteins regulate life.
- Our thoughts are hidden in the connections between billions of neurons in our brain.

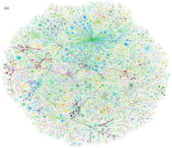
The Network



The Network

Behind many systems there is an intricate wiring diagram, a network, that defines the interactions between the components

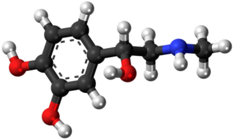
Where Can we Find Networks?



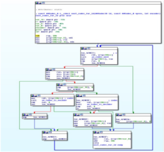
Internet

Social networks

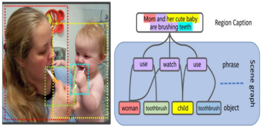
Information retrieval



Biomedical graphs



Program graphs



Scene graphs

The Network

"We will never be able to model and predict these systems unless we understand the networks behind them!"

Jure Leskovec

Why Networks? Why Now?

- Universal language for describing complex data
  - Networks from science, nature, and technology are more similar than one would expect
- Shared vocabulary between fields
  - Computer Science, Social Science, Physics, Economics, Statistics, Biology
- Data availability computational challenges
  - Web/mobile, bio, health, and medical
- Impact!
  - Social networking, Social media, Drug design

Classical ML Tasks on Graphs

- Community detection
  - Predict a type of a given node
- Link Prediction
  - Predict whether two nodes are linked e.g Content recommendation
- Community Detection
  - Identify densely linked clusters of nodes
- Graph similarity
  - How similar are two (sub)graphs

pycon tanzania

Structure of Networks

A network is a collection of objects where some pairs of objects are connected by links. What is the structure of the network?

Graph or Network

- Network often refers to real systems

  Web, Social network, Metabolic network

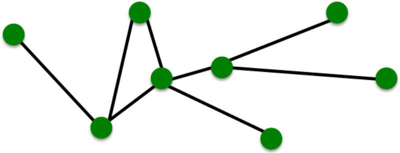
  Language: Network, node, link

- Graph is a mathematical representation of a network

  Web graph, Social graph (a Facebook term)

  Language: Graph, vertex, edge

Components of a Network

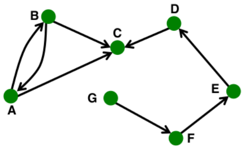


- Objects: nodes, vertices N
- Interactions: links, edges E
- System: network, graph G(N,E)

Directed Vs Undirected Graphs

Directed

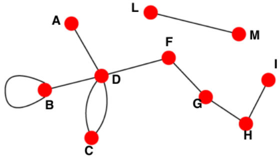
- Links: directed (arcs)



- Examples:
  - Phone calls
  - Following on Twitter

Udirected

- Links: undirected (symmetrical, reciprocal)



- Examples:
  - Collaborations
  - Friendship on Facebook

Getting the Intuition of Graph Neural Networks

Getting the Intuition of Graph Neural Networks

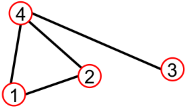
- Nowadays, a lot of information are represented in graphs.
- For example
  - Google's Knowledge Graph that helps with the Search Engine Optimization (SEO)
  - Chemical molecular structure
  - Document citation networks (document A has cited document B) and
  - Social media networks (who is connected to who?)

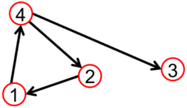
Getting the Intuition of Graph Neural Networks

- I encountered GNN first time in 2020 while I was working in one of my client's work(research) and they caught my attention.

Representing Graphs: Adacency Matrix

- Adjacency matrices are able to represent the existence of edges the connect the node pairs through the value in the matrices.



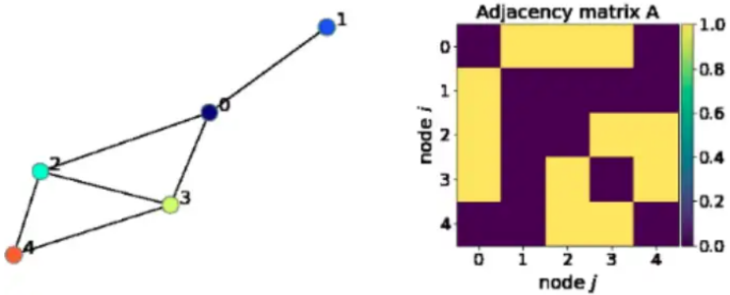


$A_{ij} = 1$ if there is a link from node i to node j

$A_{ij} = 0$ otherwise

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Adacency Matrix (A)



Adacency Matrix in NumPy

```
import numpy as np

A = np.matrix([
    [0, 1, 1, 1, 0],
    [1, 0, 0, 0, 0],
    [1, 0, 0, 1, 1],
    [1, 0, 1, 0, 1],
    [0, 0, 1, 1, 0]],
    dtype=float
)
```

Node Attributes Matrix (X)

- Unlike adjacency matrices that models the relationship between nodes, this matrix represents the features or attributes of each node.

