



SESSION # 15

네트워크 분석

By Team 5
@ Jinny, Suzy, Mike, Clare
Date _ 2017. 09.05

CONTENTS

01 Network?

02 네트워크의 지표

03 중심성의 이론적 배경과 계산 알고리즘

04 네트워크 활용 사례

05 NetworkX와 Gephi

01. Network?

■ 네트워크 이론

- 수학의 그래프 이론에 따라, **연결 구조와 연결 강도** 등을 바탕으로 사용자의 영향력을 측정하는 방법.
- 사람, 그룹, 조직, 컴퓨터 및 데이터 등 **객체간의 관계** 및 **네트워크 특성과 구조를 분석**하고 시각화하는 분석 방법론
- 범죄 수사, 첩보, 조직 분석, 커뮤니케이션망 분석, 에이즈 확산 연구, 제약 연구 등의 분야에 활발하게 응용

01. Network?

■ 네트워크란?

- 다수의 **점**과 점들을 연결하는 **다수의 선**으로 구성된 망.
노드(Node), 엣지(Edge)로 구성됨
- **노드 (Node)**: 개체(사람, 조직, 사물 등) – 점으로 표시
- **엣지 (Edge)**: 개체들 간의 관계 – 선으로 연결

01. Network?

■ 네트워크의 종류

- **방향 네트워크 (directed network):** 정보전달, 국가간의 수출/수입 등 방향성이 있는 경우로, 송신자와 수신자가 확실함.
- **무방향 네트워크 (undirected network):** 외교관계, 혈연관계 등 액터 관계의 존재 자체를 문제로 하는 경우로, 일반적으로 방향성이 없는 관계임.

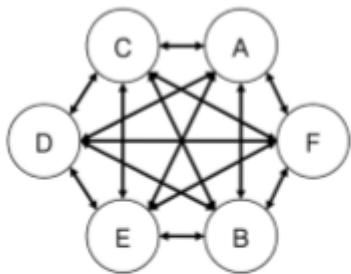
01. Network?

■ 네트워크의 종류

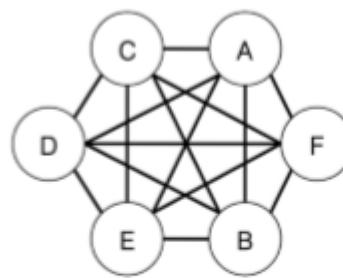
- **완전 그래프 (complete graph):** 각 정점에서 다른 모든 정점을 연결하여 가능한 최대의 연결선을 가진 그래프

정점이 n개인 방향 그래프에서 최대 엣지의 개수: $n(n-1)$ 개

정점이 n개인 무방향 그래프에서 최대 엣지의 개수: $n(n-1)/2$ 개



(a) 방향 그래프
: 30개 엣지



(b) 무방향 그래프
: 15개 엣지

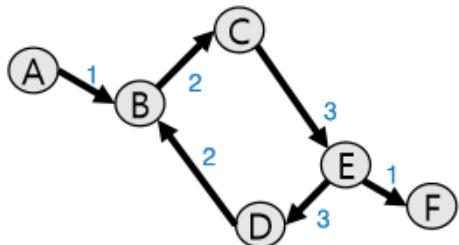
01. Network?

■ 네트워크의 종류

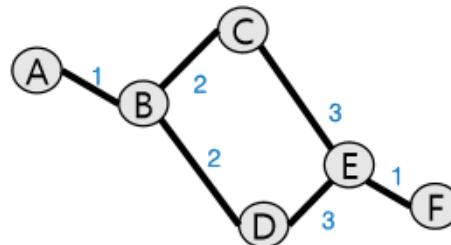
- **가중 그래프 (weighted graph):** 정점을 연결하는 엣지에 가중치를 할당한 그래프

가중치는 엣지의 정도 차이를 나타냄

친구 관계를 네트워크 그래프로 나타내는 경우, 단순한 친구관계 혹은 친분이 두터운 친구 관계에 따른 관계 정도가 가중치로 나타남



(a) 방향 그래프



(b) 무방향 그래프

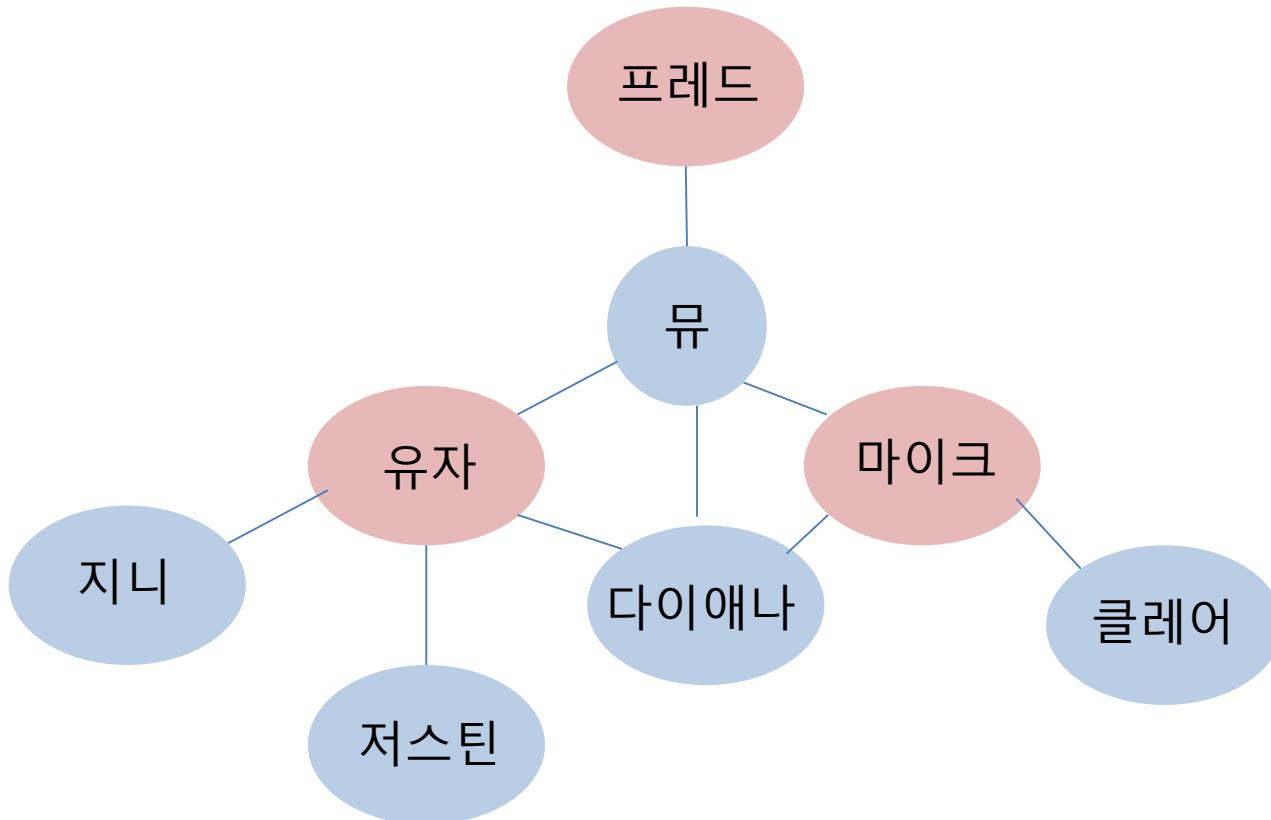
01. Network?

■ 네트워크의 예시

- 저스틴과 유자는 친구이다.
- 유자는 지니와 친구이고 다이애나와도 친구이다.
- 다이애나는 뮤와 마이크와 친구이다.
- 뮤는 유자와 친구이고 프레드, 마이크와도 친구이다.
- 마이크와 클레어는 친구이다.

01. Network?

네트워크의 예시



01. Network?

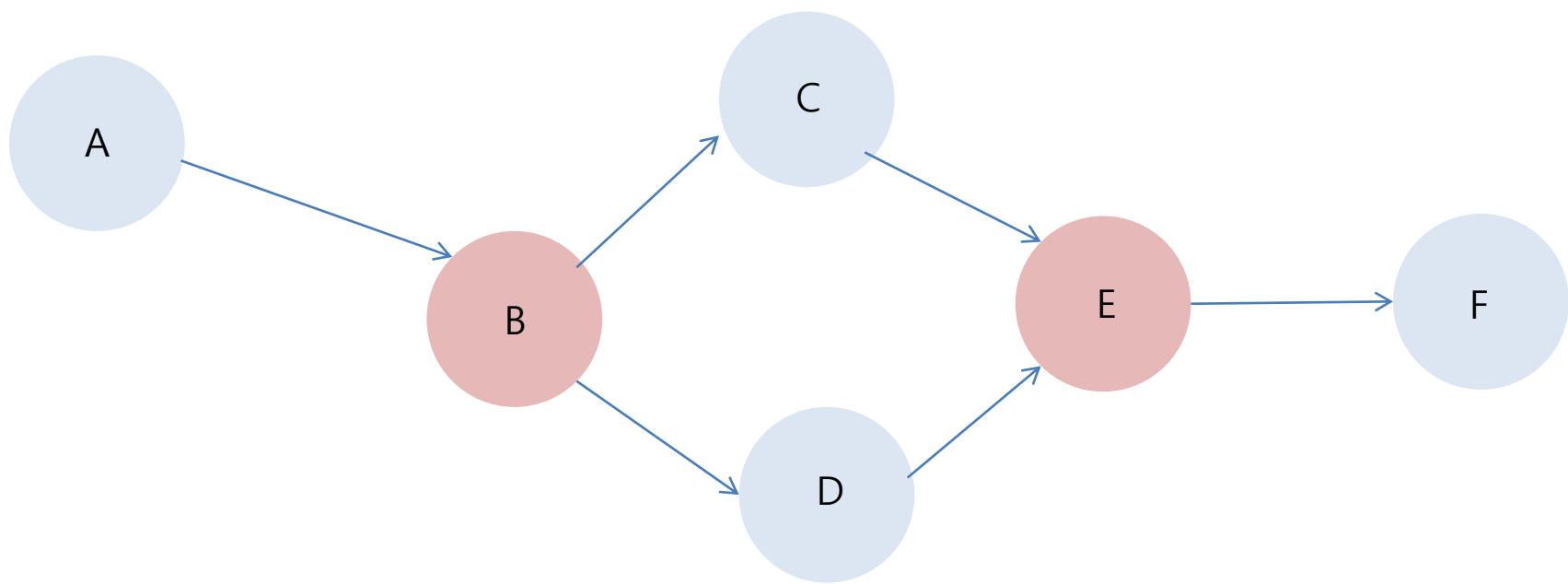
■ 네트워크의 예시

- 그래프 G란 개체를 나타내는 정점(vertex) V와 개체를 연결하는 엣지(edge) E의 집합
- $G = (V, E)$
- $V = \{A, B, C, D, E, F\}$
V : 노드(Node)
- $E = \{(A, B), (B, C), (C, E), (E, D), (D, B), (E, F)\}$
E : 엣지(Edge)

