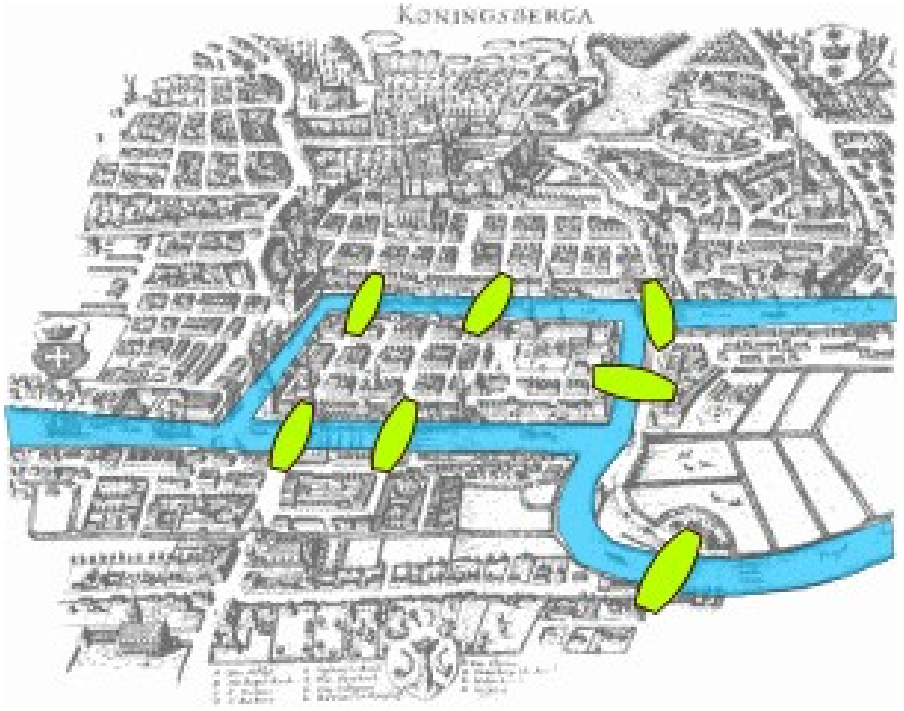
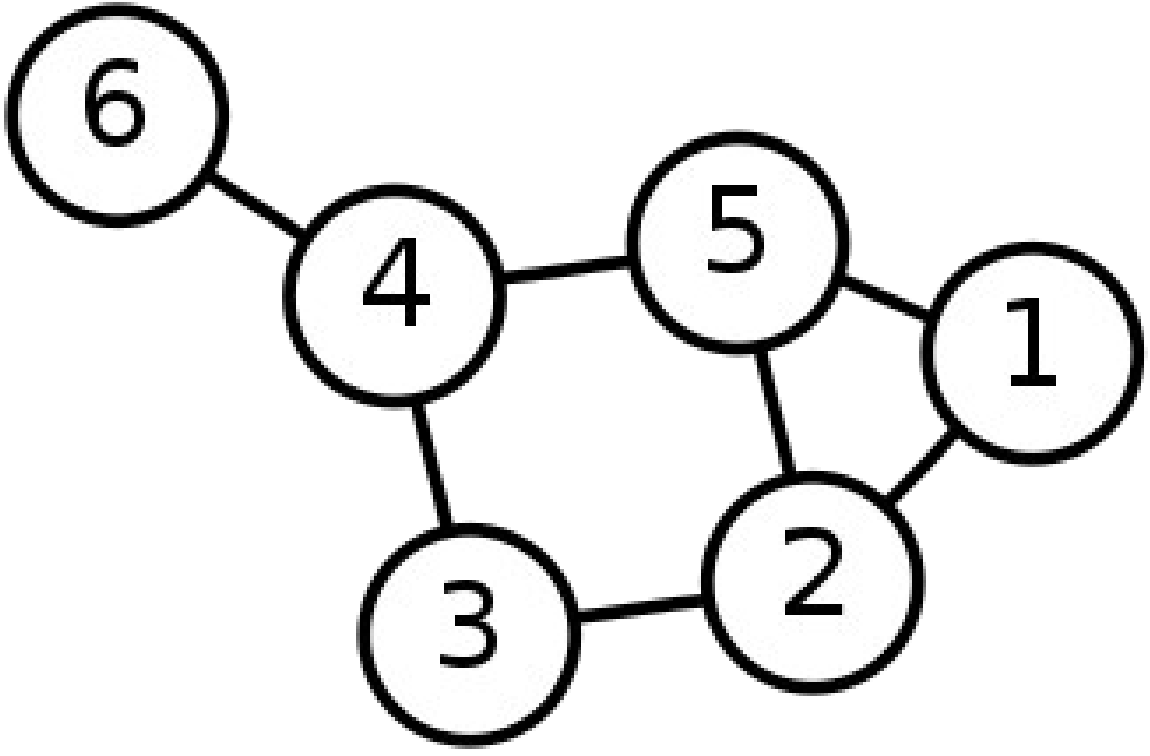
