**인문학술정보 지식그래프 구현**

인문학술정보 지식그래프 구현은 크게 1)네트워크 format 변환, 2)네트워크 시각화로 나뉜다. 모든 코드는 깃허브[[1]](#footnote-1)에 공개하였다. 또한 웹페이지 시각화는 링크[[2]](#footnote-2)를 참조.

1. **네트워크 format으로 변환**

JsonToNetwork.ipynb(py) 파일 아래 항목에 대한 기능 구현.

* 1. 활용 Python 패키지

JSON 파일 로드(json), 네트워크 분석(NetworkX, python-louvain), 시각화(matplotlib, seaborn)

* 1. JSON 파일 로드  
     JSON 파일을 Python의 딕셔너리(dict) 형태로 가져온다.

with open('./Data\_20211122.json', 'r', encoding='utf-8') as file:

    data = json.load(file)

딕셔너리의 key 값이 Node로 시작하면 Node 관련 정보이며, Link로 시작하면 Edge 관련 정보.

* 1. NetworkX로 네트워크 객체 만들기
     1. 네트워크 객체에 Node 정보 투입

Node의 고유 id와 label 등의 속성값(attribute)을 ‘add\_nodes\_from’ 함수를 활용해 투입.

for node\_key in Nodes:

    print(node\_key)

    G.add\_nodes\_from([(N['id'],{k:v for k,v in N.items() if k!='id'}) for N in Nodes[node\_key]])

* + 1. 네트워크 객체에 Edge 정보 투입

Edge의 고유 id와 label 등의 속성값(attribute)을 ‘add\_edges\_from’함수를 활용해 투입

for edge\_key in Edges:

    print(edge\_key)

    G.add\_edges\_from([(E['sourceid'],E['targetid'],{k:v for k,v in E.items() if 'id' not in k}) for E in Edges[edge\_key]])

* 1. Community Detection (커뮤니티 탐지)

Node와 Edge로 구성된 네트워크에서 특정 커뮤니티(그룹)이 있는지 파악한다. Louvain community detection 알고리즘[[3]](#footnote-3)으로 Node의 커뮤니티 아이디를 부여한다. 이때 best\_partition 함수를 활용한다.

partition = community.best\_partition(G)

커뮤니티 탐지 결과를 NetworkX의 kamada\_kawai\_layout[[4]](#footnote-4) 함수를 활용해 시각화한다.

pos = nx.drawing.layout.kamada\_kawai\_layout(G)

# color the nodes according to their partition

cmap = cm.get\_cmap('viridis', max(partition.values()) + 1)

nx.draw\_networkx\_nodes(G, pos, partition.keys(), node\_size=40,

                       cmap=cmap, node\_color=list(partition.values()))

nx.draw\_networkx\_edges(G, pos, alpha=0.5)

plt.show()

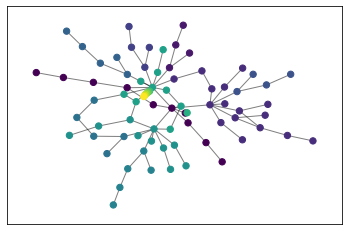


그림 1. 커뮤니티 시각화 결과

* 1. JSON 파일로 네트워크 객체 저장

네트워크 객체를 SigmaJS에서 읽을 수 있도록 JSON 형태로 다시 저장. NetworkX의 json\_graph.node\_link\_data[[5]](#footnote-5) 함수를 활용해 JSON 형태로 변환 가능.

G\_Json = nx.json\_graph.node\_link\_data(G)

이때 위에서 커뮤니티 탐지 결과 등의 추가 속성값(x,y 좌표 등)을 JSON 객체에 추가.

for node in G\_Json['nodes']:

    node['key'] = node.pop('id')

    node['label'] = node.pop('name')

    node['attributes'] = {}

    node['attributes']['group'] = partition[node['key']]

    node['attributes']['x'] = int(kamada\_kawai\_pos[node['key']][0] \* 256)

    node['attributes']['y'] = int(kamada\_kawai\_pos[node['key']][1] \* 256)

    node['attributes']['color'] = "rgba("+ ",".join([str(int(i \* 255)) for i in pal[partition[node['key']]]])+ ")"

1. **네트워크 시각화**

Index.html 파일에 아래 내용을 구현.

* 1. SigmaJS 버전 설정

SimaJS v1을 활용해 시각화를 진행. Index.html에서 Javascript(JS)로 개발.

<!-- START SIGMA IMPORTS -->

<script src="src/sigma.core.js"></script>

<script src="src/conrad.js"></script>

<script src="src/utils/sigma.utils.js"></script>

<script src="src/utils/sigma.polyfills.js"></script>

<script src="src/sigma.settings.js"></script>

* 1. Data load  
     위 1단계에서 저장한 JSON 파일(‘graph\_data.json’)을 SigmaJS에서 로드.