01. Network?

■ 네트워크의 예시

- $G = (V, E)$



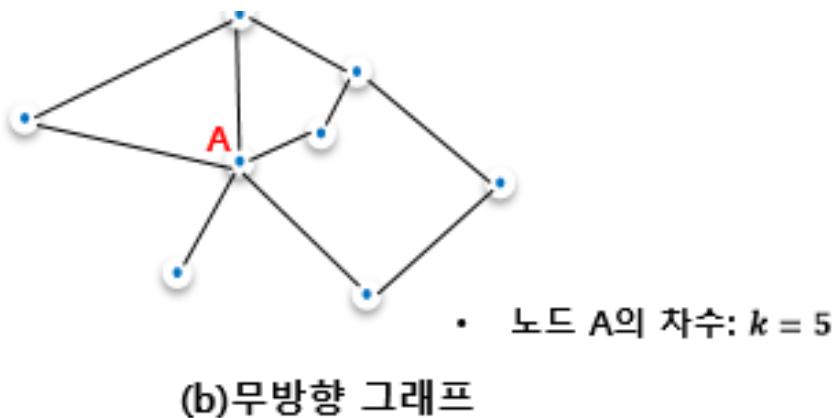
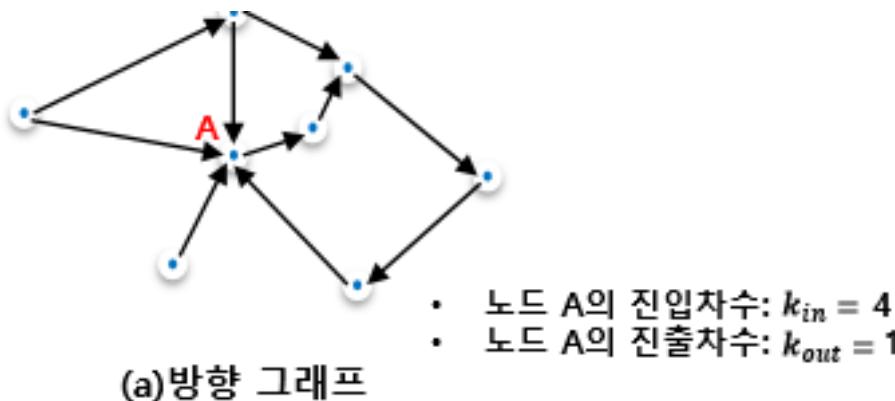
02. 네트워크의 지표

■ 차수(degree)의 정의

- 노드에 연결된 엣지들의 수로 해당 노드가 다른 노드들과 얼마나 많이 연결되어 있는가에 대한 측정지표
- 방향 그래프의 경우, 진입차수(indegree)와 진출차수(outdegree)로 구분
 - 진입차수(indegree) : 해당 노드에 들어오는 엣지들의 수
 - 진출차수(outdegree) : 해당 노드에서 나가는 엣지들의 수

02. 네트워크의 지표

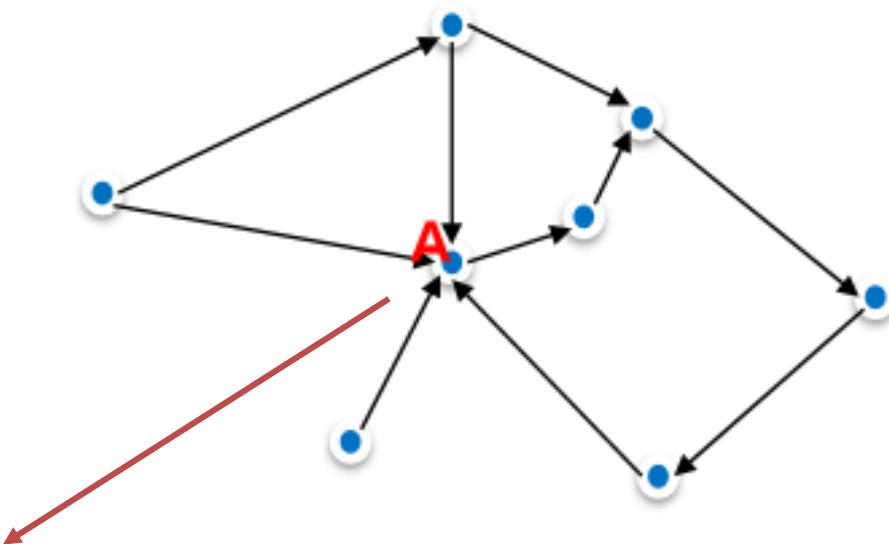
차수(degree)의 정의



02. 네트워크의 지표

■ 허브(hub)의 정의

- 노드 중에서 가장 높은 차수를 가지고 있는 노드



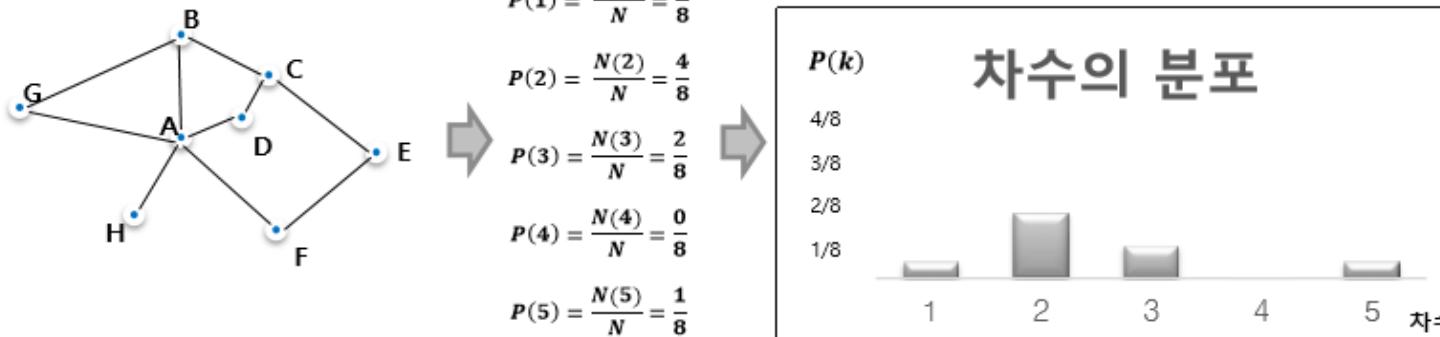
Hub: 노드 A

02. 네트워크의 지표

차수의 분포 $P(k)$

- 그래프에서 차수 K 를 갖는 노드의 비율을 의미
- 차수 k 를 갖는 노드의 수를 전체 노드 수 N 으로 나눈 값

$$P(k) = \frac{N(k)}{N}, \quad k = 1, 2, \dots, n$$



02. 네트워크의 지표

무작위 네트워크와 Scale-Free 네트워크

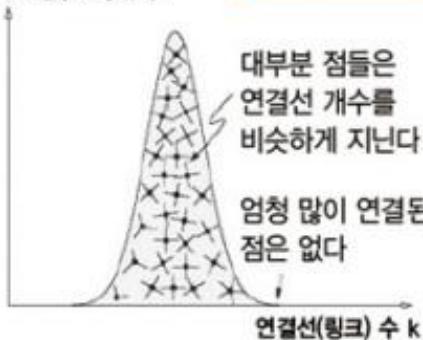
네트워크의 두 가지 유형

연결선 k개를
지닌 점(노드)의 수

가우시안 분포

대부분 점들은
연결선 개수를
비슷하게 지닌다

엄청 많이 연결된
점은 없다

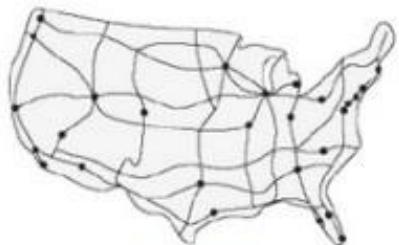
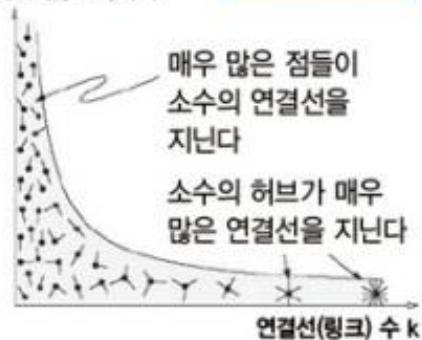


연결선 k개를
지닌 점(노드)의 수

멱함수 분포

매우 많은 점들이
소수의 연결선을
지닌다

소수의 허브가 매우
많은 연결선을 지닌다



고속도로망



항공망

멱함수(오른쪽)는 네트워크에서 얼마나 많은 점들이 몇 개의 연결선으로 연결돼 있는지를 보여주는 연결선 분포 함수다. 우리에게 익숙한, 평균 주변에 많이 모여 있는 종 모양의 '가우시안 분포'(왼쪽)와 달리, 멱함수 분포는 많은 연결선을 지닌 허브를 비롯해 다양성이 존재함을 보여준다.

링크

LINKED
The New Science of Networks

21 세기
를
지배하는
네트워크
과학

네트워크가 왜 과학의
대상이 되었는가?
인터넷에서부터 대체로 거대
기업에 이르기까지 모든
네트워크의 구조로 친숙한
한 학제적 과학자와 이해
세상에 나온다.

