RANDOM WALK APPROACHES TO MACHINE LEARNING OF NETWORKS

PROJECT PRESENTATION

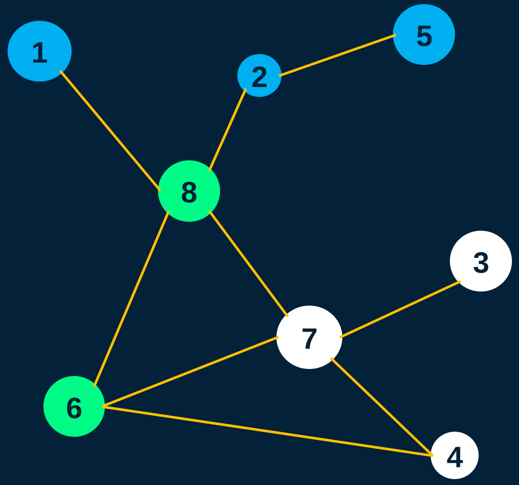
Michigan State University

TABLE OF CONTENTS

01

NETWORK EMBEDDING

Understanding Graphs for Representation Learning

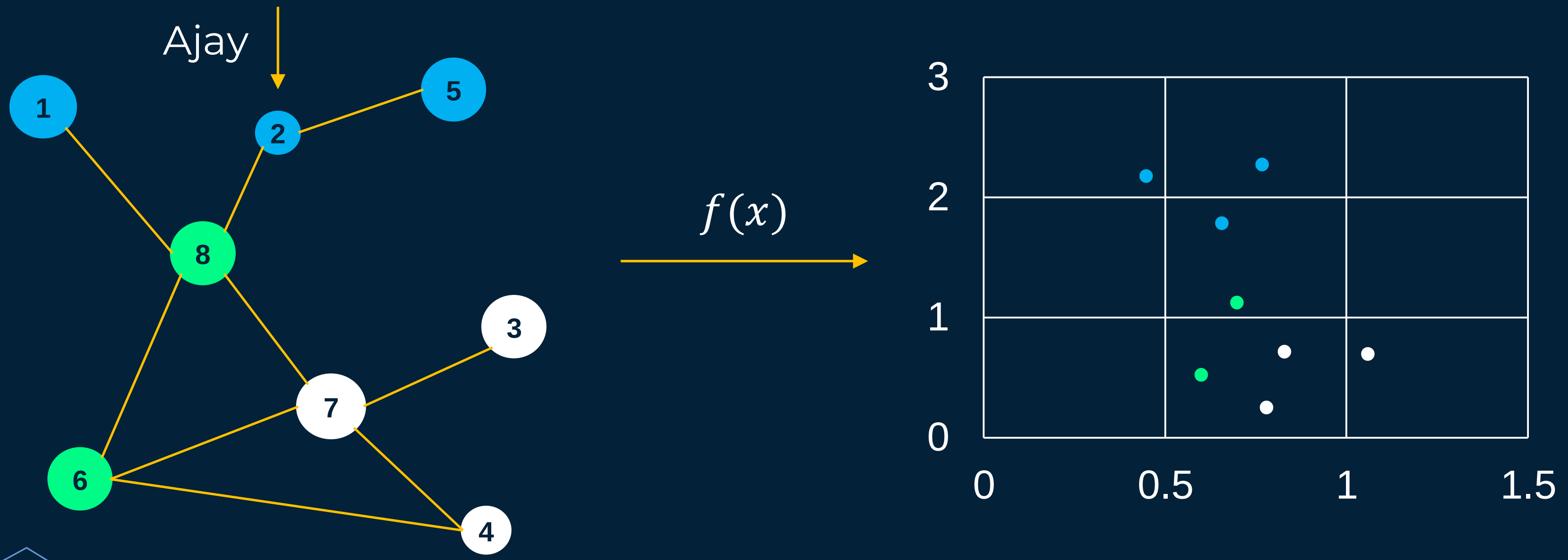


- Visualizes relationships between the data
- Consists of:
  1. Nodes: capture data as vectors
  2. Edges: connect related nodes
- Representation Learning extracts hidden features from the graph for complex analysis.

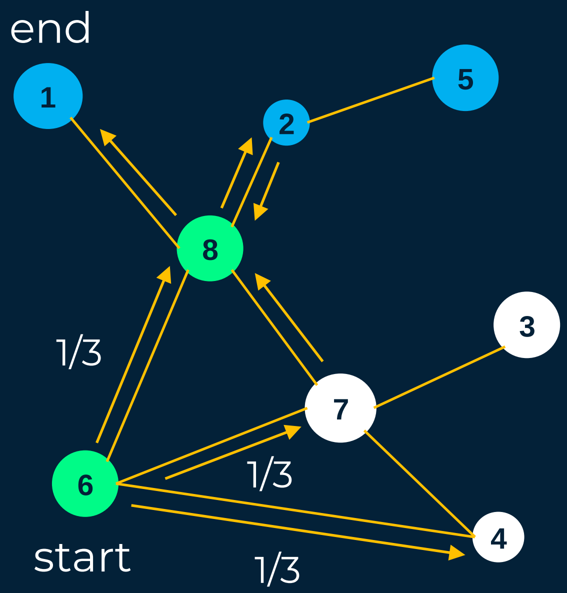
e.g. link prediction, node classification, community detection.

Understanding Graphs for Representation Learning

$$[0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$



Analyzing Graphs with Random Walk



Steps = 5

- Choose starting point and steps
- Determine probabilities based on neighbor nodes
- Collect data from walks

SkipGrams and their relation to DeepWalk

- NLP model: words in sentence context
- Applied on the paths to maximize the probability of observing a node's neighborhood
- Nodes with similar neighborhood share similar embeddings
- The objective function of DeepWalk is the following cross entropy:

$$min_y - logP(\{v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}\}|y_i)$$

W is the window size which restricts the
size the random walk context

The meaning and transformation of the SkipGram formula

Looking back at the previous formula,
SkipGram removes the ordering constraint

In the end, it will be transformed into:

$$min_y - log \sum -w \leq j \leq w \, P(v_{i+j}|y_i)$$

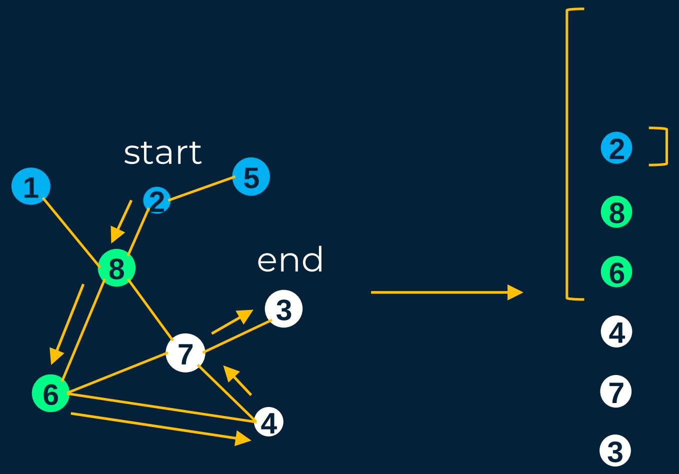
Conditional probability $P(v_{i+j}|y_i)$ defined using the softmax function:

$$P\left(v_{i+j}\middle|y_j\right) = \frac{\exp(y_{i+j}^T y_i)}{\sum_{k=1}^{|v|} \exp(y_k^T y_i)}$$