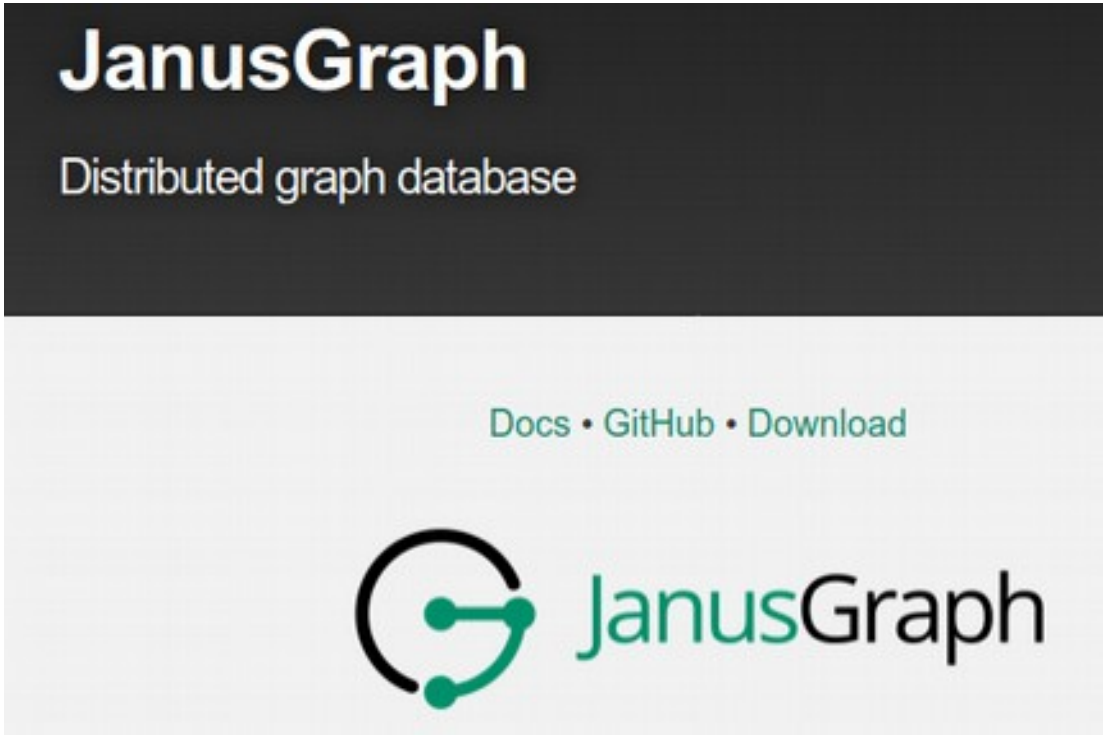
Yapay sinir ağları