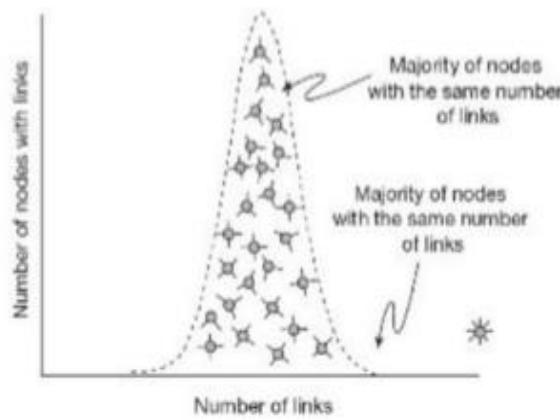
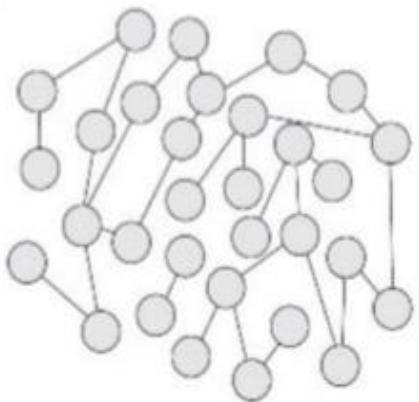
A. L. 레이저시 차별
경영학 학자를 출판

풀이사전

02. 네트워크의 지표

■ 무작위 네트워크(Random Graph, 랜덤 그래프)

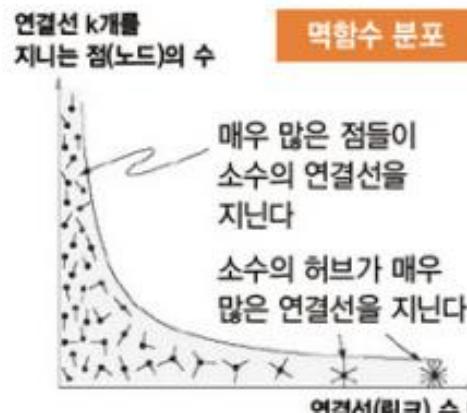
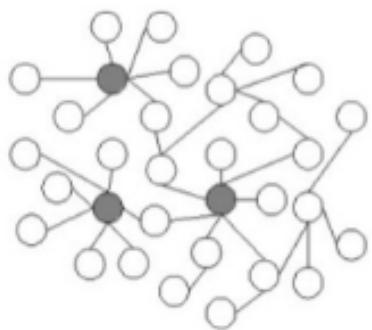
- 대다수 노드들이 유사한 수의 엣지를 갖고 있는 네트워크
- 현실에서 나타나는 네트워크의 경우 매우 높은 엣지를 가지는 노드가 나타나 무작위 네트워크로 설명이 불가능한 문제가 있음



02. 네트워크의 지표

척도 없는 그래프 (scale-free graph)

- 대부분 노드들이 소수의 엣지를 갖고 있고 몇 개의 노드들이 거대한 엣지를 가지는 네트워크
- 인터넷, 소셜 네트워크와 같은 현실의 많은 네트워크에서 Scale-free graph의 모습을 보임



02. 네트워크의 지표

밀도 (density)

- 최대 가능한 엣지들의 개수에 대한 실제 엣지들의 개수의 비
- (실제 네트워크에 존재하는 엣지의 개수) / (모든 노드가 전부 연결되어 있다는 가정하에 구한 총 엣지 수)
- 노드와 노드 사이에 엣지들이 얼마나 밀집되어 있는지를 판단할 수 있는 척도
 - 높은 밀도를 갖는 그래프는 낮은 밀도를 갖는 그래프에 비해 노드들 간에 더 많이 연결되어 있음.
 - 그래프 밀도는 0과 1사이의 값을 가짐
 - 그래프에서 노드들 간에 완전 연결되어 있는 경우, 밀도는 1의 값을 가짐

02. 네트워크의 지표

밀도 (density)의 의미와 예시

- 네트워크에서 밀도는 네트워크 내 구성원이 서로 얼마나 많은 관계를 맺고 있는가를 표현하는 지표임
- A라는 학교에서 특정 학급 학생들 간 네트워크를 N1, 학교의 학생들 간 네트워크를 N2라고 할 경우, **한 학급의 학생들 간은 서로 알고 있으나, 전체 학교의 학생들 간에서 서로 모를 수도 있다는 가정** 하에 N1의 밀도가, N2의 밀도보다 높다고 할 수 있음.

02. 네트워크의 지표

■ 중심성 (centrality)

- 개체가 전체 네트워크에서 얼마만큼 중심에 가까이 자리 잡고 있는지를 나타내는 지표
 - 특정한 노드가 많은 다른 노드들과 연결되어 있는 경우, 그 노드는 네트워크의 가운데 쪽으로 위치하게 됨.
 - 중심성은 네트워크 분석에서 개체가 가지는 영향력을 분석하는 데 많이 사용됨

02. 네트워크의 지표

■ 중심성 (centrality) 지표의 종류

- 연결정도 중심성 (degree centrality)
- 근접 중심성 (closeness centrality)
- 매개 중심성 (betweenness centrality)
- 아이겐벡터 중심성 (eigenvector centrality)

02. 네트워크의 지표

연결 정도 중심성 (degree centrality)

- 네트워크에서 한 노드가 다른 노드들과 직접적으로 연결되어 있는지를 측정하는 지표
- 특정 노드의 연결 정도 중심성은 특정 노드와 직접 연결된 노드의 수를 특정 노드와 직, 간접적으로 연결된 모든 노드의 수로 나눈 값
 - 연결된 노드의 수가 많을수록 연결 정도 중심성 상승
 - 이 지표는 단순히 1촌 만을 고려한 것으로 국지적인 범위의 역할만 파악 가능 -> 확산된 정도는 보지 못함 (한계점)

$$D_c(i) = \frac{i\text{와 직접 연결된 노드의 수}}{i\text{와 직·간접 연결된 노드의 수}}$$

02. 네트워크의 지표

연결 정도 중심성 (degree centrality)

자기 자신과 연결된 엣지 수 / ?????? [표준화를 위한 나누기]

(표준화? - ∵ 네트워크 크기에 따라 연결된 노드 수가 같아도 그 비중이 다르기 때문)

표준화하는 방법은 많으나 책에서는 네트워크 내 모든 엣지 수로 나눔
(네트워크 내 가능한 최대 수치인 (총 노드의 개수 - 1)로 나누거나
해당 네트워크 내 가장 많은 엣지를 가진 단일 노드의 엣지 수로 나누기도 함)

물론 방향성이 있으면 In-Degree와 Out-Degree를 따로 보아야 함

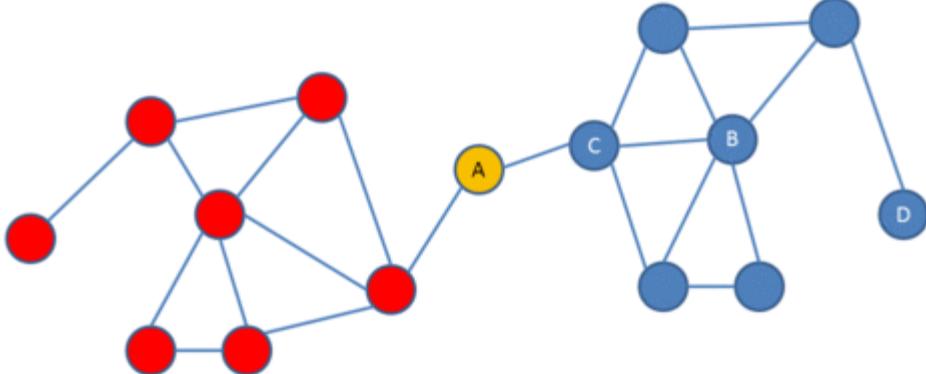
02. 네트워크의 지표

■ 근접 중심성 (closeness centrality)

- 직접적으로 연결된 노드뿐만 아니라 간접적으로 연결된 노드까지 포함해 중심성을 측정하는 지표
- 특정 노드와 네트워크에 있는 다른 노드들 간의 최단 거리가 짧을 수록 근접 중심성이 높다는 전제

$$C_c(i) = \frac{n - 1}{\sum_{j=1}^n d(i,j)}$$

04 중심성의 이론적 배경과 계산 알고리즘



노란색 노드는
꽤 중요해 보이는데도
degree centrality가 낮다.

하지만!
betweenness centrality가 출동한다면?

02. 네트워크의 지표

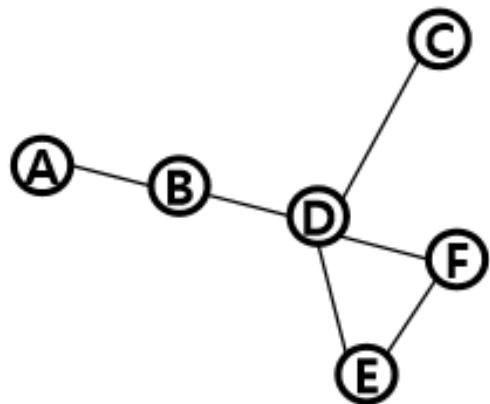
■ 매개 중심성 (betweenness centrality)

- 해당 노드를 통과하는 최단 경로들의 개수로 정의. 경로의 끝에 있는 노드의 경우, 두 노드 간의 최단 경로가 존재하지 않으므로 매개 중심성은 0임.
 - 한 노드가 다른 노드들 간의 네트워크를 구축하는데 중계자 혹은 매개자로서의 역할 정도를 나타내는 지표
 - 전체 네트워크 내에서 얼마나 다리 역할을 하는지를 나타낼 수 있음. 즉, 상이한 집단 간을 연결하는 노드일수록 매개 중심성이 높게 나타남

$$B_c(i) = \sum_{j < k} g_{jk}(n_i)$$

02. 네트워크의 지표

중심성 지표 산출 예시



■ 연결정도 중심성

A	B	C	D	E	F
$1/5=0.2$	$2/5=0.4$	$1/5=0.2$	$4/5=0.8$	$2/5=0.4$	$2/5=0.4$

■ 근접 중심성

A	B	C	D	E	F
$5/12=0.42$	$5/8=0.625$	$5/10=0.5$	$5/6=0.83$	$5/9=0.56$	$5/9=0.56$

■ 매개 중심성

A	B*	C	D**	E	F
0	4	0	8	0	0

*노드 B를 포함하는 최단경로: (A,D), (A,D,C),(A,D,E),(A,D,F)

**노드 D를 포함하는 최단경로:

(B,C),(B,E),(B,F),(A,B,C),(A,B,E),(A,B,F),(C,E),(C,F)

02. 네트워크의 지표

■ 아이겐벡터 중심성 (eigenvector centrality)

- 마이크

03 중심성의 이론적 배경과 계산 알고리즘

■ 중심성의 종류

1. 연결 중심성 (degree centrality)
2. 매개 중심성 (betweenness centrality)
3. 근접 중심성 (closeness centrality)
4. 고유벡터 중심성 (eigenvector centrality)

그 외에도 많다고 합니다.

03 중심성의 이론적 배경과 계산 알고리즘

■ 중심성의 종류 (방금 했지만 개념이 아직……)

1. 연결 중심성 (degree centrality)
2. 매개 중심성 (betweenness centrality)
3. 근접 중심성 (closeness centrality)
4. 고유벡터 중심성 (eigenvector centrality)

그 외에도 많다고 합니다.