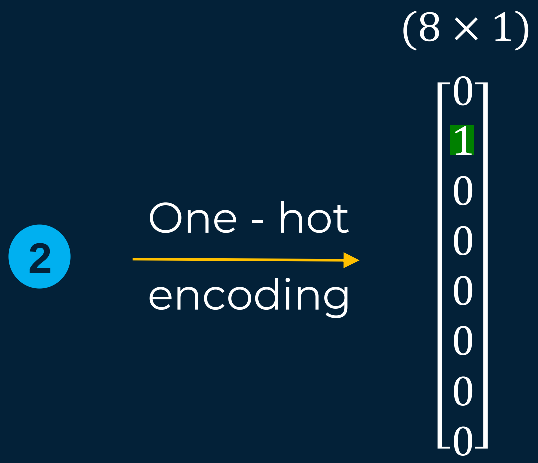
02
DeepWalk

How DeepWalk Works

Step 1: Random Walk



Step 2: One-hot encoding



$one - hot\ encoded\ vector$

$walk\ length = 5$

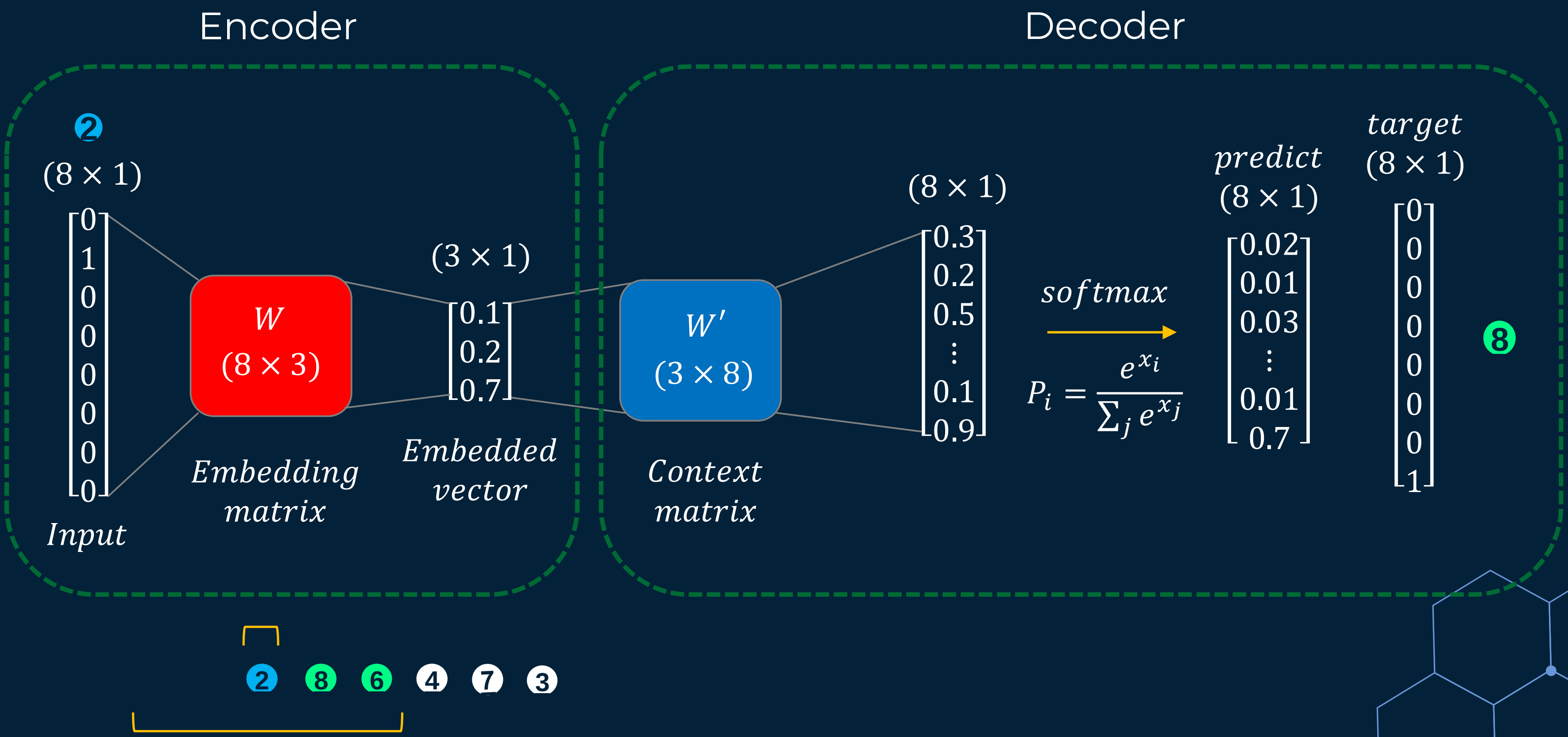
$walk\ per\ vertices$

$window\ size = 2$

$pointer$

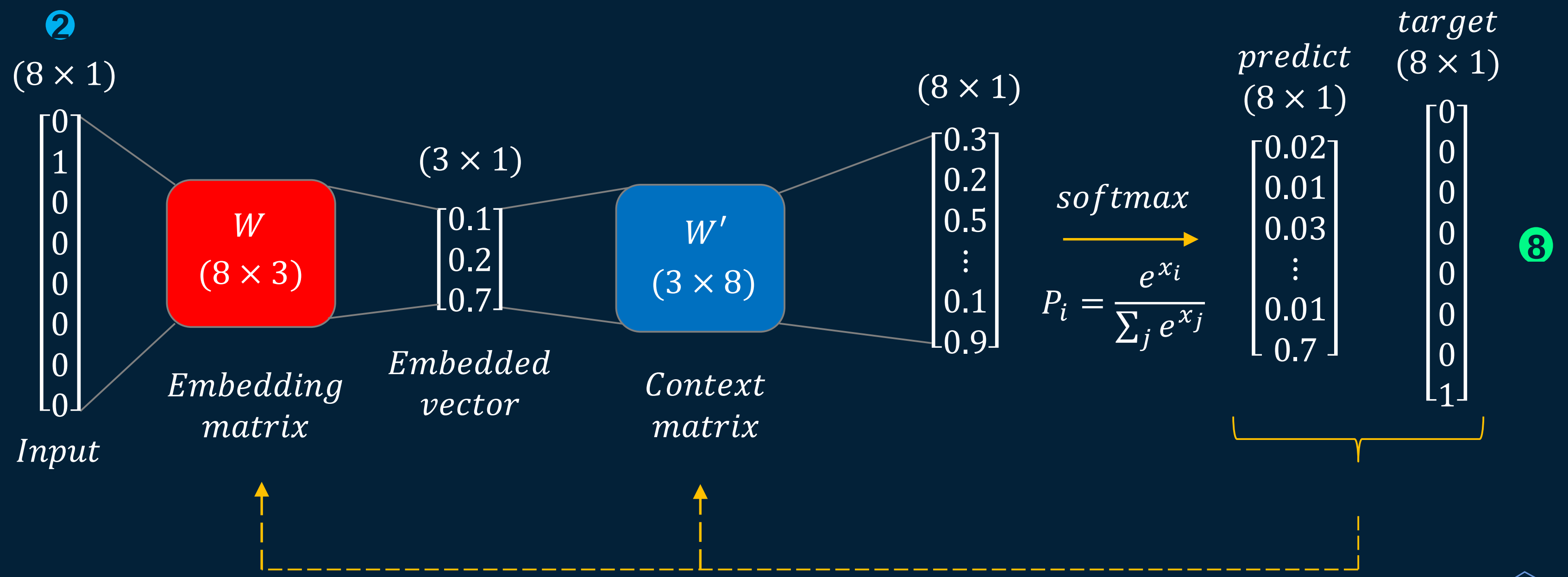
How DeepWalk Works

Step 3: Implement Skip-gram model



How DeepWalk Works

Step 3: Implement Skip-gram model



Cross entropy loss: $\min_y - \log P(\{v_{i-w}, \dots, v_{i-1}, v_{i+1}, \dots, v_{i+w}\} | y_i)$

Why SkipGrams for DeepWalk?

SkipGrams

- Application: Texts
- Context: Relationships of words in a sentence

The quick brown fox jumps over the lazy dog

DeepWalk

- Application: Graphs
- Context: Relationships of nodes in Random Walks

Teacher A – Student B
Student B – Student C

Similarities:

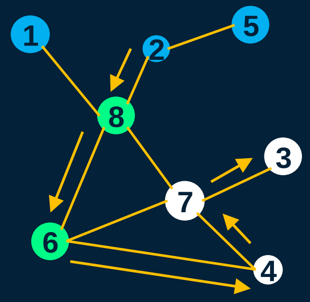
DeepWalk adapts SkipGram from natural language processing to graph data.

03
Node2Vec

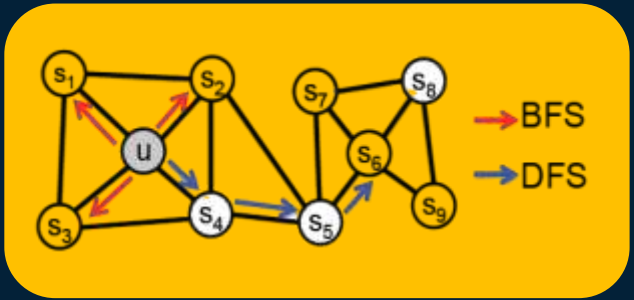
How Node2Vec Works

- Node2Vec is almost similar to DeepWalk
- Both methods use:
  1. Random walks
  2. Skipgram model
- Node2Vec have different walk algorithm to collect nodes
- Node2Vec either explores data in wide range or far range

DeepWalk



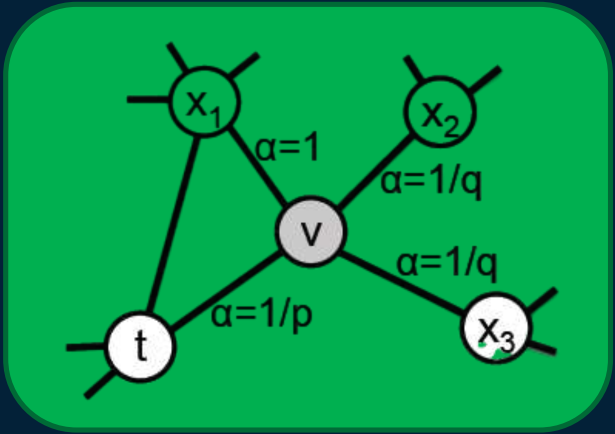
Node2Vec



How Node2Vec Works

- Current walk: t → v
- Determine probability of v → x
- Parameter $\alpha_{pq}(t,x)$ is introduced, where:

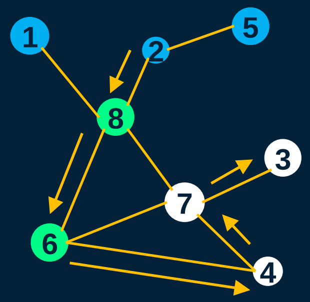
$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & if\ d_{tx} = 0 \\ 1 & if\ d_{tx} = 1 \\ \frac{1}{q} & if\ d_{tx} = 2 \end{cases}$$



- p controls likelihood of revisiting a node
- q controls likelihood of walking further or locally

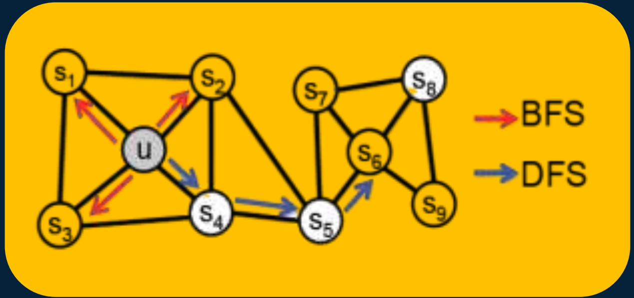
DeepWalk & Node2Vec

DeepWalk



- Explores data randomly
- Low computational cost
- Ignores important data

Node2Vec

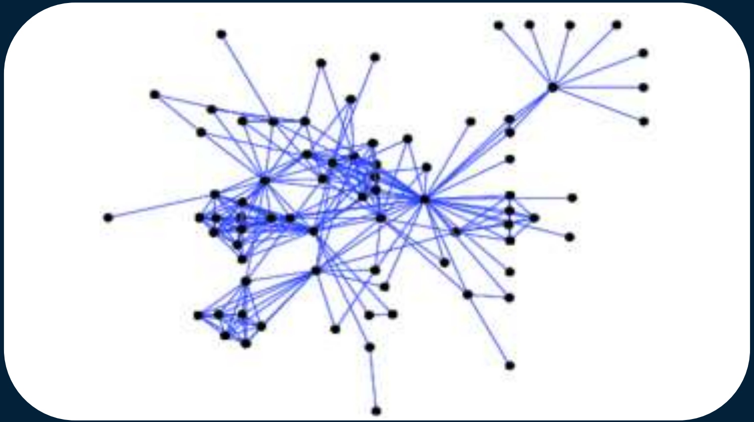


- Flexibility
- Higher computational cost
- Analyze relationships better

Effect of p and q in Node2Vec

Step 1: Prepare dataset (Les Misérable Network)

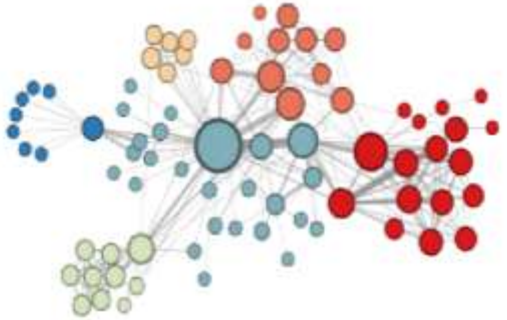
- Contains 77 nodes, 254 edges
- Nodes: characters from Les Misérable novel
- Edges: relationships between characters