Document 1

"I like pizza."

Document 2

"I hate chicken porridge."

Corpus: {i, like, hate, pizza, chicken, porridge}
Size of Corpus (*F*) = 6

	Document 1	Document 2
I	1	1
like	1	0
hate	0	1
pizza	1	0
chicken	0	1
porridge	0	1

The shape of Node attributes matrix ***X*** is 2 x 6.

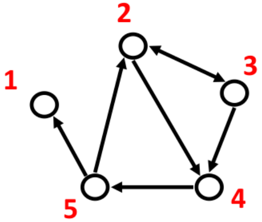
Node Attributes Matrix (X)

```
X = np.matrix([
            [i, -i]
            for i in range(A.shape[0])
        ], dtype=float)
```

Edge Attributes Matrix (E)

Representing a graph as a list of edges:

- (2, 3)
- (2, 4)
- (3, 2)
- (3, 4)
- (4, 5)
- (5, 2)
- (5, 1)



Edge Attributes

- Sometimes, edges can have its own attributes too, just like nodes.
  - Weight (e.g. frequency of communication)
  - Ranking (best friend, second best friend. . .)
  - Type (friend, relative, co-worker)
  - Sign (Friend vs. Foe, Trust vs. Distrust)
  - Properties depending on the structure of the rest of the graph: number of common friends

Complete Graph Initialization

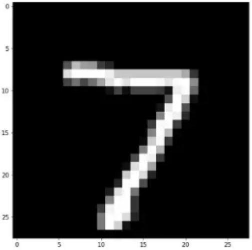
```
import networkx as nx

#Initialize the graph
G = nx.Graph(name='G')
#Create nodes
#In this example, the graph will consist of 6 nodes.
#Each node is assigned node feature which corresponds to the node name
for i in range(6):
    G.add_node(i, name=i)
#Define the edges and the edges to the graph
edges = [(0,1),(0,2),(1,2),(0,3),(3,4),(3,5),(4,5)]
G.add_edges_from(edges)
#See graph info
print('Graph Info:\n', nx.info(G))
#Inspect the node features
print('\nGraph Nodes: ', G.nodes.data())
#Plot the graph
nx.draw(G, with_labels=True, font_weight='bold')
plt.show()
#Get the Adjacency Matrix (A) and Node Features Matrix (X) as numpy array
A = np.array(nx.attr_matrix(G, node_attr='name')[0])
X = np.array(nx.attr_matrix(G, node_attr='name')[1])
X = np.expand_dims(X,axis=1)
```

Graph Neural Networks vs Convolutional Neural Networks

- The classic method to perform image classification is using Convolutional Neural Networks.
- Images of digits are represented in pixels and the CNN would run sliding kernels (or filters) across the images, and the model subsequently learn important features by looking at the adjacent pixels.

Image as a Graph

- Each node represents each pixel.
- Node feature represents the pixel value.
- Edge feature represents the Euclidean distance between each pixel.
- The closer 2 pixels are to each other, the larger the edge values.



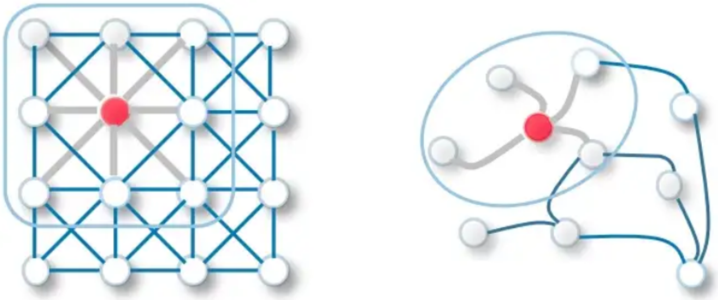
CNN vs GCN

- In CNNs, this node connection are uniform among all pixels.
- In the case where the node connections are dynamic.
- CNN will reach its limitation and that is where we need GNN to come into play.