개념을 찾으러…….



03 중심성의 이론적 배경과 계산 알고리즘

■ 본 게임 전 데이터 준비

교재 데이터를 Python으로!

```
users = [  
    { "id" = 0, "name" = "Hero"},  
    { "id" = 1, "name" = "Dunn"}, ...  
]
```

```
friendships = [  
    (0, 1), (0, 2), (6, 8), ...  
]
```

03 중심성의 이론적 배경과 계산 알고리즘

■ 본 게임 전 데이터 준비

교재 데이터를 Python으로!

```
users = [  
    { "id" = 0, "name" = "Hero"},  
    { "id" = 1, "name" = "Dunn"}, ...  
]
```

```
friendships = [  
    (0, 1), (0, 2), (6, 8), ...  
]
```

각 이용자는 사전 형태,
링크(혹은 엣지)는 튜플 형태로 구성

03 중심성의 이론적 배경과 계산 알고리즘

■ 본 게임 전 데이터 준비

링크(혹은 엣지)를 직접 추가해 봅시다.

알고리즘을 어떻게 가져가야 할까요?

```
for user in users:  
    user[“friends”] = [] # friends Key, 빈 리스트 Value 추가
```

```
for i, j in friendships:  
    users[i][“friends”].append(users[j])  
    users[j][“friends”].append(users[i]) # 방향 있는 경우엔 이 라인 제외
```

03 중심성의 이론적 배경과 계산 알고리즘

■ 본 게임 전 데이터 준비

링크(혹은 엣지)를 직접 추가해 봅시다.

알고리즘을 어떻게 가져가야 할까요?

for user in users:

 user[“friends”] = [] # friends Key, 빈 리스트 Value 추가

for i, j in friendships:

 users[i][“friends”].append(users[j])

 users[j][“friends”].append(users[i]) # 방향 있는 경우엔 이 라인 제외

```
users = [
    { “id” = 0, “name” = “Hero”},
    { “id” = 1, “name” = “Dunn”}, ...
]
```

03 중심성의 이론적 배경과 계산 알고리즘

1. 연결 중심성 (degree centrality)

책에서 주어진 대로!

자기 자신과 연결된 엣지 수 / 네트워크 내 모든 엣지 수

계산 어떻게?

특정 유저의 연결 중심성을 도출하기 위해 문자와 분모 모두 계산

03 중심성의 이론적 배경과 계산 알고리즘

1. 연결 중심성 (degree centrality)

방금 준비 과정에서 각 유저는 friends 키를 가지게 되었으므로

해당 유저 사전의 friends 키에 해당하는 Value의 길이(즉, 리스트)
→ 문자

모든 유저의 Value의 길이를 모두 합한 것

→ 분모

03 중심성의 이론적 배경과 계산 알고리즘

1. 연결 중심성 (degree centrality)

```
def degree_centrality(users, user_id):
    numerator = len(users[user_id][“friends”])
    denominator = 0
    for user in users:
        denominator += len(user[“friends”])
    return numerator / denominator
```

users는 아까 만들었던 데이터. user_id의 순서대로
users 리스트의 각 사전으로 존재하므로
input parameter로 정의하였음
당연히 numerator는 분자, denominator는 분모

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)



03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

덜 효율적이지만 이해하기 쉬운 책에 있는 알고리즘 사용

최단 경로를 구하는 알고리즘인 Breadth-First-Search (BFS)

Depth-First-Search (DFS) 등 다른 알고리즘은 프레드님께 ^-^

본 세션은 BFS로 진행!

이름에서 주는 느낌처럼 옆으로 가지치기를 하는 것!

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

그래서 첫 번째로 할 수 있는 것은 임의의 두 사람이 주어졌을 때 그들 간의 최단 경로를 모두 구하는 것이다. 최단 경로를 효율적으로 구해 주는 복잡한 방법들이 많이 있지만, 이 책에서는 덜 효율적이더라도 훨씬 이해하기 쉬운 알고리즘을 사용할 것이다. ‘Breadth-first search’라고도 알려져 있는 이 알고리즘은 그래도 이 책에서는 가장 복잡 중 하나이니, 찬찬히 살펴보도록 하자.



1. 먼저 from user로부터 다른 모든 사용자까지의 친구 목록을 구해주는 친구 목록을 구하는 과정입니다.

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

그래서 첫 번째로 할 수 있는 것은 임의의 두 사람이 주어졌을 때 그들 간의 최단 경로를 모두 구하는 것이다. 최단 경로를 효율적으로 구해 주는 복잡한 방법들이 많이 있지만, 이 책에서는 덜 효율적이더라도 훨씬 이해하기 쉬운 알고리즘을 사용할 것이다. ‘Breadth-first search’라고도 알려져 있는 이 알고리즘은 그래도 이 책에서는 가장 복잡한 알고리즘 중 하나이니, 찬찬히 살펴보도록 하자.

1. 먼저 from user로부터 다른 모든 사용자까지의 최단 경로를 계산해 주는 히

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

그래서 첫 번째로 할 수 있는 것은 임의의 두 사람이 주어졌을 때 그들 간의 최단 경로를 모두 구하는 것이다. 최단 경로를 효율적으로 구해 주는 복잡한 방법들이 많이 있지만, 이 책에서는 덜 효율적이더라도 훨씬 이해하기 쉬운 알고리즘을 사용할 것이다. ‘Breadth-first search’ 래도 이 책에서는 가장 복잡한 알고리즘



03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

최단 경로를 구하는 알고리즘인 Breadth-First-Search (BFS)

Depth-First-Search (DFS) 등 다른 알고리즘은 프레드님께 ^-^

본 세션은 BFS로 진행!

이름에서 주는 느낌처럼 옆으로 가지치기를 하는 것!

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

책은 설명이 영 별로라…… 책 설명을 풀어서 진행

목표는 노드 간 최단 경로 구하기!

유저 A로부터 유저 B까지의 최단 경로를 어떻게 구할 수 있을까?

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

책은 설명이 영 별로라…… 책 설명을 풀어서 진행

목표는 노드 간 최단 경로 구하기!

유저 A로부터 유저 B까지의 최단 경로를 어떻게 구할 수 있을까?

A의 친구(1단계) - A의 친구의 친구(2단계) - … - B(n단계)에서
n이 최소인 것! (아예 도달하지 못하면 말고 ➡ ➡)

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

책은 설명이 영 별로라…… 책 설명을 풀어서 진행

목표는 노드 간 최단 경로 구하기!

유저 A로부터 유저 B까지의 최단 경로를 어떻게 구할 수 있을까?

A의 친구(1단계) - A의 친구의 친구(2단계) - … - B(n단계)에서
n이 최소인 것! (아예 도달하지 못하면 말고 ➡ ➡)

어차피 네트워크 분석에서 핵심은 친구! (어핵친)

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

알고리즘을 한 번 상상해봅시다.

```
>>> pprint(users[0]["shortest_paths"])
{0: [[], ],
 1: [[1]],
 2: [[2]],
 3: [[1, 3], [2, 3]],
 4: [[1, 3, 4], [2, 3, 4]],
 5: [[1, 3, 4, 5], [2, 3, 4, 5]],
 6: [[1, 3, 4, 5, 6], [2, 3, 4, 5, 6]],
 7: [[1, 3, 4, 5, 7], [2, 3, 4, 5, 7]],
 8: [[1, 3, 4, 5, 6, 8],
      [2, 3, 4, 5, 6, 8],
      [1, 3, 4, 5, 7, 8],
      [2, 3, 4, 5, 7, 8]],
 9: [[1, 3, 4, 5, 6, 8, 9],
      [2, 3, 4, 5, 6, 8, 9],
      [1, 3, 4, 5, 7, 8, 9],
      [2, 3, 4, 5, 7, 8, 9]]}
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

알고리즘을 한 번 상상해봅시다.

내가 노드 A의 매개 중심성을 알고 싶으면 A의 친구를 전부 찾고,

그 친구의 친구를 전부 찾고,

그 과정에서 새롭게 출현한 노드에 도달하는 경로가 지금까지
건너간 친구의 총적

(그리고 서로 다른 경로로 온 것은 그 거리를 비교해 더 짧은 것만 최단
경로로 인정해주면 될 것 같다!)

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

이해는 가는데…… 여기서 발생하는 문제 하나.

A의 친구, A의 친구의 친구, …를 어떻게 순서대로 관리?

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

이해는 가는데…… 여기서 발생하는 문제 하나.

A의 친구, A의 친구의 친구, …를 어떻게 순서대로 관리?

큐(queue)의 개념을 알아봅시다.

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

스택 (Stack) VS 큐 (Queue)

둘 다 자료 구조에 해당

스택은 입력을 아래에서부터 위로 쌓아올리는 것
큐는 입력을 앞에서부터 뒤로 줄 세우는 것

스택은 Last In First Out

[맨 위에 자료 쌓기인 push와 맨 위 자료 빼기인 pop]

큐는 First In First Out

[맨 뒤에 자료 줄 세우기인 enqueue / 맨 앞 자료 빼기인 dequeue]

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
from collections import deque
```

deque는 꼭 뒤로만 줄을 세우지 않을 수 있는 양 방향 큐!

deque에 대한 설명을 <https://goo.gl/ycosQr>에서 지금 같이 확인!