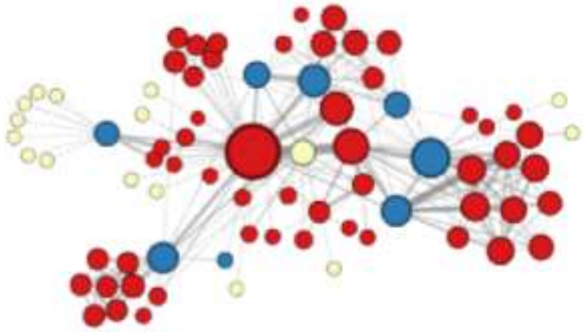
Effect of p and q in Node2Vec

Step 2: Adjust different p and q

Set $p = 1, q = 0.5$



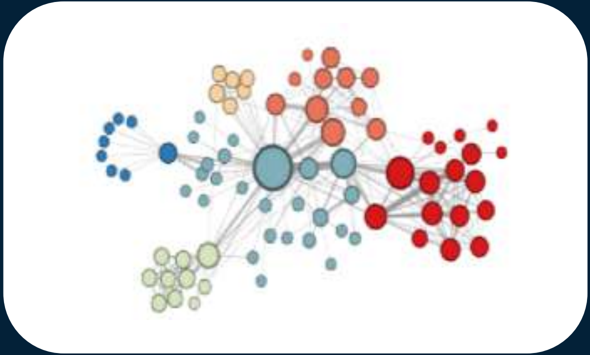
Set $p = 1, q = 2$



Effect of p and q in Node2Vec

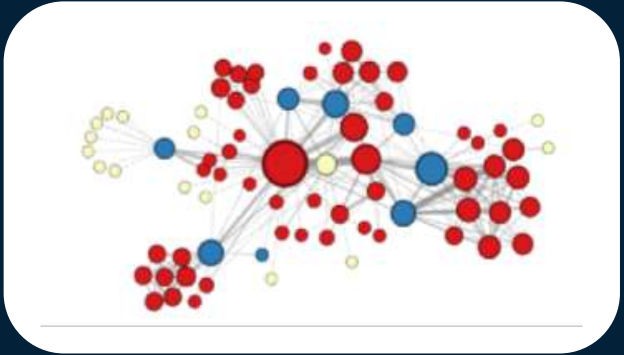
Step 3: Analyze results

Set $p = 1, q = 0.5$



$p \gg q$: deep exploration
Homophily community

Set $p = 1, q = 2$

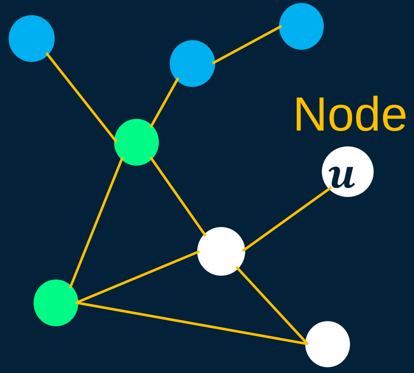


$p \ll q$: broad exploration
Structural equivalence

04

EXPERIMENTS AND EVALUATION

Experiments



$f: u \to \mathbb{R}^d$

Vector

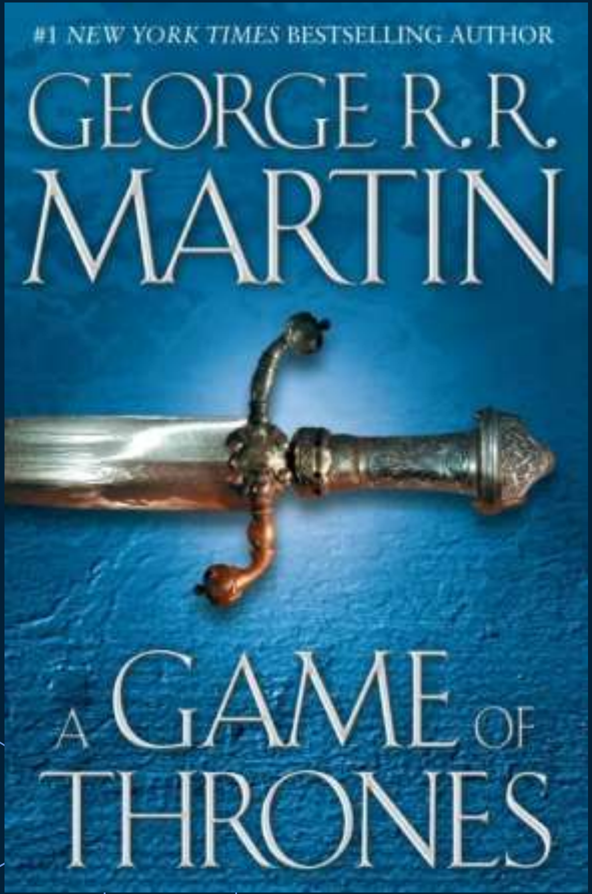
$\mathbb{R}^d$

Sufficient semantic information

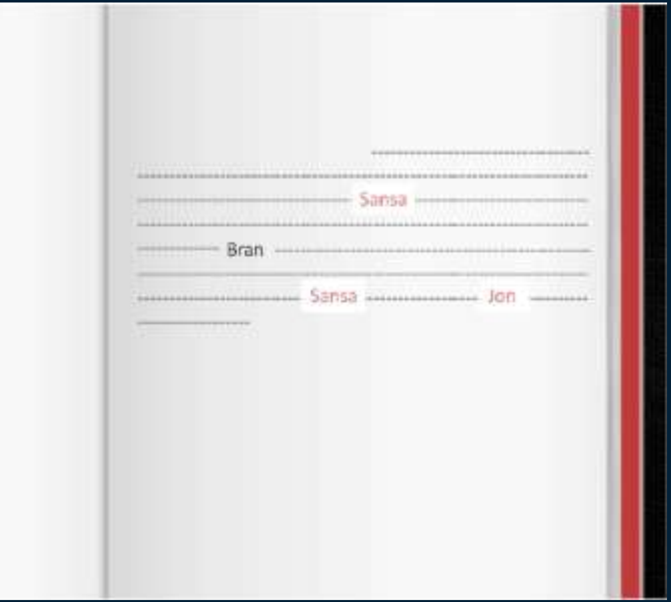
Downstream Tasks

- Node classification
- Link prediction
- Graph classification
- Anomalous node detection
- Clustering
  - Homophily community
  - structural equivalence
- ….

Game of Thrones



From Book to Network



Data

Source	Target	weight
Addam-Marbrand	Jaime-Lannister	3
Addam-Marbrand	Tywin-Lannister	6
Aegon-I-Targaryen	Daenerys-Targaryen	5
Aegon-I-Targaryen	Eddard-Stark	4
...	...	...

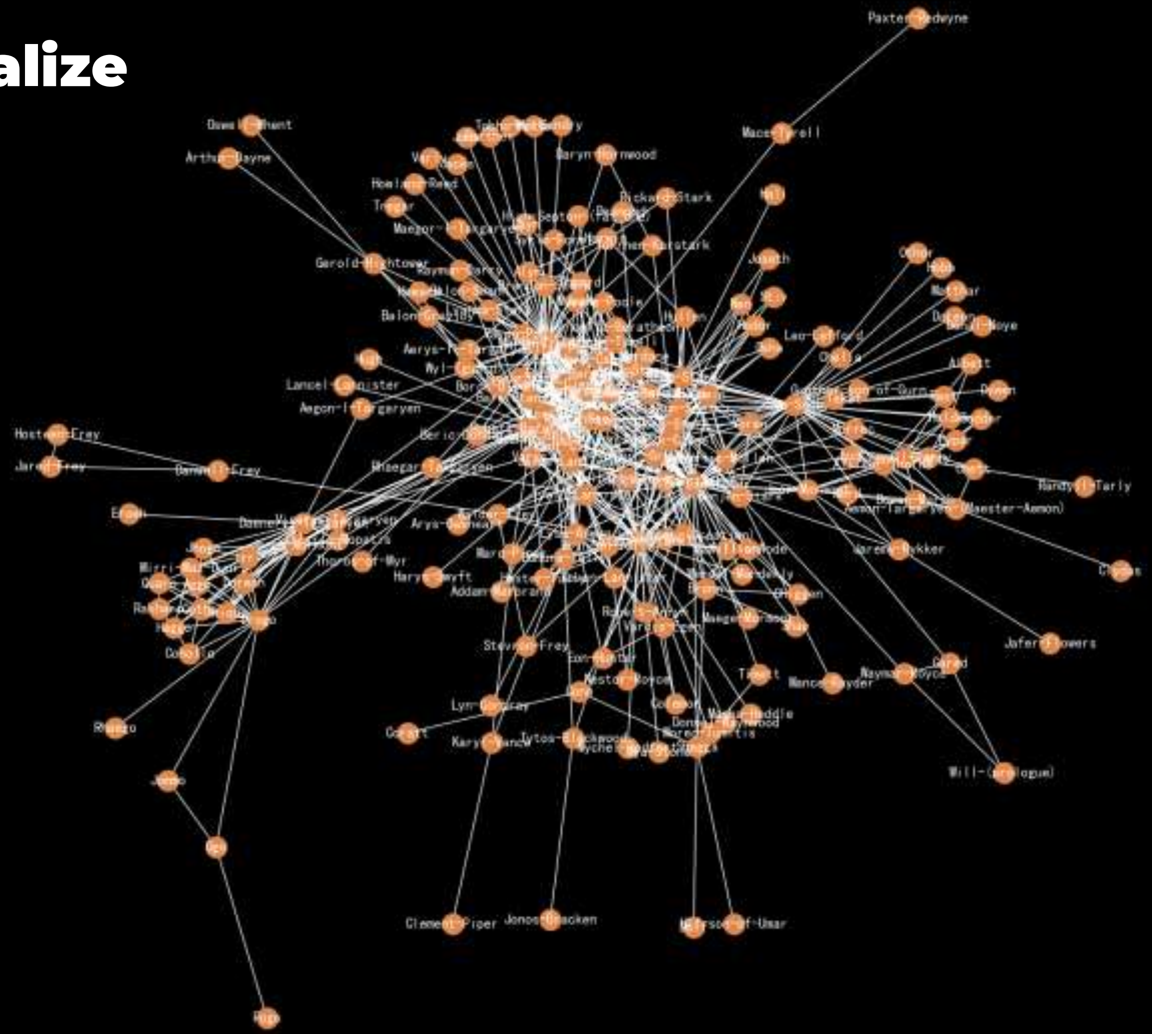
Number of nodes: 187
Number of edges: 684

Link two characters each time their names appear within 15 words.

Pre-processing-visualize

Import unweighted graph

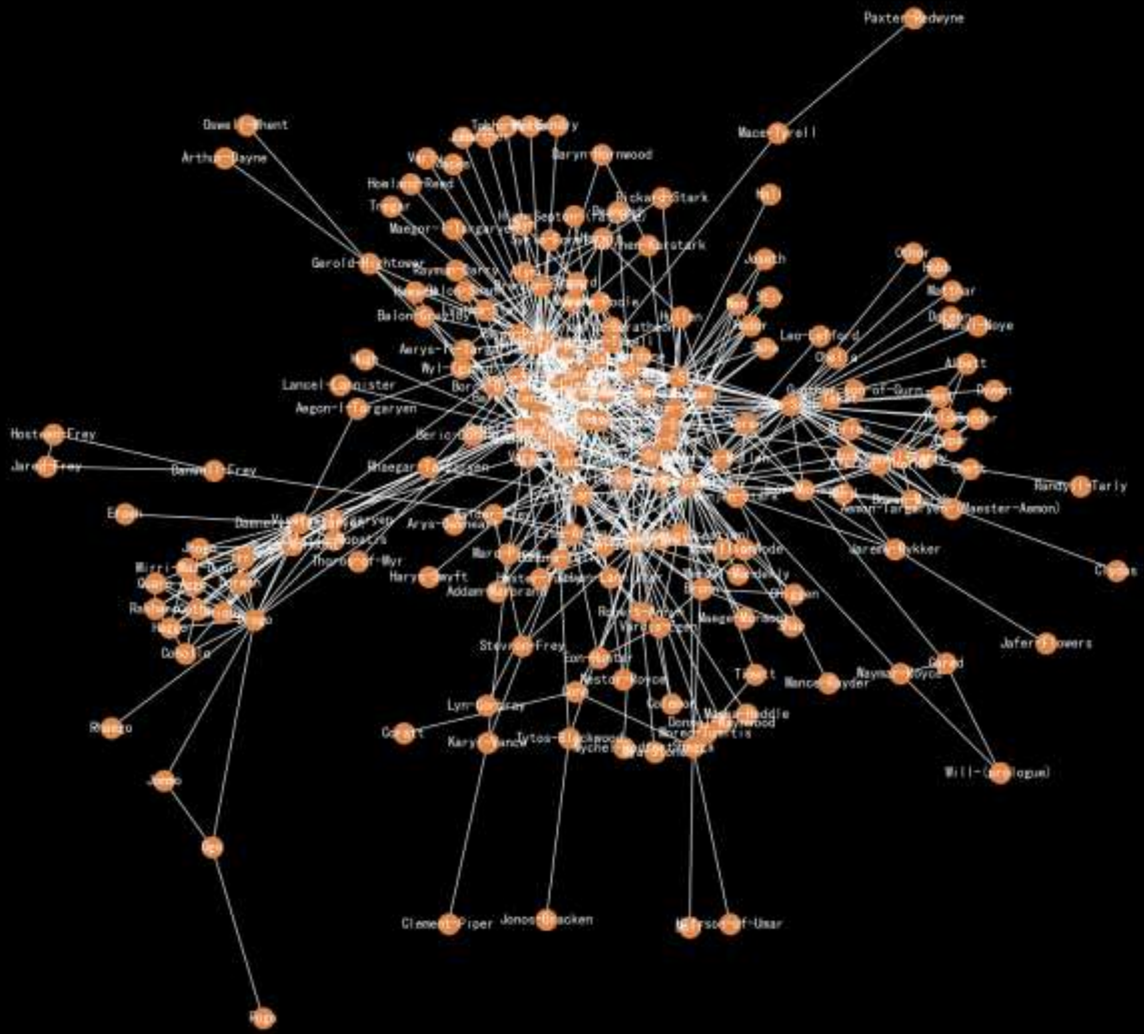
nx.spring_layout(G)