Königsberg köprü problemi




6
4
5
1
3
2


JanusGraph
Distributed graph database
Docs • GitHub • Download
JanusGraph

```
Vertex saturn = tx.addVertex(T.label, "titan", "name", "saturn", "age", 10000);
Vertex sky = tx.addVertex(T.label, "location", "name", "sky");
Vertex sea = tx.addVertex(T.label, "location", "name", "sea");
Vertex jupiter = tx.addVertex(T.label, "god", "name", "jupiter", "age", 5000);
Vertex neptune = tx.addVertex(T.label, "god", "name", "neptune", "age", 4500);
Vertex hercules = tx.addVertex(T.label, "demigod", "name", "hercules", "age", 30);
Vertex alcmene = tx.addVertex(T.label, "human", "name", "alcmene", "age", 45);
Vertex pluto = tx.addVertex(T.label, "god", "name", "pluto", "age", 4000);
Vertex nemean = tx.addVertex(T.label, "monster", "name", "nemean");
Vertex hydra = tx.addVertex(T.label, "monster", "name", "hydra");
Vertex cerberus = tx.addVertex(T.label, "monster", "name", "cerberus");
Vertex tartarus = tx.addVertex(T.label, "location", "name", "tartarus");
```

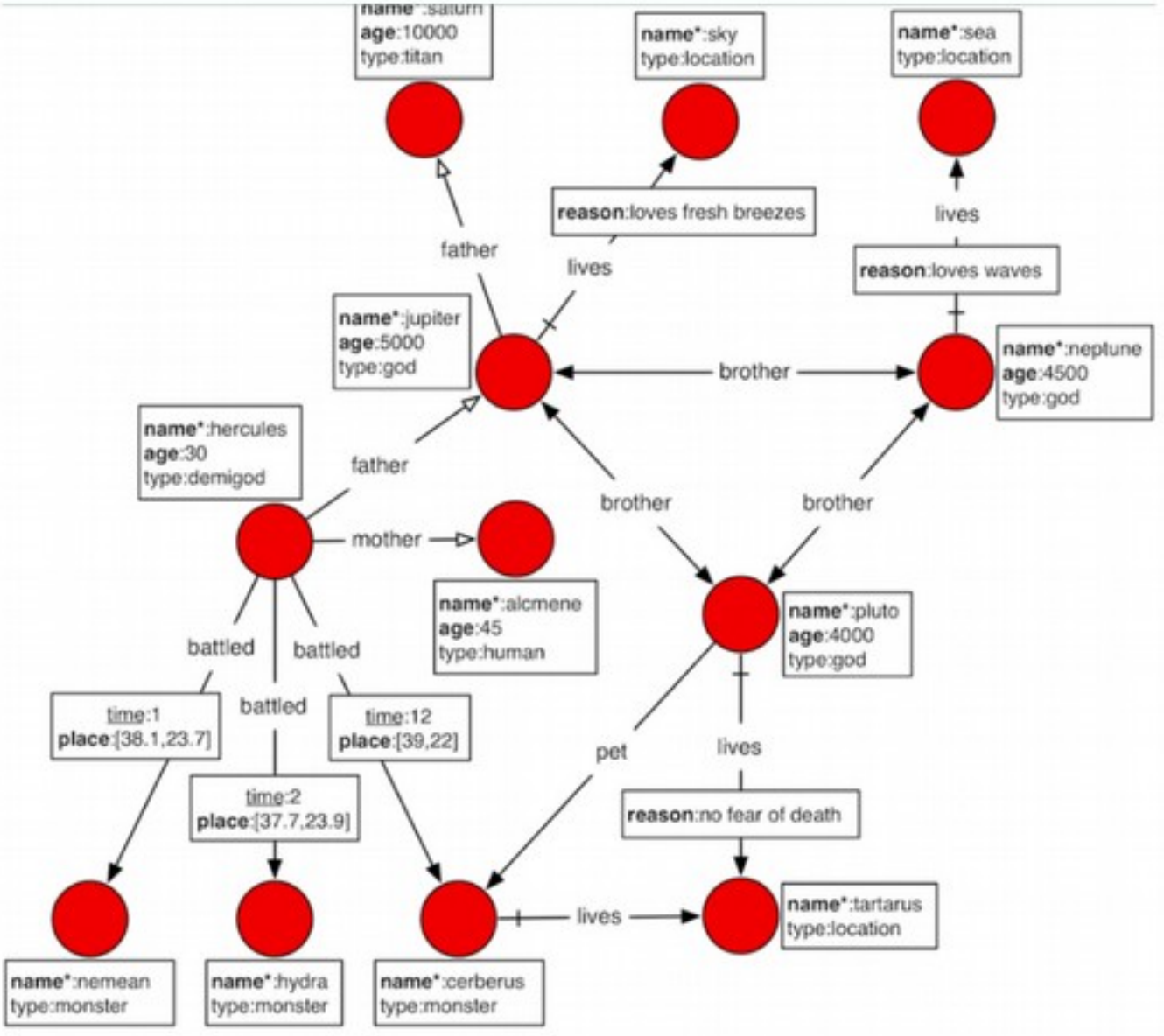
```
jupiter.addEdge("father", saturn);
jupiter.addEdge("lives", sky, "reason", "loves fresh breezes");
jupiter.addEdge("brother", neptune);
jupiter.addEdge("brother", pluto);

neptune.addEdge("lives", sea).property("reason", "loves waves");
neptune.addEdge("brother", jupiter);
neptune.addEdge("brother", pluto);

hercules.addEdge("father", jupiter);
hercules.addEdge("mother", alcmene);
hercules.addEdge("battled", nemean, "time", 1, "place", Geoshape.point(38.1f, 23.7f
hercules.addEdge("battled", hydra, "time", 2, "place", Geoshape.point(37.7f, 23.9f)
hercules.addEdge("battled", cerberus, "time", 12, "place", Geoshape.point(39f, 22f)

pluto.addEdge("brother", jupiter);
pluto.addEdge("brother", neptune);
pluto.addEdge("lives", tartarus, "reason", "no fear of death");
pluto.addEdge("pet", cerberus);

cerberus.addEdge("lives", tartarus);
```