친구, 친구의 친구, … 중 아직 경로가 잡히지 않은(즉, 처음 보는 노드)를

큐에 넣고, 큐가 빌 때까지 반복하며 관리해주면

해당 노드의 다른 노드에 대한 최단 경로를 아까 알고리즘으로 도출 가능

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
    # from_user는 users 안에 있던 각 유저의 사전
    shortest_paths_to = { from_user["id"] : [] }
    # 자기 자신에 대한 것은 빈 것으로 미리 추가
    frontier = deque((from_user, friend) for friend in
                     from_user["friends"])
    # 확인해야 하는 (이전 사용자, 다음 사용자) 큐 만들기
    # frontier는 deque[(A, B), (A, C), (A, F)] 형태 (단, 각 노드는 사전)
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
# frontier는 deque[(A, B), (A, C), (A, F)] 형태 (단, 각 노드는 사전)
```

```
while frontier: # 큐가 빌 때까지 반복
```

```
    prev_user, user = frontier.popleft() # 큐의 맨 앞 사용자를 추출
```

```
    user_id = user["id"] # 앞서 정의한 shortest_paths_to 사전의
```

```
    # 키로 활용하기 위해 친구에 해당하는 user_id를 할당
```

```
    # 큐에 사용자를 추가하는 방법을 고려해 보면 이미 해당 id에 대한
```

```
    # 경로가 저장되어 있을 수 있음!
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
... (전략) ...
```

```
while frontier: # 큐가 빌 때까지 반복
    prev_user, user = frontier.popleft()
    user_id = user["id"]
    paths_to_prev_user = shortest_paths_to[prev_user["id"]]
    # 지금 도달한 친구 이전의 친구까지의 경로에 이 친구를 더해야 함
    paths_to_user = [path + [user_id] for path in paths_to_prev_user]
    # 왜 for문이 있을까? - 지금 도달한 친구 이전의 친구까지의 경로가
    # 여러 개일 수 있기 때문
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
... (전략) ...
```

```
while frontier: # 큐가 빌 때까지 반복
    paths_to_user = [path + [user_id] for path in W
                     paths_to_prev_user]
    old_paths_to_user = shortest_paths_to.get(user_id, [])
    # shortest_paths_to 사전에 등록되어 있던, 새로 도달한 친구에
    # 이르는 경로를 가져 옴 (사전의 get 메소드는 Key로 Value 변화)
    # get 메소드에 키 이외에 하나 더 적으면 그건 그 Key가 없을 때
    # 에러를 내지 않고 그 더 적힌 개체를 반환 (이 경우 빈 리스트)
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
... (전략) ...
```

```
while frontier: # 큐가 빌 때까지 반복
    old_paths_to_user = shortest_paths_to.get(user_id, [])
    # 지금 old_paths_to_user로 기존 경로 확인
    if old_paths_to_user:
        min_path_length = len(old_paths_to_user[0])
        # 왜 인덱스 0이 추가되었을까? - 경로가 두 개 이상일 수 있으니까
    else:
        min_path_length = float('inf')
    # 기존 경로가 존재하면 그 길이를, 없으면 무한대를
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
... (전략) ...
```

```
while frontier: # 큐가 빌 때까지 반복
```

```
    new_paths_to_user = [path for path in new_paths_to_user ¶
                         if len(path) <= min_path_length ¶
                         and path not in old_paths_to_user]
```

```
# 기존 경로와 거리가 같거나 짧으면서 원래 경로 리스트에 없던 것
```

```
shortest_paths_to[user_id] = old_paths_to_user + ¶
                             new_paths_to_user
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
... (전략) ...
```

```
while frontier: # 큐가 빌 때까지 반복
    shortest_paths_to[user_id] = old_paths_to_user + W
                                new_paths_to_user
    frontier.extend((user, friend) for friend in user["friends"] W
                    if friend["id"] not in shortest_paths_to)
    # 아직 한 번도 등장하지 않은 친구를 frontier에 추가
```

```
return shortest_paths_to
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
... (전략) ...
```

new_paths_to_user에 해당하는
path들이 더 짧으면?

```
while frontier: # 큐가 빌 때까지 반복
```

```
    shortest_paths_to[user_id] = old_paths_to_user + ¶  
                                new_paths_to_user
```

```
    frontier.extend((user, friend) for friend in user["friends"] ¶
```

```
                    if friend["id"] not in shortest_paths_to)
```

```
    # 아직 한 번도 등장하지 않은 친구를 frontier에 추가
```

```
return shortest_paths_to
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
# 앞의 shortest_paths_to[user_id]에 경로 리스트를 할당할 때  
# 정말 최단 거리에 해당하는 것들만 있게끔 해야  
# 책 안 틀렸는데 틀렸다고 한 것일까봐 조금 걱정되긴 하는데.....
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

```
# 앞의 shortest_paths_to[user_id]에 경로 리스트를 할당할 때  
# 정말 최단 거리에 해당하는 것들만 있게끔 해야  
# 책 안 틀렸는데 틀렸다고 한 것일까봐 조금 걱정되긴 하는데.....
```

```
# 기존 경로 중에 제일 짧은 값과 새 경로 중에 제일 짧은 값 비교 필요  
# old_min_path_length와 new_min_path_length 할당!
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

```
def shortest_paths_from(from_user):
```

기존 경로 중에 제일 짧은 값과 새 경로 중에 제일 짧은 값 비교 필요
old_min_path_length와 new_min_path_length 할당!

```
    if old_paths_to_user:  
        old_min_path_length = len(min(old_paths_to_user, key = lambda x : len(x)))  
    else:  
        old_min_path_length = float("inf")  
  
    # 길지 않은 새로운 경로만 저장  
  
    new_min_path_length = len(min(new_paths_to_user, key = lambda x : len(x)))  
  
    if new_min_path_length < old_min_path_length:  
        shortest_paths_to[user_id] = [path for path in new_paths_to_user #  
                                       if len(path) == new_min_path_length]  
  
    elif new_min_path_length == old_min_path_length:  
        new_paths_to_user = [path for path in new_paths_to_user #  
                            if len(path) == new_min_path_length #  
                                and path not in old_paths_to_user]  
  
        shortest_paths_to[user_id] = old_paths_to_user + new_paths_to_user  
  
    else:  
        pass
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

고치고 나면 마음 한 구석이 편안해집니다.

이제 최단 경로 구하기는 끝! 이제 users 리스트에 반영하면
매개 중심성을 구할 수 있습니다.

for user in users:

```
    user[“shortest_paths”] = shortest_paths_from(user)
```

이제 각 유저 사전마다 shortest_paths라는 Key에 다른 노드로 가는
최단 경로들을 담은 사전이 Value로 추가

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

총 최단 경로의 개수를 구하고, 그 기여를 얼마나 하였는지 확인!

```
for source in users:  
    source_id = source[“id”]  
    for target_id, paths in source[“shortest_paths”].items():  
        # 사전의 items 메소드는 Key와 Value를 뮤어서 보여줌  
        # 책에는 iteritems 메소드인데 Python 3에서는 안 됨
```

03 중심성의 이론적 배경과 계산 알고리즘

2. 매개 중심성 (betweenness centrality)

총 최단 경로의 개수를 구하고, 그 기여를 얼마나 하였는지 확인!

```
for source in users:  
    source_id = source["id"]  
    for target_id, paths in source["shortest_paths"].items():  
        if source_id < target_id:  
            num_paths = len(paths)  
            contrib = 1 / num_paths  
            for path in paths:  
                for id in path:  
                    if id not in [source_id, target_id]:  
                        users[id]["betweenness_centrality"] += contrib
```

03 중심성의 이론적 배경과 계산 알고리즘

3. 근접 중심성 (closeness centrality)

근접 중심성은 해당 사용자가 갖는 다른 모든 사용자와의 최단 거리의 합을 분모로 갖는 수치

⇒ 앞서 구한 최단 거리를 활용하여 최단 거리 합을 구하고 활용!

03 중심성의 이론적 배경과 계산 알고리즘

3. 근접 중심성 (closeness centrality)

근접 중심성은 해당 사용자가 갖는 다른 모든 사용자와의 최단 거리의 합을 분모로 갖는 수치

⇒ 앞서 구한 최단 거리를 활용하여 최단 거리 합을 구하고 활용!

```
def farness(user):
    return sum(len(paths[0]) +
               for paths in user["shortest_paths"].values())
```

```
for user in users:
    user["closeness_centrality"] = 1 / farness(user)
```

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

결국엔 인접행렬과 eigenvector를 구하는 문제

인접행렬은 friendships 리스트에 포함된 튜플을 활용해 만들기

```
import numpy as np
adj_matrix = np.zeros((len(users), len(users)))
# 디폴트로 (유저 수)차 정사각 영행렬을 정의
for i, j in friendships: # friendships = [(0, 1), (0, 2), (7, 8), ...]
    adj_matrix[i][j] = 1
    adj_matrix[j][i] = 1
# 서로 연결된 것에 1 할당 (인접행렬은 방향성 없는 경우 대칭행렬)
```

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

`numpy.linalg.eig`를 활용하면 eigenvector도 간단히!

`numpy.linalg.eig`

`numpy.linalg.eig (a)`

[source]

Compute the eigenvalues and right eigenvectors of a square array.

Parameters: `a : (..., M, M) array`

Matrices for which the eigenvalues and right eigenvectors will be computed

Returns: `w : (..., M) array`

The eigenvalues, each repeated according to its multiplicity. The eigenvalues are not necessarily ordered. The resulting array will be of complex type, unless the imaginary part is zero in which case it will be cast to a real type. When `a` is real the resulting eigenvalues will be real (0 imaginary part) or occur in conjugate pairs

`v : (..., M, M) array`

The normalized (unit "length") eigenvectors, such that the column `v[:,i]` is the eigenvector corresponding to the eigenvalue `w[i]`.

Raises: `LinAlgError`

If the eigenvalue computation does not converge.

See also:

`eigvals` eigenvalues of a non-symmetric array.

`eigh` eigenvalues and eigenvectors of a symmetric or Hermitian (conjugate symmetric) array.

`eigvalsh` eigenvalues of a symmetric or Hermitian (conjugate symmetric) array.

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

numpy.linalg.eig를 활용하면 eigenvector도 간단히!

Examples

```
>>> from numpy import linalg as LA
```

(Almost) trivial example with real e-values and e-vectors.

```
>>> w, v = LA.eig(np.diag((1, 2, 3)))
>>> w; v
array([ 1.,  2.,  3.])
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

```
from numpy import linalg as LA
```

```
w, v = LA.eig(adj_matrix) # 한 줄로 진짜 끝
```

```
>>> w[0]
2.6688229150885681
>>> v[:,0]
array([-0.38578095, -0.51479052, -0.51479052, -0.47331326, -0.23360825,
       -0.15014578, -0.08355213, -0.08355213, -0.07284006, -0.02729295])
>>> -v[:,0]
array([ 0.38578095,  0.51479052,  0.51479052,  0.47331326,  0.23360825,
       0.15014578,  0.08355213,  0.08355213,  0.07284006,  0.02729295])
>>>
```

03 중심성의 이론적 배경과 계산 알고리즘

4. 고유벡터 중심성 (eigenvector centrality)

```
from numpy import linalg as LA
```

```
w, v = LA.eig(adj_matrix)
```

```
for user_id, eigenvector_centrality in enumerate(-v[:,0]):  
    print(user_id, eigenvector_centrality, sep = " : ")
```

```
0 : 0.385780946874  
1 : 0.514790515611  
2 : 0.514790515611  
3 : 0.473313262048  
4 : 0.233608248547  
5 : 0.150145784827  
6 : 0.083552131302  
7 : 0.083552131302  
8 : 0.0728400577961  
9 : 0.0272929527787  
>>>
```

03 중심성의 이론적 배경과 계산 알고리즘

GH_network_data.csv를 지금까지 한 것처럼 전처리?