Processing-DFS

Node2Vec
(G,
dimensions=32,
p=5,
q=0.5,
walk_length=10,
num_walks=600,
workers=4)

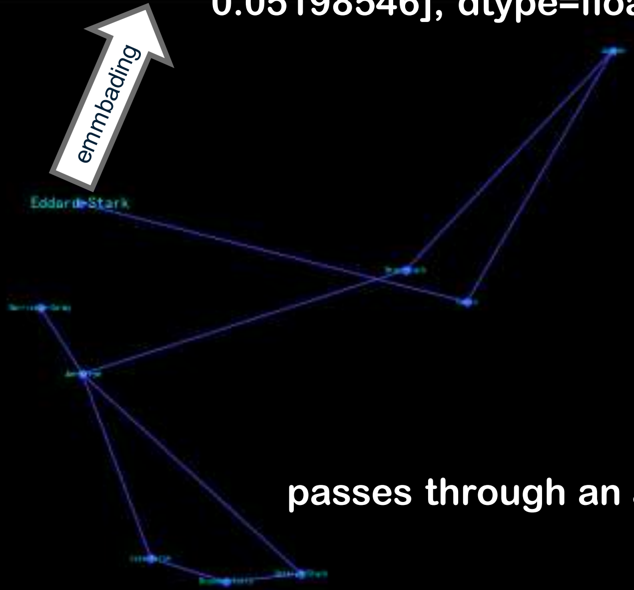
DFS depth-first search

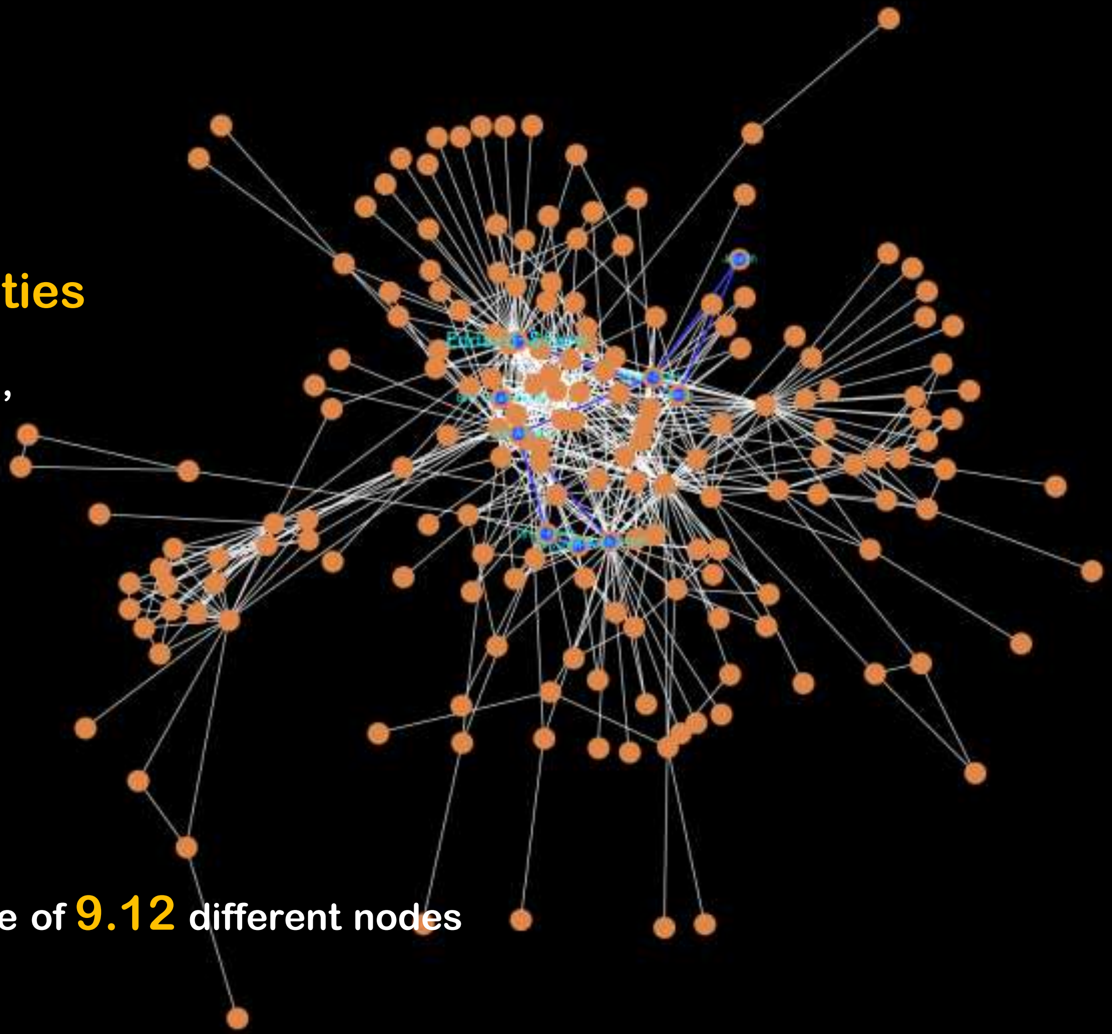


Processing-DFS

DFS depth-first search,
find homogeneous communities

array([-0.36179334, -0.11370167, ... ,
0.05198546], dtype=float32)