name*:saturn
age:10000
type:titan
name*:sky
type:location
name*:sea
type:location
reason:loves fresh breezes
lives
father
lives
reason:loves waves
name*:jupiter
age:5000
type:god
brother
name*:neptune
age:4500
type:god
name*:hercules
age:30
type:demigod
father
brother
brother
mother
name*:alcmene
age:45
type:human
name*:pluto
age:4000
type:god
battled
battled
battled
time:1
place:[38.1,23.7]
time:2
place:[37.7,23.9]
time:12
place:[39,22]
pet
lives
reason:no fear of death
lives
name*:nemean
type:monster
name*:hydra
type:monster
name*:cerberus
type:monster
name*:tartarus
type:location

Personalised view

chemical entity

molecular entity

main group molecular entity

polyatomic entity

molecule

cyclic compound

polycyclic compound

aromatic compound

p-block molecular entity

carbon group molecular entity

organic molecular entity

organic cyclic compound

heterocyclic compound

bicyclic compound

heteropolycyclic compound

pnictogen molecular entity

organic molecular entity

organic heterocyclic compound

organic aromatic compound

heterobicyclic compound

organic heteropolycyclic compound

nitrogen molecular entity

heteroorganic entity

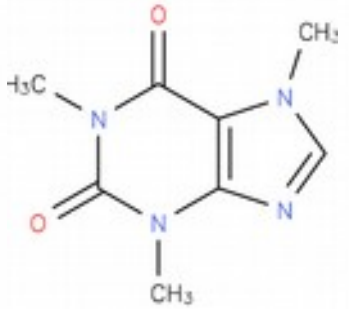
organonitrogen heterocyclic compound

heteroarene

organic heterobicyclic compound

organonitrogen compound

imidazopyrimidine



alkaloid

purines methylxanthine

purine alkaloid

trimethylxanthine

caffeine

UniProt KB

UniProt Knowledge Base of protein sequences.