```
Source_ID,Q1_Target_Id,Q2_Target_Id
0,"1, 16, 17","7, 18, 19"
1,"12, 16, 17","0, 6, 12"
2,"1, 12, 17","7, 18, 4"
3,"8, 5, 16, 11","18, 0, 12, 11"
4,"8, 16, 13","7, 15, 16, 11"
5,"2, 18, 19","2, 7, 8, 4, 15, 6"
6,"18, 1, 8, 5, 16","7, 8, 5, 17, 11"
```

```
with open("GH_network_data.csv", 'r', encoding = 'UTF-8') as f:
    lines = f.readlines()
    users = []
    for line in lines[1:]:
        user = {}
        splitted = line.split(',')
        user["ID"] = int(splitted[0])
        user["Q1_Target_ID"] = [int(item) for item in splitted[1].split()]
        user["Q2_Target_ID"] = [int(item) for item in splitted[2].split()]
        users.append(user)
```

이 부분은 원본 코드와는 다른 내용입니다.

'Q1_Target_ID'와 'Q2_Target_ID'를 제외하고 앞의 수가만 추출

03 중심성의 이론적 배경과 계산 알고리즘

GH_network_data.csv를 지금까지 한 것처럼 전처리?

```
Source_ID,Q1_Target_Id,Q2_Target_Id
0,"1, 16, 17","7, 18, 19"
1,"12, 16, 17","0, 6, 12"
2,"1, 10, 17, 17, 10, 4"
```

숙제¹, 叔齊

/-째/

명사

중국 은(殷)나라의 현인(賢人)(?~?). 이름은 지(智), 자는 공달(公達). 주(周)나라 무왕(武王)이 은나라 주왕(紂王)을 치려고 할 때, 형 백이(伯夷)와 함께 간하였으나 받아들여지지 않고 무왕의 천하가 되자 수양산(首陽山)에 들어가 굶어 죽었다고 함.

```
splitter = re.compile(',')
user["ID"] = int(user[0])
for edge in splitter.split(user[1]):
    user["Q1_Target_ID"] = int(edge)
    for edge in splitter.split(user[2]):
        user["Q2_Target_ID"] = int(edge)
        for edge in splitter.split(user[3]):
            user["Q3_Target_ID"] = int(edge)
            Users.append(user)
```

03 중심성의 이론적 배경과 계산 알고리즘

설문조사의 in/out-degree centrality

```
def out_degree_centrality(users, user_id, Q):
    if Q not in [1, 2]:
        print("Q must be 1 or 2")
    else:
        denominator = 0
        if Q == 1:
            numerator = len(users[user_id]["Q1_Target_ID"])
            for user in users:
                denominator += len(user["Q1_Target_ID"])
            return numerator / denominator
        else:
            numerator = len(users[user_id]["Q2_Target_ID"])
            for user in users:
                denominator += len(user["Q2_Target_ID"])
            return numerator / denominator

def in_degree_centrality(users, user_id, Q):
    if Q not in [1, 2]:
        print("Q must be 1 or 2")
    else:
        numerator = 0
        denominator = 0
        if Q == 1:
            for user in users:
                if user_id in user["Q1_Target_ID"]:
                    numerator += 1
                    denominator += len(user["Q1_Target_ID"])
            return numerator / denominator
        else:
            for user in users:
                if user_id in user["Q2_Target_ID"]:
                    numerator += 1
                    denominator += len(user["Q2_Target_ID"])
            return numerator / denominator
```

03 중심성의 이론적 배경과 계산 알고리즘

설문조사의 in/out-degree centrality

```
def out_degree_centrality(users, user_id, Q):
    if Q not in [1, 2]:
        print("Q must be 1 or 2")
    else:
        denominator = 0
        if Q == 1:
            numerator = len([user["Target_ID"] for user in users
                           if user["User_ID"] == user_id])
        else:
            numerator = len([user["User_ID"] for user in users
                           if user["Target_ID"] == user_id])
        denominator += len(user["User_ID"])
        return numerator / denominator
```

숙제¹, 叔齊

/-째/

명사

중국 은(殷)나라의 현인(賢人)(?~?). 이름은 지(智), 자는 공달(公達). 주(周)나라 무왕(武王)이 은나라 주왕(紂王)을 치려고 할 때, 형 백이(伯夷)와 함께 간하였으나 받아들여지지 않고 무왕의 천하가 되자 수양산(首陽山)에 들어가 굶어 죽었다고 함.