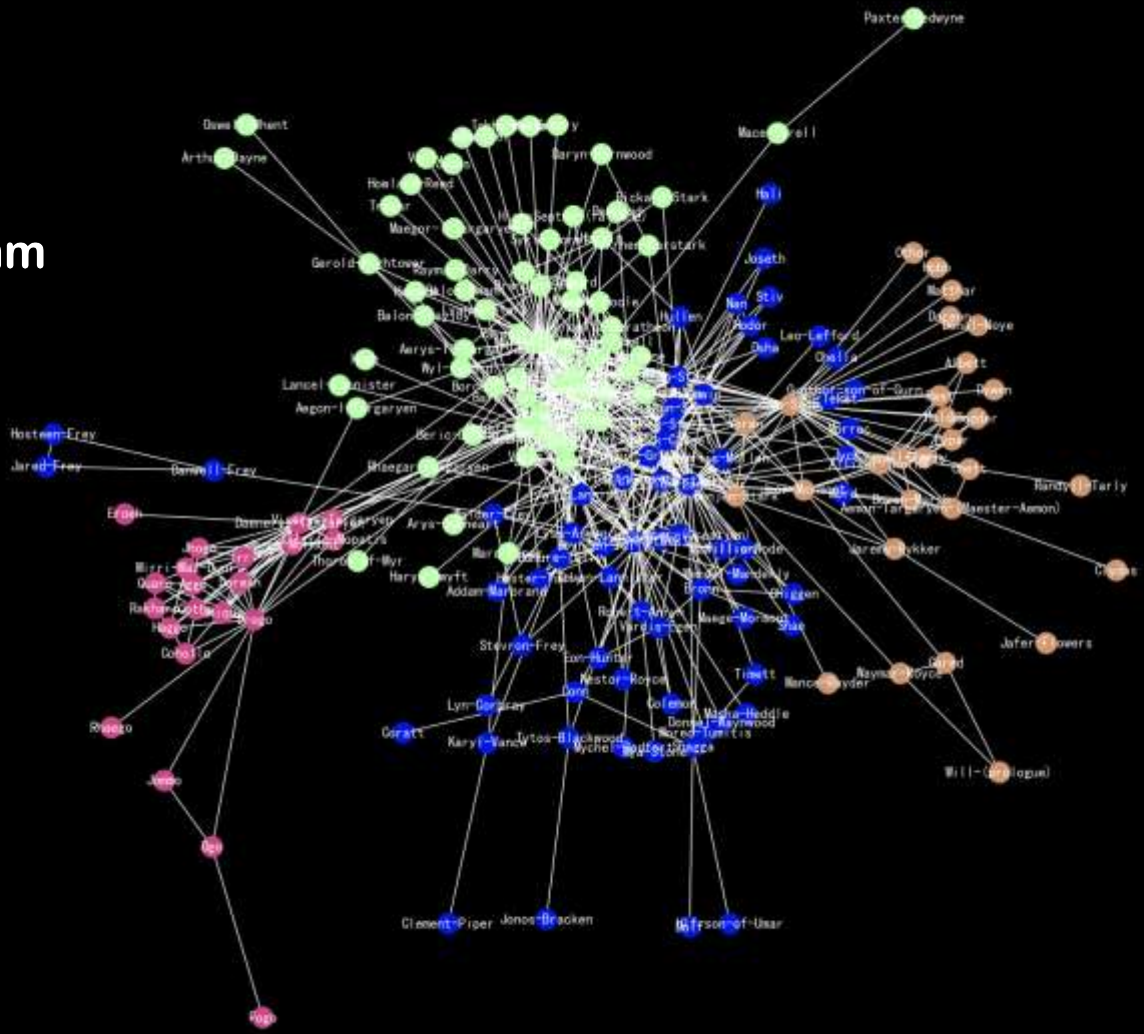


passes through an average of 9.12 different nodes

Processing

Kmeans clustering algorithm

n_clusters=4

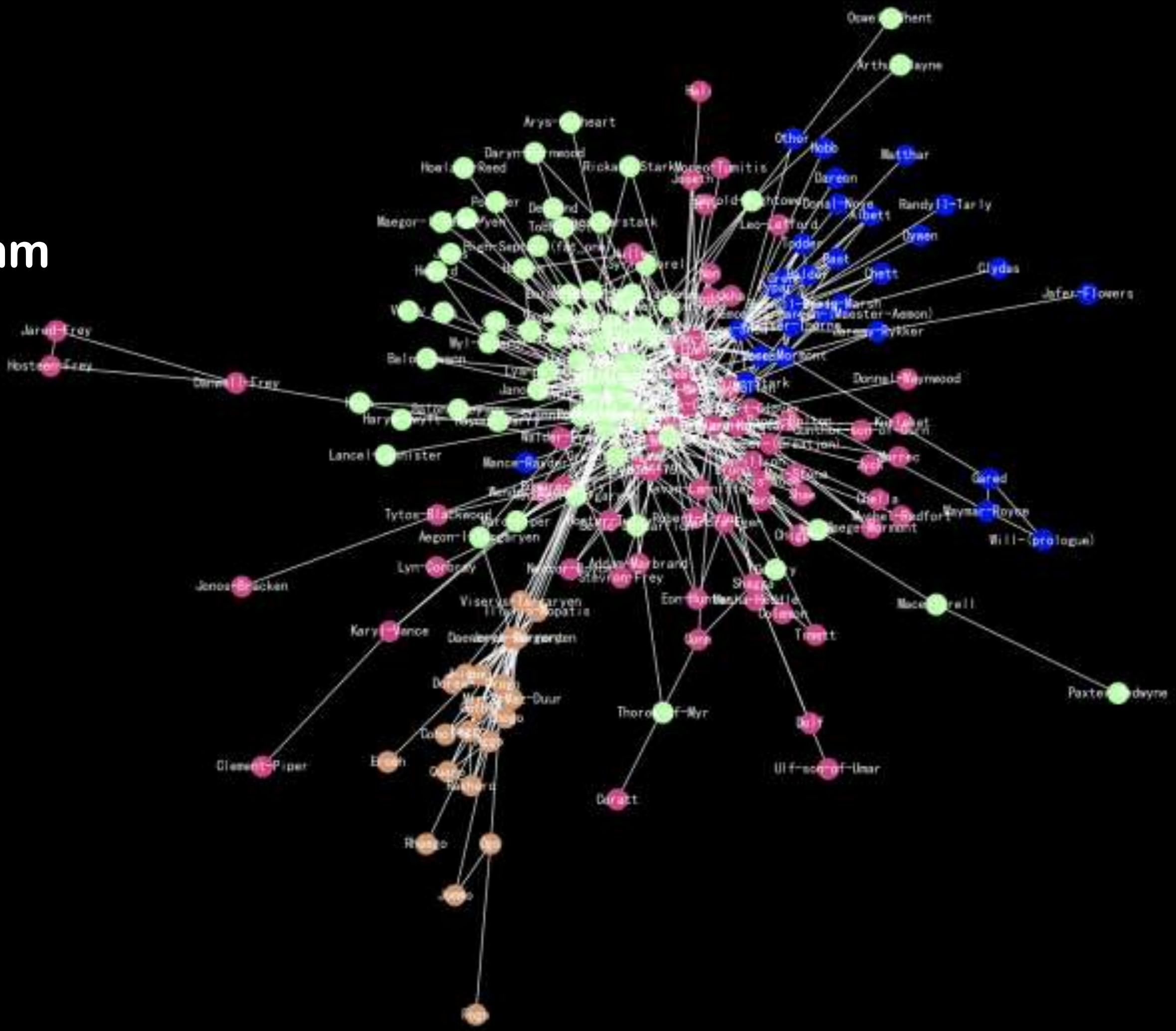


Post-processing

Kmeans clustering algorithm

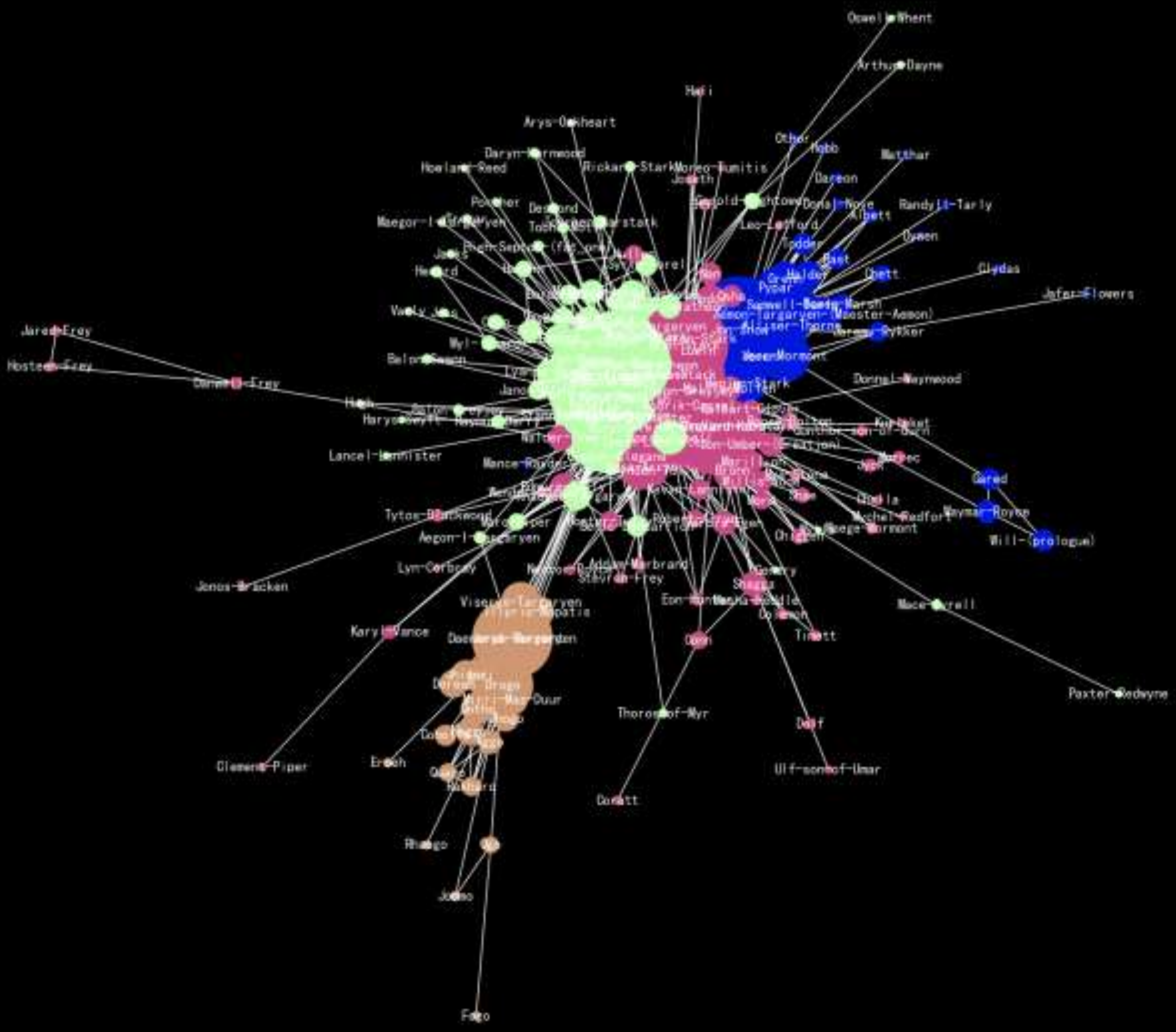
n_clusters=4

Consider **weights** between nodes



Post-processing

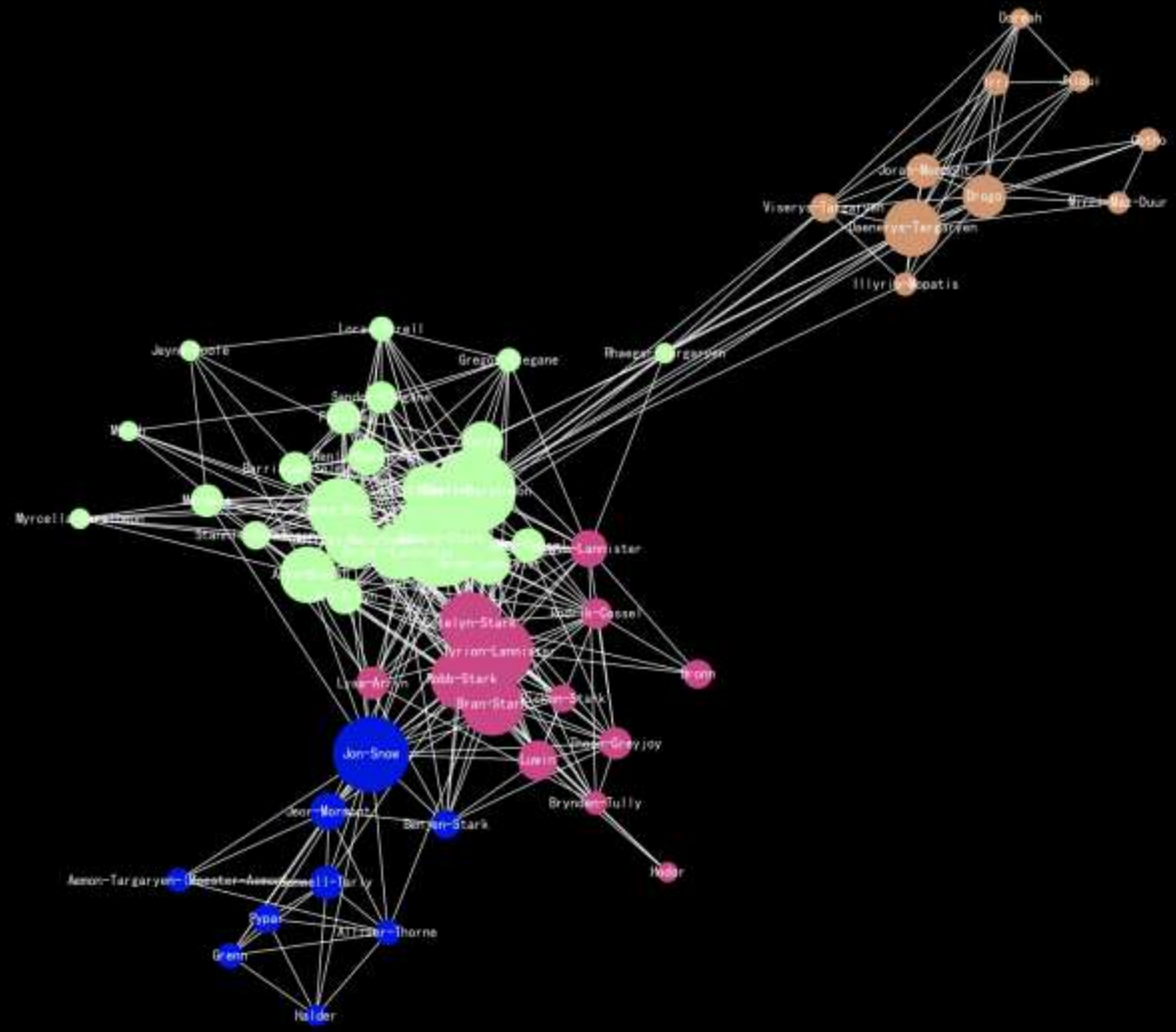
visualize the weights



Post-processing

visualize the weights

Get rid of low-weights characters

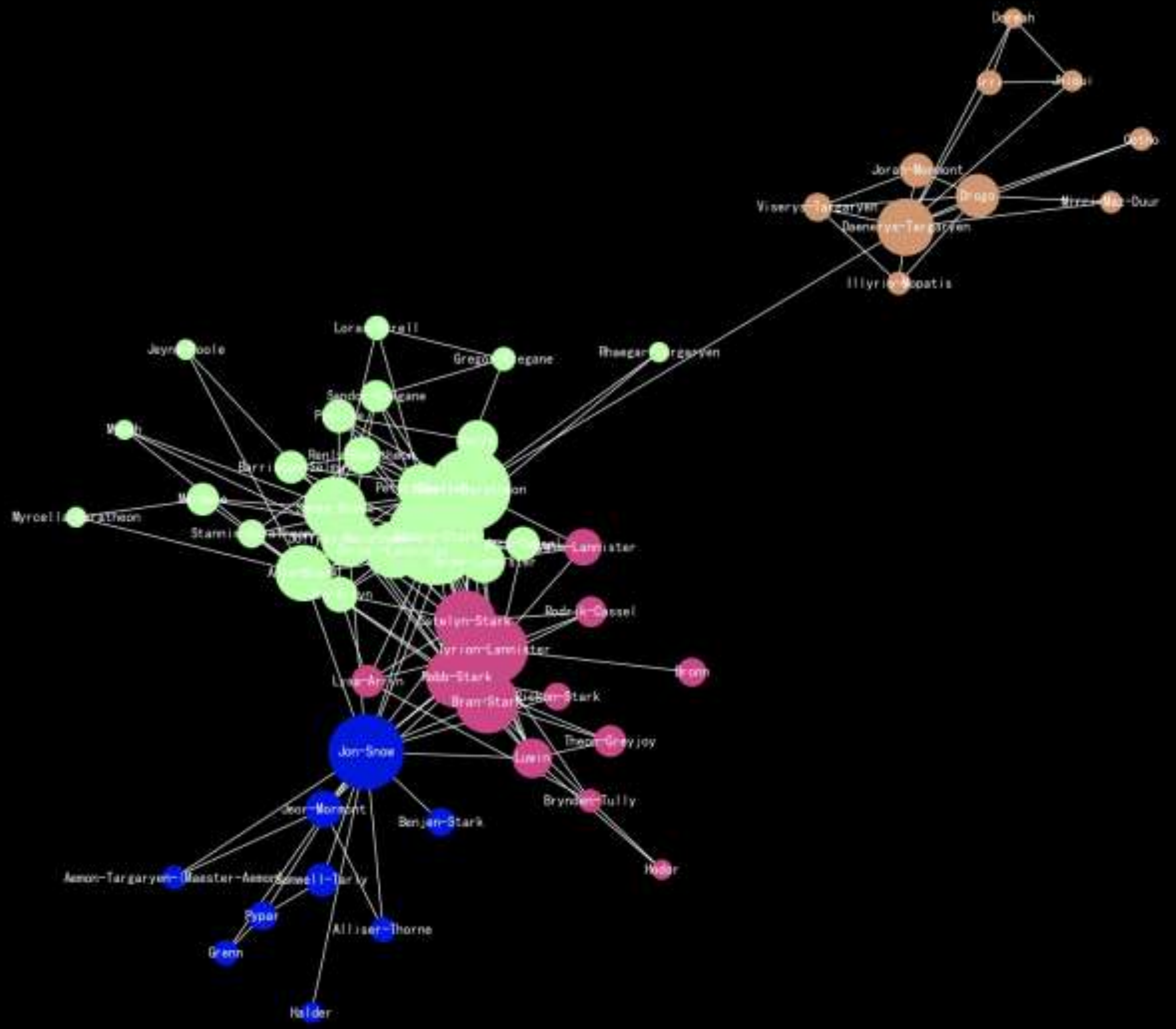


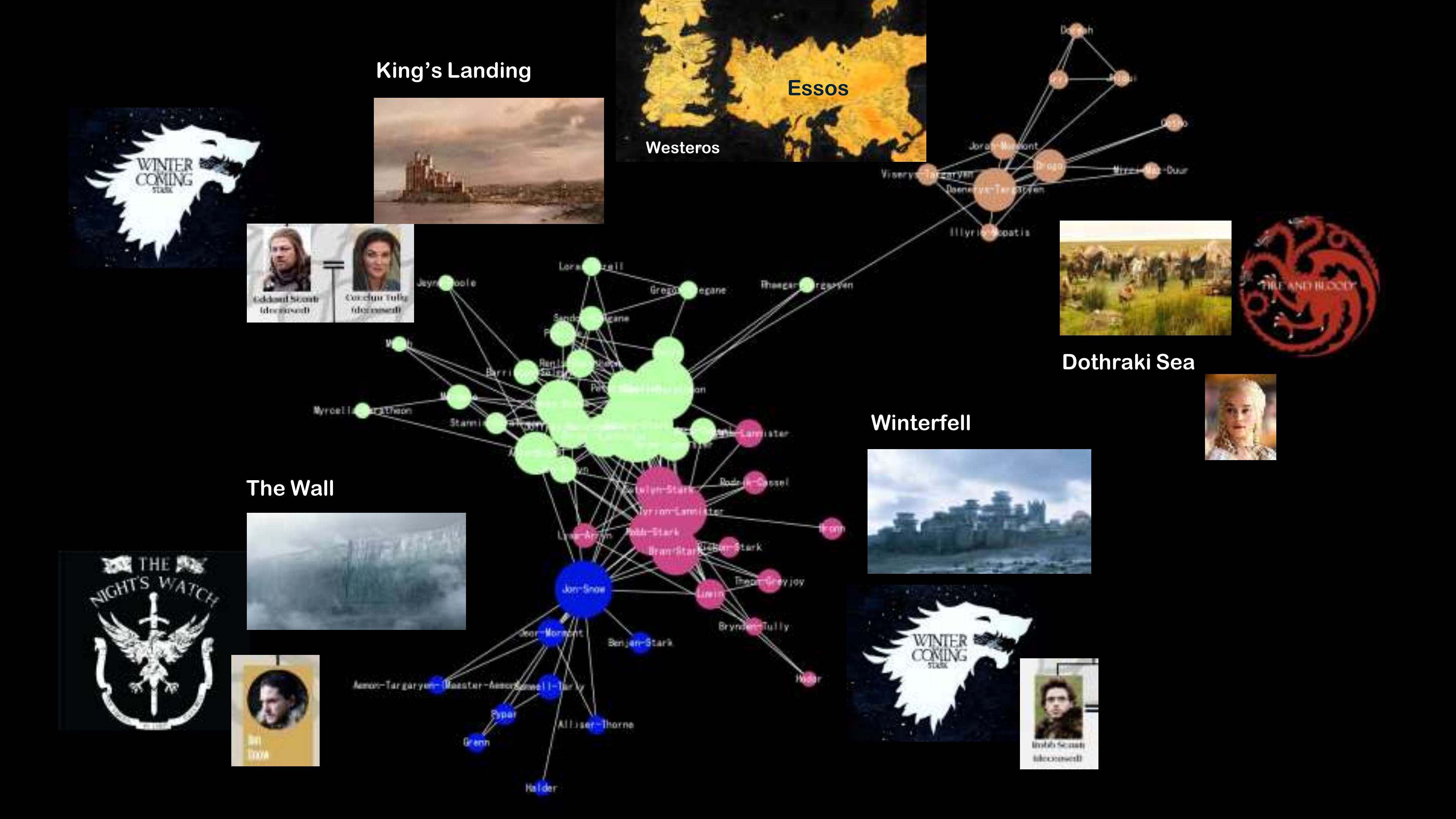
Post-processing

visualize the weights

Get rid of low-weights characters

Cut out some of the less frequent connections





King's Landing
Essos
Westeros
Dothraki Sea
Winterfell
The Wall
WINTER IS COMING
STARK
THE NIGHT'S WATCH
Jorah-Mormont
Drogo
Viserys-Targaryen
Daenerys-Targaryen
Illyrio-Mopatis
Rhaegar-Targaryen
Jeyne-Poole
Myrcella-Baratheon
Catelyn-Stark
Tyrion-Lannister
Robb-Stark
Bran-Stark
Rodrik-Cassel
Bronn
Theon-Greyjoy
Luwin
Brynden-Tully
Hodor
Jon-Snow
Jeor-Mormont
Benjen-Stark
Aemon-Targaryen
Maester-Aemon
Samwell-Tarly
Pypar
Grenn
Alliser-Thorne
Halder

Test

 # Find similar nodes of Jon-Snow node

```
>>>model.wv.most_vector('Jon-Snow')

<<<[( 'Dareon', 0.7889388203620911),          # Same group of recruits as Jon Snow.
    ('Donal-Noye', 0.7577574849128723),      # Night's Watch the blacksmith
    ('Matthar', 0.7506797313690186),         # Same group of recruits as Jon Snow.
    ('Othor', 0.740899384021759),            # Ranger of the Night's Watch
    ('Dywen', 0.737879216667099),
    ('Rast', 0.7340330481529236),
    ('Hobb', 0.7294636368751526),
    ('Grenn', 0.7205644249916077),
    ('Albett', 0.7146326303482056),
    ('Todder', 0.706096887588501)]
```