Simply

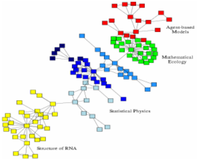
- The major difference between CNNs and GNNs is that CNNs are specially built to operate on regular (Euclidean) structured data, while GNNs are the generalized version of CNNs where the numbers of nodes connections vary and the nodes are unordered (irregular or non-Euclidean structured data).

Regular vs Irregular Data

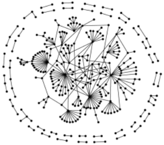


Example of Non-Euclidian Domains

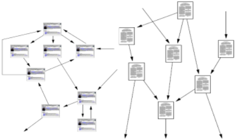
Social networks



Economic networks

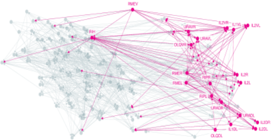


Communication graphs



Information networks:
Web & citations

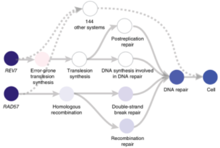
Internet



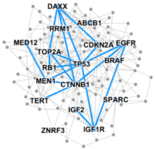
Networks of neurons

Example of Non-Euclidian Domains

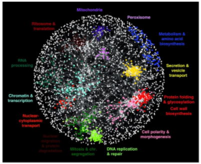
Patient networks



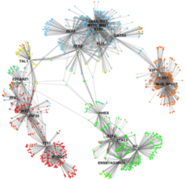
Hierarchies of cell systems



Disease pathways



Genetic interaction networks



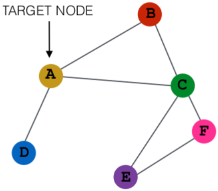
Gene co-expression networks

Cell-cell similarity networks

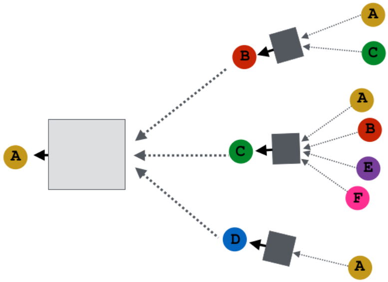
How GCN learn

Show me your friend(s) and I will tell you who you are!

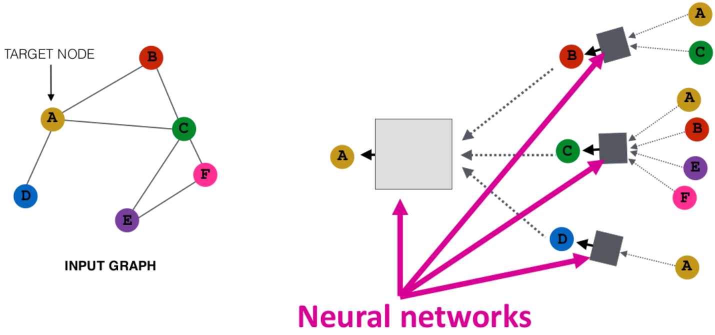
Information Aggregation in GCN



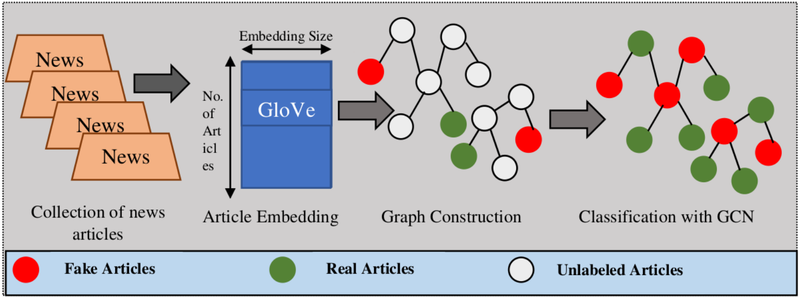
INPUT GRAPH



Information Aggregation in GCN



Hands on guide



Hands on

Refer to this notebook here

Prerequisites

- Good background in:
  - Algorithms and Graph theory
  - Probability and statistics
  - Linear algebra
- Programming Tools
  - You should be able to write non-trivial programs (in Python)
  - Other tools include NetworkX, iGraph, PyTorch Geometry(PyG), Spektral, Jraph built on top of Jax etc

References

- Jure Leskovec
  - Machine Learning with Graphs
- Thomas Kipf

  Graph Convolutional Networks

Thank You!, Twitter: @nsomazr

Pycon . Tanzania

December 2022 . Zanzibar

pycon tanzania