```
def out_degree_centrality(users, user_id, Q):
    if Q not in [1, 2]:
        return None
    else:
        numerator = 0
        denominator = 0
        for user in users:
            if user["User_ID"] == user_id:
                if Q == 1:
                    numerator += len(user["Target_ID"])
                else:
                    numerator += len([user["User_ID"] for user in users
                                   if user["Target_ID"] == user_id])
                denominator += 1
            else:
                denominator += len(user["User_ID"])
        return numerator / denominator
```


04 네트워크 활용 사례

네트워크와 브랜드

 **Elon Musk**  

Just discovered a great Tesla ad made by 2 recent college grads. I love it! youtube.com/watch?v=KKbRAa...

3:20 PM - 14 Mar 2014



 YouTube @YouTube

1,573 RETWEETS 1,452 FAVORITES



<https://organicmedialab.com/2015/06/23/network-is-eating-the-world-2/>

04 네트워크 활용 사례

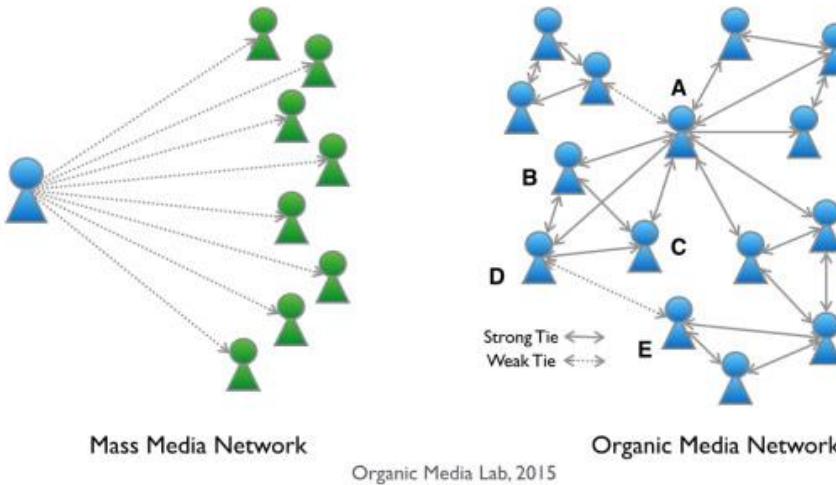
네트워크와 소비자

매개의 주체, 개인

- 주는 대로 받아들이는 대중이 사라지고 스스로 미디어가 되어 네트워크를 형성한 개인(노드)들이 존재하는 상황
- 대중 매체라 불리던 TV나 신문도 이제 나와 같은 하나의 노드에 불과하다 (물론 나보다는 훨씬 목소리가 크다).

구분	산업 사회	정보 사회
중심 개념	생산자 중심	소비자 중심
마케팅 믹스 전략	제품 (Product) 가격 (Price) 유통 (Place) 촉진 (Promotion)	소비자 혜택 (Customer Benefits) 소비자 기회비용 (Cost of Customer) 편리성 (Convenience), 커뮤니케이션(Communication)
미흡하지만 4P->4C etc. 마케팅>믹스 의 변화도~		

매스 미디어/오가닉 미디어 네트워크 (Mass Media or Organic Media Network)



04 네트워크 활용 사례

“세상 참 좁다” - SNS와 바이럴 마케팅

‘여섯 단계의 분리’ (Six degrees of separation)

이 세상에 살고 있는 어떤 두 사람 간에 그들을 연결해 줄 경로(path)가 존재하고 심지
어 이 경로가 평균 6개의 링크(지인관계)로 이루어져 있다는 주장
페이스북에서 임의의 두 사람간의 거리(경로 길이)는 평균 4.7

[정의]

- **바이럴 마케팅**은 소셜 미디어를 통해 거미줄처럼 네트워크되어 있는 소비자들에게 바이러스처럼 빠르게 확산되는 새로운 마케팅 현상.
- 누리꾼이 이메일이나 SNS 등 전파 가능한 매체를 통해 자발적으로 어떤 기업 또는 제품을 홍보하도록 유도하는 마케팅 기법.

[콘텐츠 지속성과 네트워크]

- 콘텐츠가 너무 많기 때문에, 연결되지 않은 콘텐츠는 눈에 보이지 않는다.
- 콘텐츠가 지속되려면 ‘관계’를 만들어 주어야 한다. (오가닉 마케팅의 견지)

어떻게 한 사람으로부터 시작해서 네트워크 전체를 감염시킬 수 있을까?

04 네트워크 활용 사례

“세상 참 좁다” - SNS와 바이럴 마케팅

EX. 네트워크 특성에 기초한 마케팅 타겟 설정

변형된 와츠와 스트로가츠 모델
(A Variation of Watts-Strogatz Model)

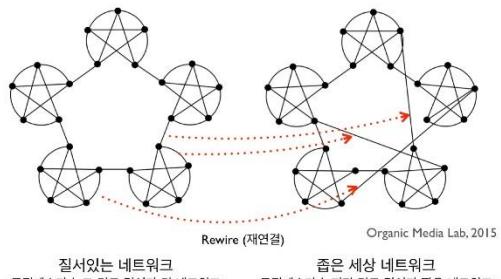
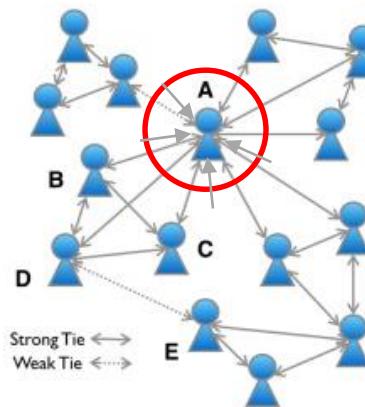


Image Source: Fang et al., "Balancing exploration and exploitation through structural design," Organization Science, 2010.

약한 연결(Weak Tie)은
메시지를 멀리 퍼지게 한다



허브(Hub)는
한번에 많은 메시지를 보낸다

소셜 네트워크는 수많은 무리들이 전세계에 흩어져 있지만
(허브에 의해) **널리** (사회적 다리에 의해) **멀리** 메시지가 전파될 수 있는 구조를 가지고 있다

타겟팅이 더 궁금하다면? <http://m.itdaily.kr/news/articleView.html?idxno=84748>

04 네트워크 활용 사례

바이럴 마케팅 성공 사례



바이럴마케팅 성공사례 ② 코카콜라 댄스 자판기

영상 출처: 유튜브



유튜브 110만건 이상
폭발적인 조회수 기록



바이럴마케팅 성공사례 ① The T-Mobile Dance

영상 출처: 유튜브



바이럴마케팅 대표 성공사례
바이럴마케팅의 의미를 보여줌

바이럴마케팅 성공사례 ③ 하이네켄 광고

영상 출처: 유튜브



인상을 강렬하게 남기는
재밌고 즐거운 광고

<http://originalm.history.com/16>

04 네트워크 활용 사례

상권 분석

12

한국의류학회지

Vol. 40 No. 2, 2016

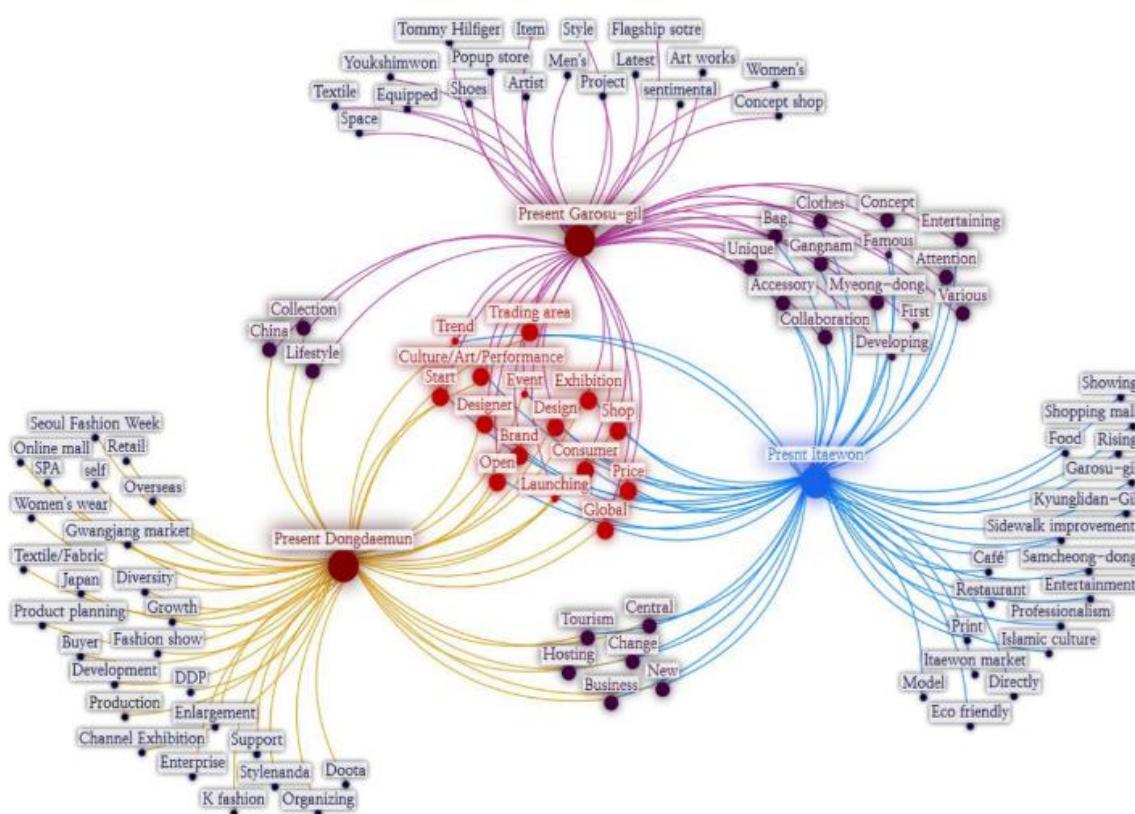


Fig. 2. Relationship between three trading areas in the present (2013-2014).

04 네트워크 활용 사례

충성 고객과 이탈 고객

經營科學
第26卷 第1號
2009年 3月

183

사회 네트워크 분석을 이용한 충성고객과 이탈고객의
구매 특성 비교 연구*

김재경** · †최일영** · 김혜경** · 김남희**

Social Network Analysis to Analyze the Purchase Behavior Of
Churning Customers and Loyal Customers

Jae Kyeong Kim** · †Il Young Choi** · Hyea Kyeong Kim** · Nam Hee Kim**

충성고객 네트워크는 구매 제품의 종류가 10개 이상~15개 미만일 때 높은 연결중심성을 보였으나 이탈고객 네트워크는 구매 제품 종류의 수에 관계없이 연결중심성이 높았다. 또한 이탈고객 네트워크가 충성고객 네트워크보다 밀도가 높게 나타났다. 이는 이탈 고객들은 광고, 쿠폰 등의 마케팅 활동과 관련된 특정 제품을 구매함으로써 고객간 유사성이 높은 반면에, 충성 고객들은 자신에게 필요한 품목 중심으로 구매함으로써 고객간 유사성이 낮다고 해석할 수 있다. 또한 충성고객과 이탈고객의 구매 제품을 분석한 결과 이탈고객은 충성고객보다 메이크업 제품을 더 많이 구매함을 알 수 있었다.

04 네트워크 활용 사례

■ 제품 가치와 네트워크

Gal Oestreicher-Singer, Barak Libai, Liron Sivan, Eyal Carmi, & Ohad Yassin

The Network Value of Products

Traditionally, the value of a product has been assessed according to the direct revenues the product creates. However, products do not exist in isolation but rather influence one another's sales. Such influence is especially evident in e-commerce environments, in which products are often presented as a collection of web pages linked by recommendation hyperlinks, creating a large-scale product network. The authors present a systematic approach to estimate products' true value to a firm in such a product network. Their approach, which is in the spirit of the PageRank algorithm, uses available data from large-scale e-commerce sites and separates a product's value into its own intrinsic value, the value it receives from the network, and the value it contributes to the network. The authors demonstrate their approach using data collected from the product network of books on Amazon.com. Specifically, they show that the value of low sellers may be underestimated, whereas the value of best sellers may be overestimated. The authors explore the sources of this discrepancy and discuss the implications for managing products in the growing environment of product networks.