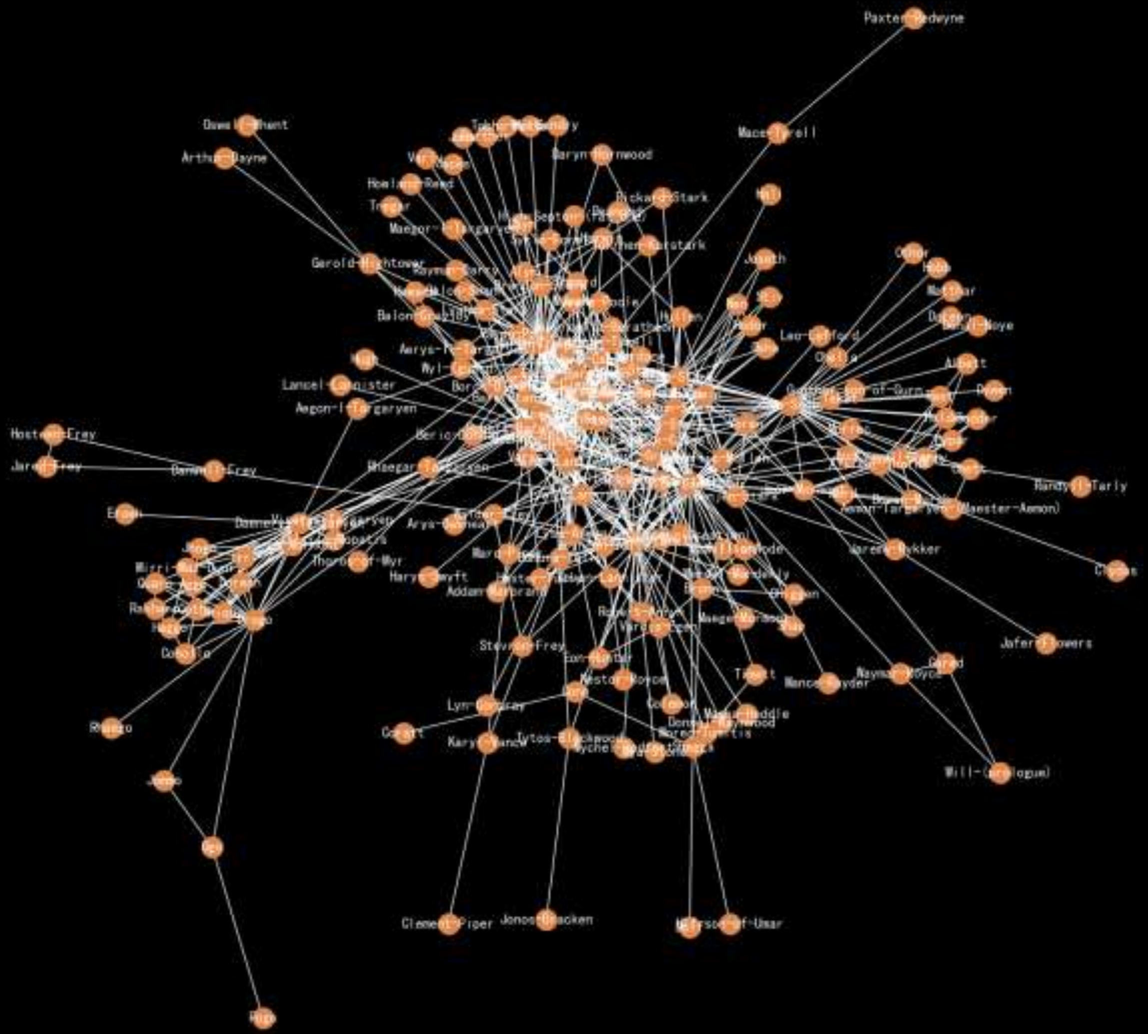
Comrades

homogeneous communities

Processing-BFS

Node2Vec
(G,
dimensions=32,
p=0.1,
q=100,
walk_length=10,
num_walks=600,
workers=4)

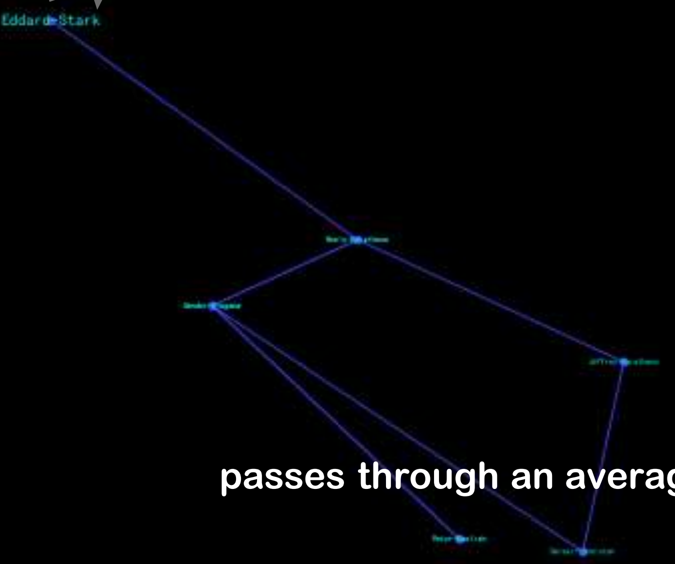
BFS breadth-first search

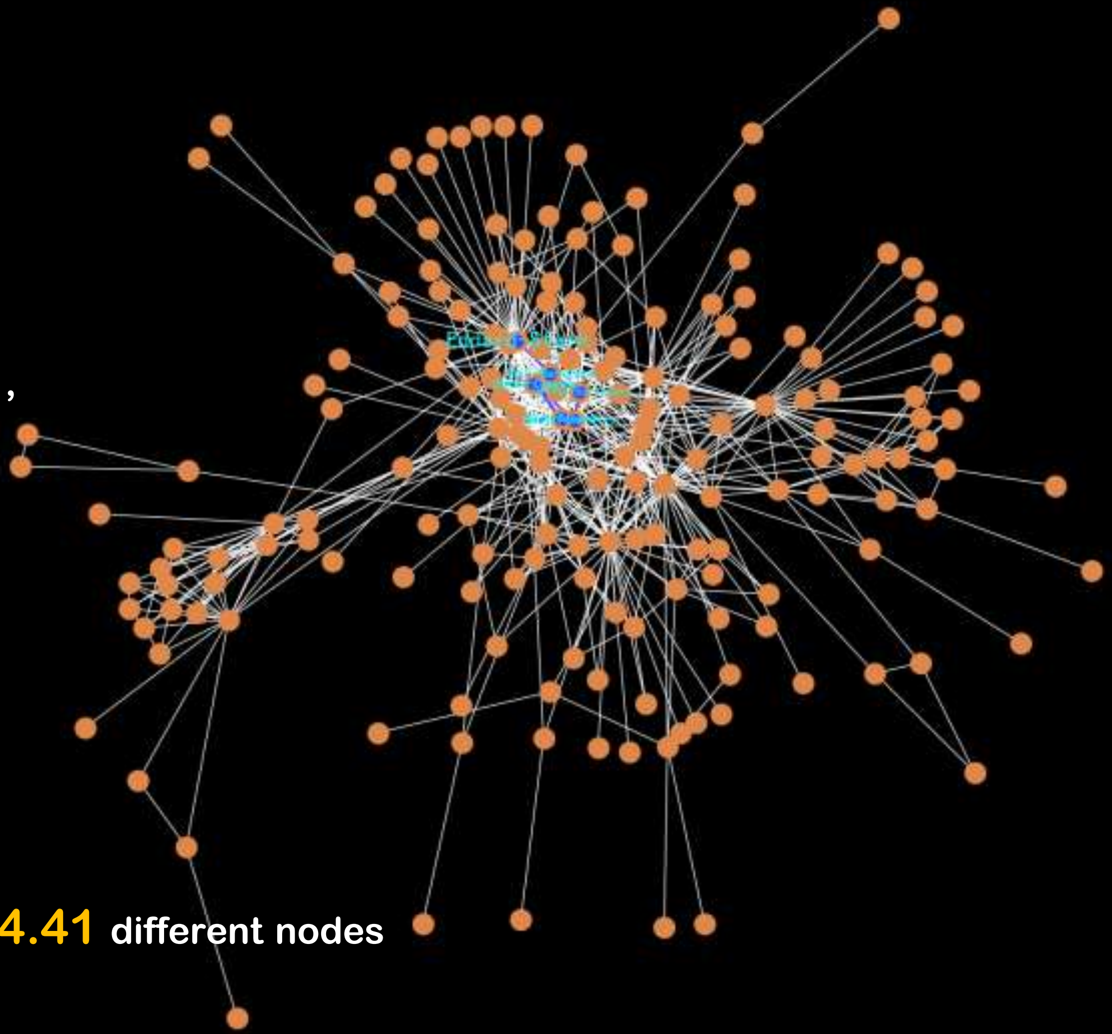


Processing-BFS

BFS breadth-first search,
find structural equivalence

array([0.31407943, 0.24742755, ... ,
        -0.5104667], dtype=float32)





passes through an average of **4.41** different nodes

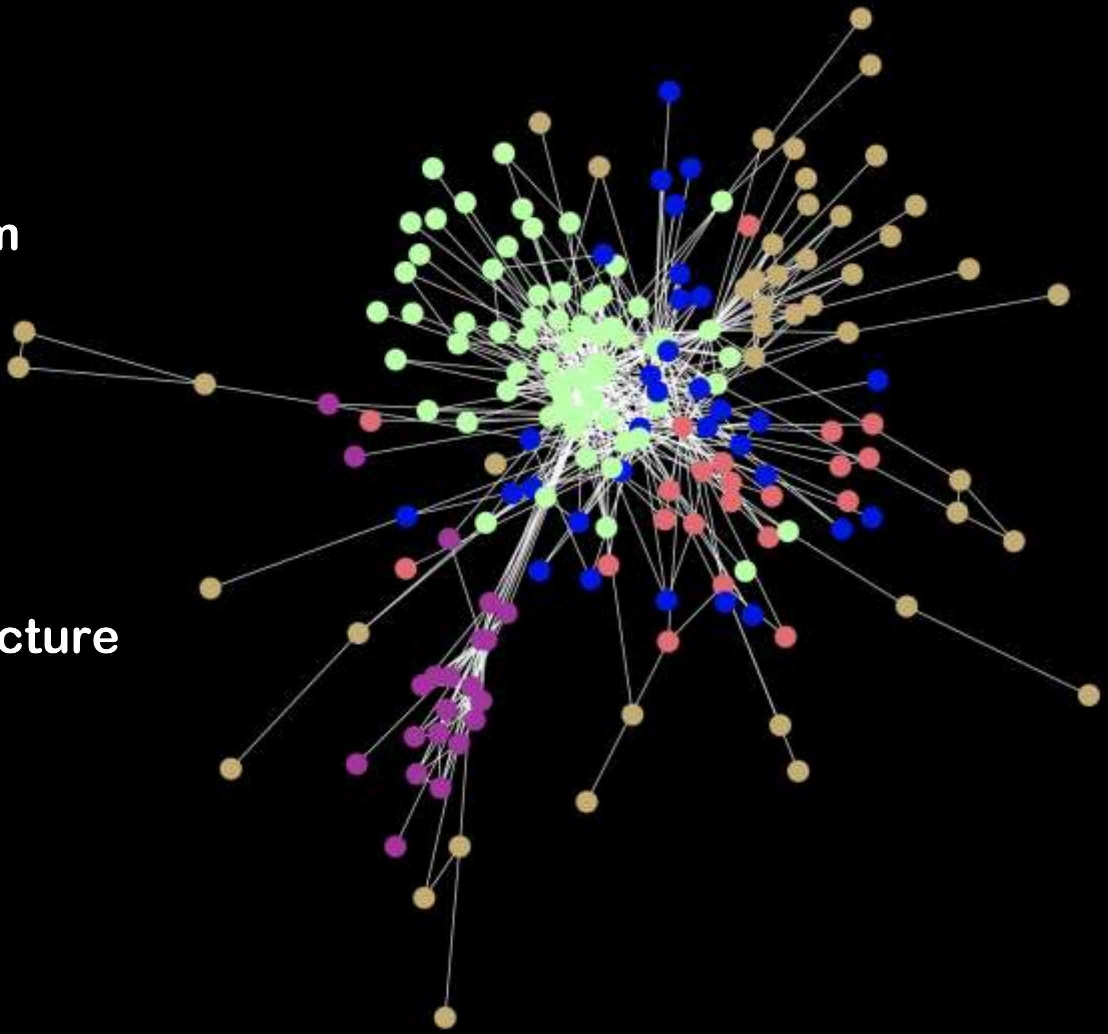
Post-processing

Kmeans clustering algorithm

n_clusters=6

If we consider weights between nodes

Nodes are classified by structure



Test

Find similar nodes of Jon-Snow node

```
>>>model.wv.most_vector('Jon-Snow')

<<<[('Alliser-Thorne', 0.6657752394676208), # master-at-arms at Castle Black
    ('Bowen-Marsh', 0.6607412695884705), #  First Steward at Castle Black
    ('Halder', 0.647807240486145),
    ('Grenn', 0.6452684998512268),
    ('Chett', 0.6423465609550476),
    ('Jeor-Mormont', 0.6395081877708435), # 997th Lord Commander of the Night's Watch
    ('Pypar', 0.6281514763832092),
    ('Samwell-Tarly', 0.6267001032829285),
    ('Dareon', 0.621441304683685 3),
    ('Hobb', 0.6167237758636475)]
```