$(document).ready(function(){

        $.getJSON('graph\_data.json', function(data) {

          //populate the cars datalist

          $(data.nodes).each(function() {

              nodeOption = "<option value=\"" + this.name + "\">" + this.name + "</option>";

              $('datalist').append(nodeOption);

* 1. Node 위치 정하기  
     Node와 Edge 객체 생성 후 랜덤한 위치에 배치.

s = new sigma({

        graph: g,

        container: 'graph-container',

        renderer: {

            container: document.getElementById('graph-container'),

            type: 'canvas'

        },

* 1. Node 크기 정하기

Node에 연결된 Edge 수를 기반으로 Degree를 자동으로 계산한 다음 이를 바탕으로 Node의 크기를 정함.

for (i = 0; i < len; i++) {

nodes[i].size = s.graph.degree(nodes[i].id);

            }

* 1. Node/Edge 색깔 정하기  
     Class(참고문헌, 서적 등)를 활용해 Node의 색깔을 부여. Edge의 경우 Edge label(관계 속성)에 따라 색상 부여.

                console.log(nodes)

                    s.graph.nodes().forEach(function(n) {

                        n.color = nodeColors[n.id].color

                    }

                    );

                    s.graph.edges().forEach(function(e) {

                        e.color = edgeColors[e.id].color

                    });

* 1. Force-atlas2 layout 적용  
     Force-atlas2 layout[[6]](#footnote-6)을 적용해 Node와 Edge를 시각화.

            s.startForceAtlas2();

            window.setTimeout(function () {

                s.killForceAtlas2()

            }, 1000);

* 1. 부가기능 구현
     1. Zoom in/out, Full screen, Initial state 등의 화면 제어 기능을 아이콘으로 구현

 var c = s.camera;

    $(document).ready(function () {

        $(".zoom-in").bind("click", function () {

            sigma.misc.animation.camera(c, {

                ratio: c.ratio / c.settings('zoomingRatio')

            }, {

                duration: 200

            });

        });

        $(".zoom-out").bind("click", function () {

            sigma.misc.animation.camera(c, {

                ratio: c.ratio \* c.settings('zoomingRatio')

            }, {

                duration: 200

            });

        });

        $(".initial").bind("click", function () {

            s.cameras[0].goTo({x: 0, y: 0, angle: 0, ratio: 1});

        });

    });

* + 1. Mouse hover  
       마우스를 Node 위에 올려놓으면 세부 정보가 뜨는 기능.  
       Hover한 Node를 하이라이트 처리해 손쉽게 볼 수 있게 함. 또한 해당 Node에 연결된 Edge만 하이라이트 표기.

 s.bind('hovers', function(e){

                var hoverNode = e.data

                if (e.data.current.nodes.length !==0){

                    var nodeId = e.data.current.nodes[0].id

                    var toKeep\_in = s.graph.neighbors(nodeId);

                    toKeep\_in[nodeId] = nodeId

                }

                else {

                    var nodeId = e.data.leave.nodes[0].id

                    var toKeep\_out = s.graph.neighbors(nodeId);

                    // toKeep\_out[nodeId] = nodeId

                }

* + 1. Search  
       Node/Egde의 label과 attributes에 나온 텍스트 정보를 기반으로 검색하는 기능.

검색 시 해당 Node가 화면 중심으로 오고 연결된 Edge를 표현함.

function searchFunc() {

        // Declare variables

        var input, filter, ul, li, a, i, txtValue;

        input = document.getElementById('seeker');

        filter = input.value.toUpperCase();

        ul = document.getElementById("myUL");

        li = ul.getElementsByTagName('li');

* + 1. Node class filter  
       Node에 부여된 class(e.g. ‘Paper’)에 따라 특정 class의 Node만 볼 수 있는 기능.

 function applyCategoryFilter(e) {

                var c = e.target[e.target.selectedIndex].value;

                filter

                    .undo('node-category')

                    .nodesBy(

                        function (n, options) {

                            return !c.length || n[options.property] === c;

                        },

                        {

                            property: 'class\_field',

                        },

                        'node-category'

                    )

                    .apply();

* + 1. Node Click

Node를 누르면 해당 Node의 세부 정보가 뜨고, URL 정보가 있을 경우 해당 주소로 이동.

   var config = {

        node: [{

            show: 'clickNode',

            hide: 'hovers',

            cssClass: 'sigma-tooltip',

            position: 'bottom',

            //autoadjust: true,

            template:

                '<div class="arrow"></div>' +

                ' <div class="sigma-tooltip-header">{{name}}</div>' +

                '  <div class="sigma-tooltip-body">' +

                '    <table class="sigma-tooltip-body" style="width:100%">' +

                '      <tr><th>Class Field</th> <td>{{class\_field}}</td></tr>' +

                '      <tr><th>Name</th> <td>{{name}}</td></tr>' +

                '      <tr><th>URL</th> <td><a style="word-break: break-all" href="{{url}}">{{url}}</a></td></tr>' +

                '      <tr><th>Group</th> <td>{{group}}</td></tr>' +

                '      <tr><th>id</th> <td>{{id}}</td></tr>' +

                '    </table>' +

                '  </div>',

1. <https://github.com/ByungjunKim/PaperKnowledgeGraph> [↑](#footnote-ref-1)
2. <https://byungjunkim.github.io/PaperKnowledgeGraph/> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/Louvain_method> [↑](#footnote-ref-3)
4. <https://networkx.org/documentation/stable/reference/generated/networkx.drawing.layout.kamada_kawai_layout.html> [↑](#footnote-ref-4)
5. <https://networkx.org/documentation/stable/reference/readwrite/generated/networkx.readwrite.json_graph.node_link_data.html> [↑](#footnote-ref-5)
6. <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0098679> [↑](#footnote-ref-6)