1. A2AGL3
   Ryanodine receptor 3

NetworkX

Stable (notes)

2.2 — September 2018
download | doc | pdf

Latest (notes)

2.3 development
github | doc | pdf

Archive

Contact

Mailing list
Issue tracker

Software for complex networks

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Features

- Data structures for graphs, digraphs, and multigraphs
- Many standard graph algorithms
- Network structure and analysis measures
- Generators for classic graphs, random graphs, and synthetic networks
- Nodes can be "anything" (e.g., text, images, XML records)
- Edges can hold arbitrary data (e.g., weights, time-series)
- Open source 3-clause BSD license
- Well tested with over 90% code coverage
- Additional benefits from Python include fast prototyping, easy to teach, and multi-platform

```
import codecs
import pandas
import networkx as nx
import sys
import matplotlib.pyplot as plt
import pygraphviz as gv
from networkx.drawing.nx_agraph import graphviz_layout, to_agraph
from wordcloud import WordCloud
from Levenshtein import distance
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial.distance import squareform
import numpy as np
```

```
class Granph(object):

    def __init__(self, filename):
        print(sys.version)
        self.fn = filename
        df = pandas.read_csv(filename, sep='\t', encoding='utf-8')
        self.cat2id = {}
        self.cat2budget = {}
        for index, row in df.iterrows():
            cat = row['kw']
            budget = float(row['budget'])
            id = str(row['id'])
            for cc in cat.split(','):
                cc = cc.rstrip().strip()
                print(id, cc)
                if cc is not '':
                    if cc not in self.cat2budget:
                        self.cat2budget[cc] = []
                        self.cat2id[cc] = []
                self.cat2budget[cc].append(budget)
                self.cat2id[cc].append(id)
        #print(self.cat2id)
        #print('\n\n\n')
        #print(self.cat2budget)
```

```
def buildGraph(self):
    self.g = nx.Graph()
    for key in self.cat2id.keys():
        bb = 0
        for b in self.cat2budget[key]:
            bb = bb + b
        self.g.add_node(key, color='indigo', width=bb/1000000, height=bb/1000000, style='filled', fillcolor='indigo', fontcolor='white', fontsize=bb/50000)
        for id, budget in zip(self.cat2id[key], self.cat2budget[key]):
            self.g.add_node(id, color='lightcyan', width=budget/1000000, height=budget/1000000, style='filled', fillcolor='lightcyan')
            self.g.add_edge(key, id, color='black', penwidth=budget/1000000*4, label=str(budget))
    a = to_agraph(self.g)
    styles = {
        'graph': {
            'splines': 'true',
            'overlap': 'porthoyx'
        }
    }
    a.graph_attr.update('graph' in styles and styles['graph'])
    # grpv = open('graphviz.txt', 'w')
    # grpv.writelines(str(a))
    # grpv.close()
    a.layout(prog='neato')
    a.draw('%s.svg'%(self.fn))
```

```
def buildWordcloud(self):
    #print(self.cat2budget)
    c2b = {}
    for k in self.cat2budget:
        budget = 0
        for p in self.cat2budget[k]:
            budget = budget + p
        c2b[k] = budget
    wc = WordCloud(width=1600, height=800)
    wc.generate_from_frequencies(c2b)
    plt.figure()
    plt.imshow(wc, interpolation='bilinear')
    plt.axis('off')
    plt.savefig('%s.png'%(self.fn))
    plt.show()
```

```
def dendro(self):
    df = pandas.read_csv(self.fn, sep='\t', encoding='utf-8')
    items = []
    for index, row in df.iterrows():
        item = {}
        item['subjects'] = row['kw'].split(',')
        item['budget'] = row['budget']
        items.append(item)
    print(items)
    mxLeaves = []
    mxRows = []
    for i in items:
        label = ''
        for l in i['subjects']:
            label = label + l + '+'
        label = label + str(i['budget'])
        mxLeaves.append(label)
        mxCols = []
        for i1 in items:
            score = 0
            for k in i['subjects']:
                for k1 in i['subjects']:
                    score = score + distance(k, k1)
            if score == 0:
                score = 1
            mxCols.append(1 / score)
        mxRows.append(mxCols)
    print(mxLeaves)
    fig = plt.figure(figsize=(16,32))
    z = linkage(mxRows, 'ward')
    d = dendrogram(z, labels=mxLeaves, orientation='left', leaf_font_size=8)
    plt.savefig('fig.svg', dpi=600)
```

```
p = Granph('dpt projeleri.csv')
#p.buildGraph()
#p.buildWordcloud()
p.dendro()
```