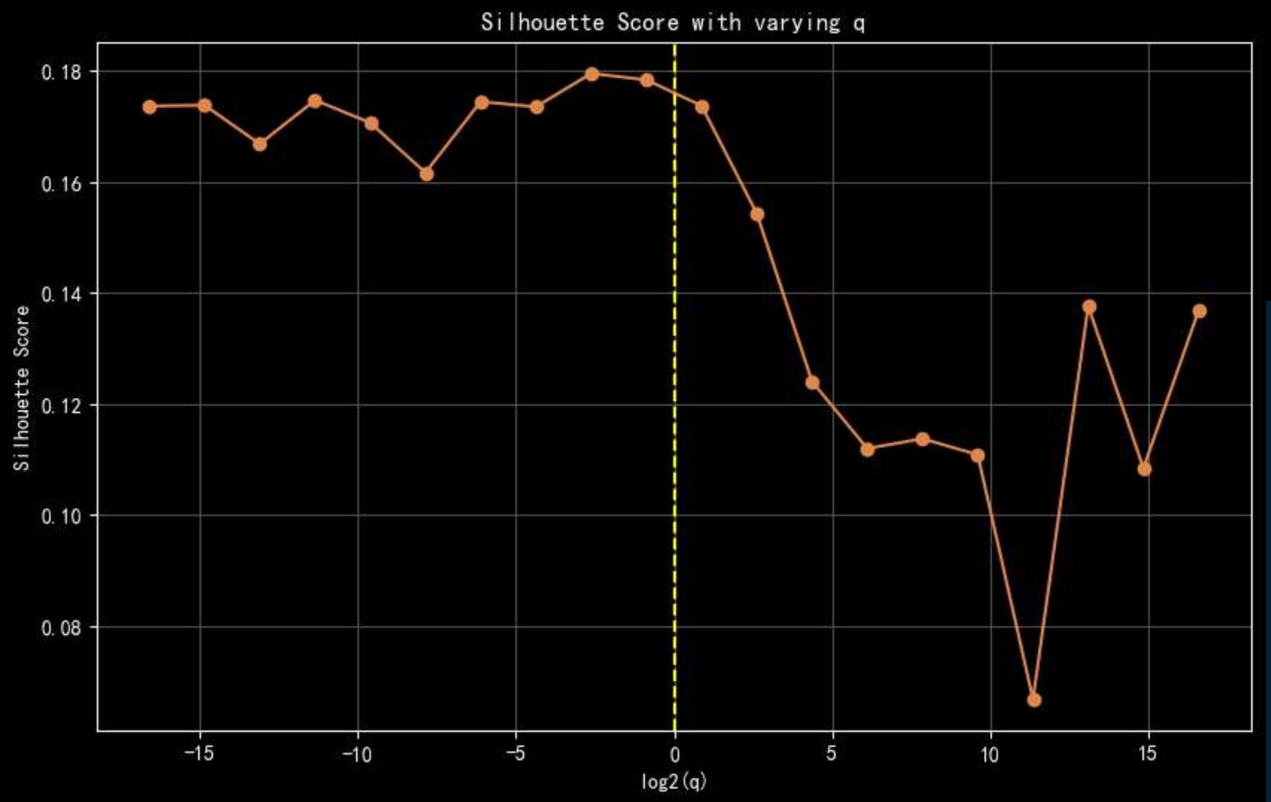


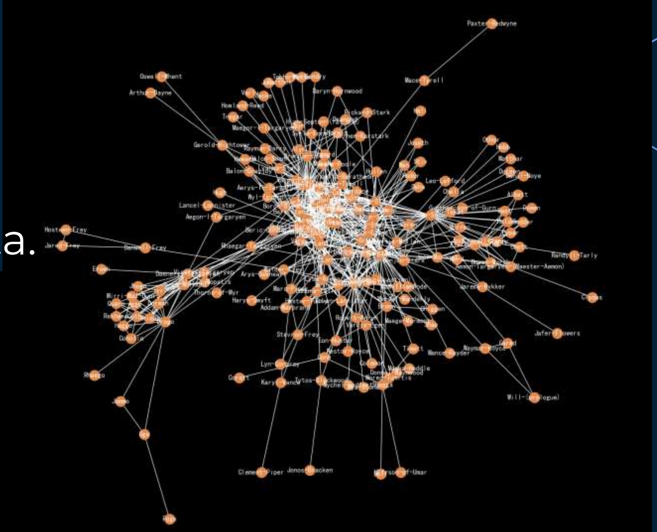

structural equivalence

Evaluation

p=1, q=$[10^{-5}, 10^{5}]$

Interpretation and validation of consistency within clusters of data.





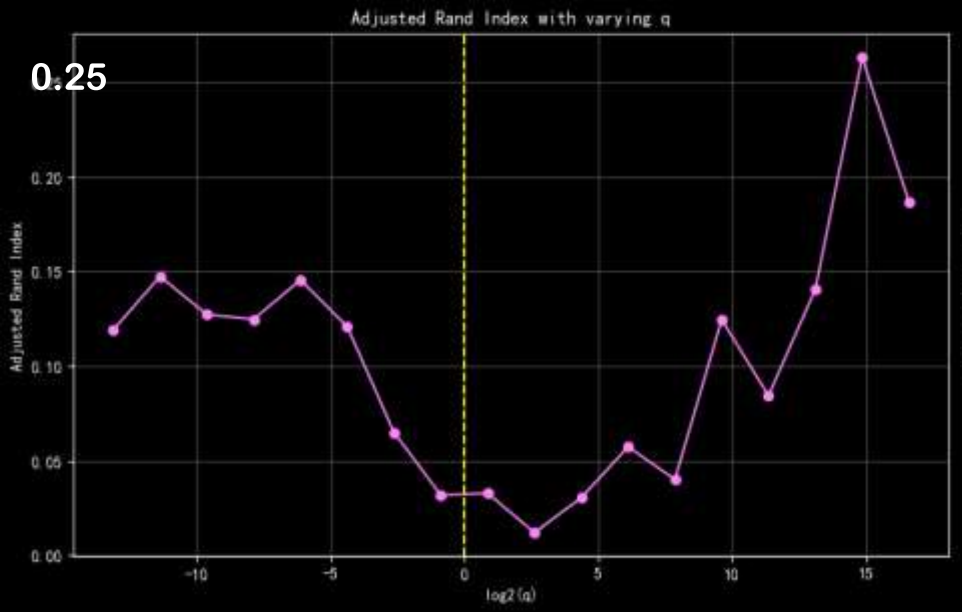
The structural differentiation is not so obvious

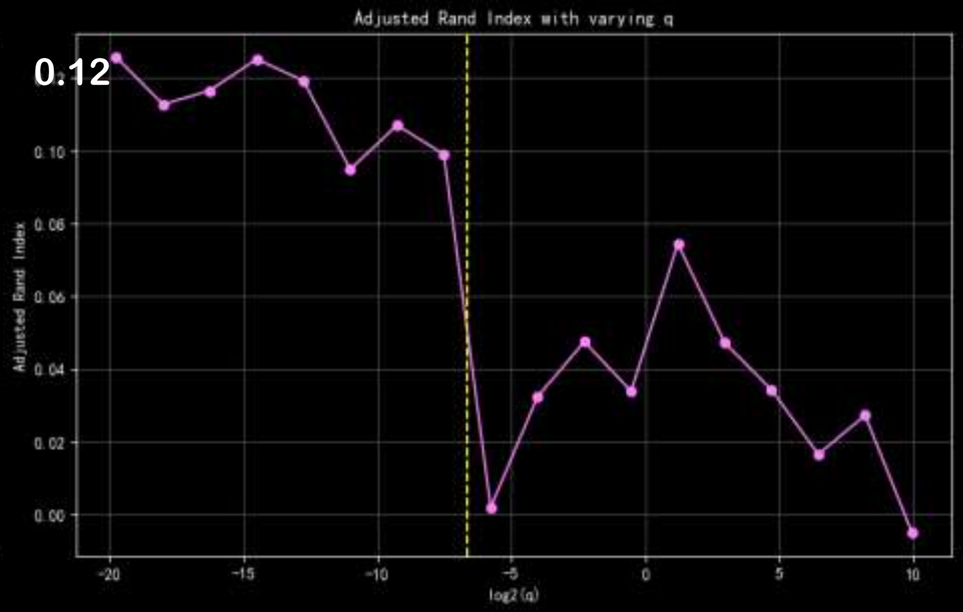
Lead to the results tend to be stable when q>>p

Evaluation-Parameter sensitivity

p=1, q=$[10^{-5}, 10^{5}]$

p=10^{-2}, q=$[10^{-7}, 10^{3}]$



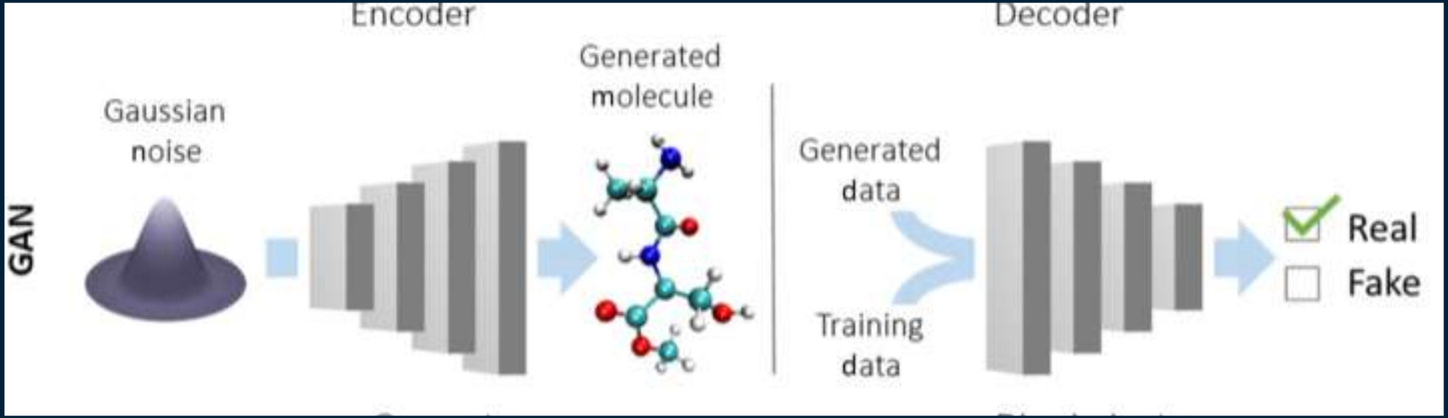


While a low q encourages outward exploration, it is balanced by a low p which ensures that the walk does not go too far from the start node.

06

FUTURE DIRECTIONS

Random Walk in predictions



- Their algorithms based on learning automata
- Based on Q-learning
- Based on deep learning and neural network
- Based on game theory
- Their algorithms for complex analysis
- Chemical molecules