Keywords: product value, cross-selling, electronic commerce, recommendation systems, social networks

04 네트워크 활용 사례

■ 네트워크와 브랜드 가치

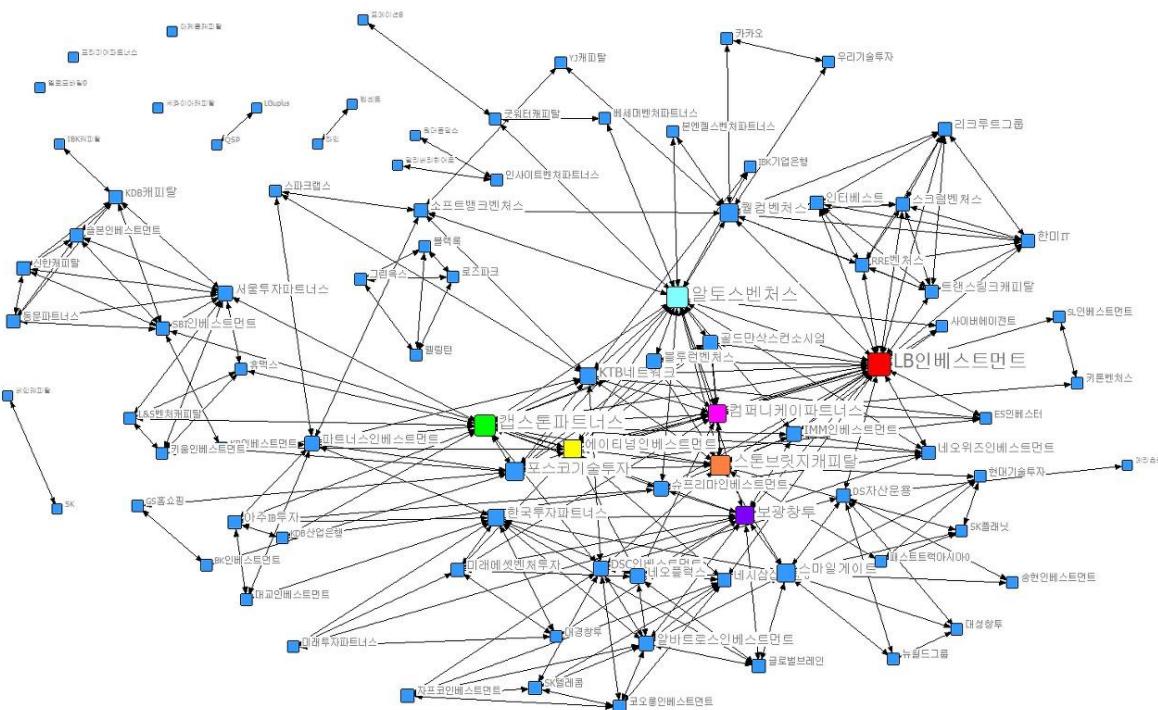
브랜드는 '인식'의 싸움이다

구매자가 그 브랜드의 상품, 유통 채널, 직원, 그리고 커뮤니케이션을 접하면서
오랜 기간에 걸쳐 형성한 긍정적 또는 부정적 인상의 결정체
-장 노엘 캐퍼러-

브랜드란 사업자가 제공하는 것이 아니라 **고객, 시장을 통해 '구축된다'** 는 관점
브랜드의 발달, 성장, 진화, 소멸의 사이클은 사용자의 경험에서 시작된다

04 네트워크 활용 사례

네트워크와 벤처캐피탈



- 벤처 캐피탈 (Venture Capital)
 - 벤처기업에 주식투자 형식으로 투자하는 기업 또는 기업의 자본
 - 장래성과 수익성에 주목하여 투융자

04 네트워크 활용 사례

Journal – Network & VC

Whom You Know Matters: Venture Capital Networks and Investment Performance

Yael V. Hochberg, Alexander Ljungqvist, Yang Lu

<Abstract>

Many financial markets are characterized by strong relationships and networks rather than arm's-length, spot-market transactions. We examine the performance consequences of this organizational choice in the context of relationships established when VCs syndicate portfolio company investments, using a comprehensive sample of U.S. based VCs over the period 1980 to 2003. VC funds whose parent firms enjoy more influential network positions have significantly better performance, as measured by the proportion of portfolio company investments that are successfully exited through an initial public offering or a sale to another company. Similarly, the portfolio companies of better networked VC firms are significantly more likely to survive to subsequent rounds of financing and to eventual exit. The magnitude of these effects is economically large, and is robust to a wide range of specifications. Once we control for network effects in our models of fund and portfolio company performance, the importance of how much investment experience a VC has is reduced, and in some specifications, eliminated. Finally, we provide initial evidence on the evolution of VC networks.

04 Journal – Network & VC

■ 논문 소개 및 목적 - Syndicate ⇔ Networking

- Networks are widespread in many financial markets
 - VCs tend to **syndicate** their investments with other VCs,
Rather than investing alone(Lerner(1994a))
 - **신디케이트 (Syndicate)**
 - 일반적으로는 기업연합을 말하는 것이지만 증권용어로는 주식이나
공사채 등의 유가증권 발행 시 그 인수를 위하여 결성되는 인수단
 - 인수 증권의 판매력 확대와 위험분산
- EX) 양질의 딜 소싱을 하는 VC + 벤처기업 육성을 잘하는 VC
=> Networking이 매우 중요

04 Journal – Network & VC

■ 논문 소개 및 목적 – 네트워킹의 장점

- 1) 미래의 호혜성
- 2) 리스크 감소 + 정보 증가
→ 더 좋은 투자처 발굴
- 3) 포트폴리오 다양화 가능성
→ 각 VC 기업이 지닌 전문가들의 협업
→ 섹터 바운더리를 확장할 수 있음
- 4) 포트폴리오 기업들의 성장 가능성 up

Network =



- In the VC market,
greater centricity may translate into better access to information, deal flow,
deeper pools of capital, expertise, contacts, and so on.

그러나,

네트워킹이 만연하다~ 정도의 논문은 많지만
조직화된 행동의 성과, 결과와 관련한 연구는 부족

04 Journal – Network & VC

Network Analysis Methodology

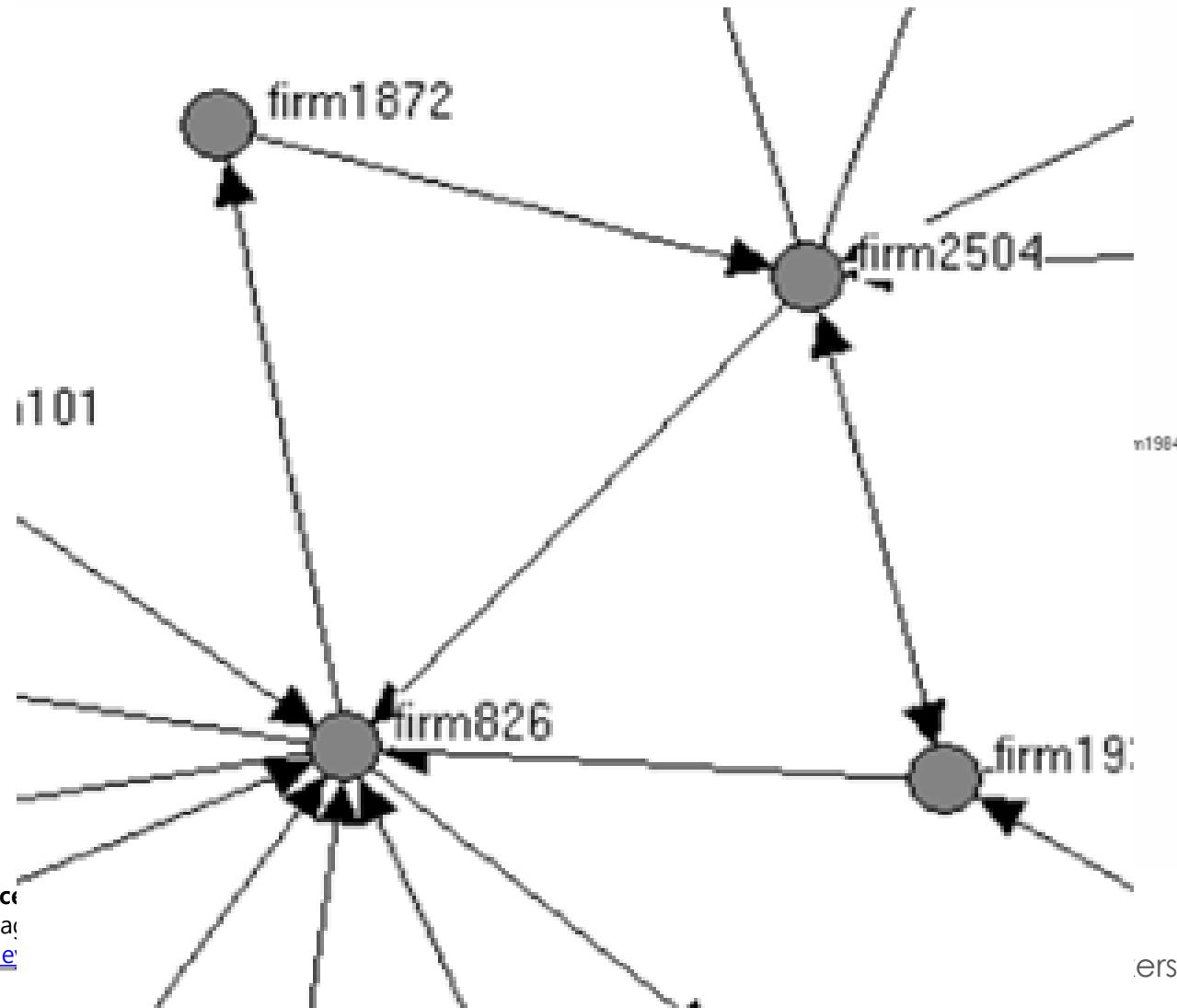
Graph Theory(그래프 이론)

- 변수 상호간의 정상적인 의미에서의 그래프를 다루는 것이 아니라, 특수한 의미의 그래프를 다루는 수학적 방법이다.
- 점과 그것들을 연결하는 선에 의해 구성된다.
선이 점과 연결되는 방식, 순환의 방식, 선이 점과 접하거나 분리하는 방식, 그래프가 하위 그래프를 갖는가 등을 다룬다.
- 점은 개인이나 집단이고 선은 그들 간의 사회적 관계의 망이 된다.

사회학사전, 2000. 10. 30., 사회문화연구소

04 Journal – Network & VC

Network of Biotech VC firms, 1990-1994



04 Journal – Network & VC

II. Sample & Data

VC fund & 포트폴리오 기업 두 가지 관점

1. Sample: US based VCs over the period 1980 to 2003

2. Fund의 라이프사이클에 따른 분류

- 벤처 투자 기본 투자 기간이 미국에선 약 10년 가량
 - 1980년(벤처투자 개념 등장 시기)~1999년 까지 생성된 펀드
 - 2003년부터 성과 측정 -> 최소 4년의 성장 기간 제공
- ➔ VC 산업의 유동성 반영하기 위함

04 Journal – Network & VC

II. Sample & Data

Fund characteristics

fund size (\$m)

sequence number

first fund (fraction, %)

seed or early-stage fund (fraction, %)

Fund performance

exit rate (% of portfolio companies exited)

IPO rate (% of portfolio companies sold via IPO)

M&A rate (% of portfolio companies sold via M&A)

dollar exit rate (% of invested \$ exited)

dollar IPO rate (% of invested \$ exited via IPO)

dollar M&A rate (% of invested \$ exited via M&A)

Fund parent's experience (as of vintage year)

days since parent's first investment

no. of rounds parent has participated in so far

aggregate amount parent has invested so far (\$m)

no. of portfolio companies parent has invested in so far

Network measures (as of vintage year)

outdegree

indegree

degree

betweenness

eigenvector

Competition

VC inflows in fund's vintage year (\$bn)

Investment opportunities

average P/E ratio in fund's first 3 years

average B/M ratio in fund's first 3 years

VC fund & 포트폴리오 기업 두 가지 관점

A. Fund Characteristics

B. Measuring Fund Performance

익명 혹은 통합된 IRR 대신 Exit(M&A와 IPO)으로

C. Company-Level Performance Measure

다음 라운드에서의 생존은 잠정적인 성공 시그널로 이해

D. VC firm Experience

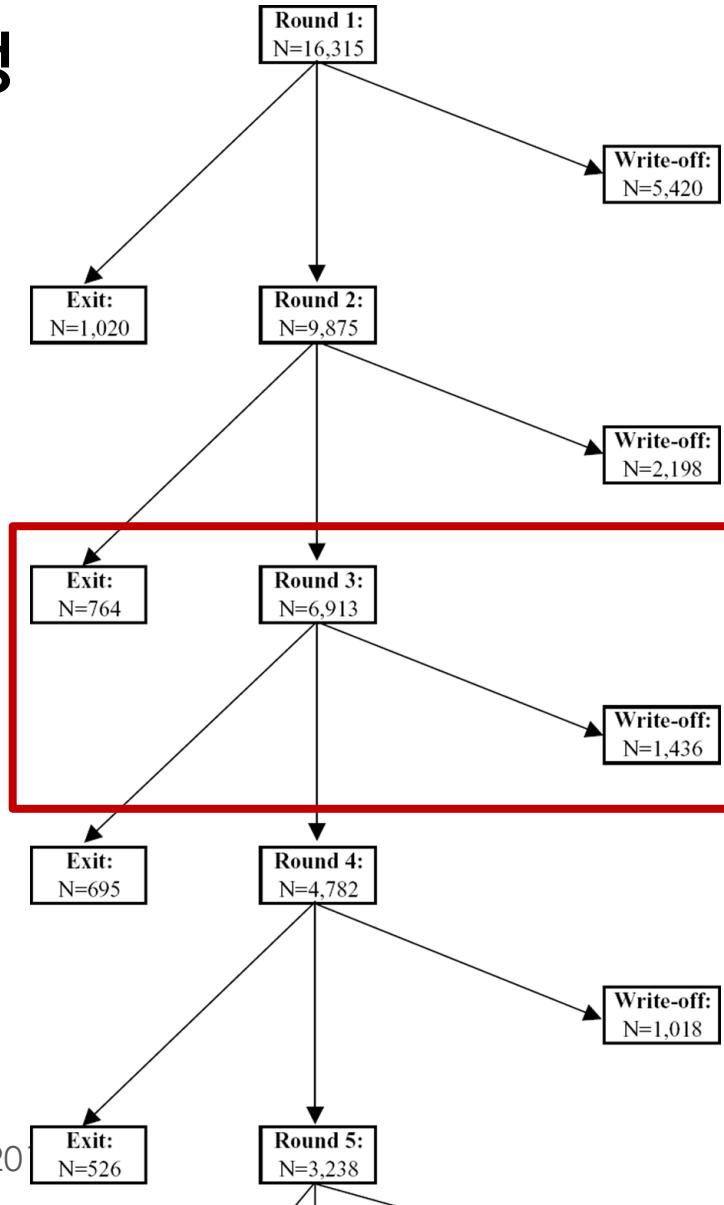
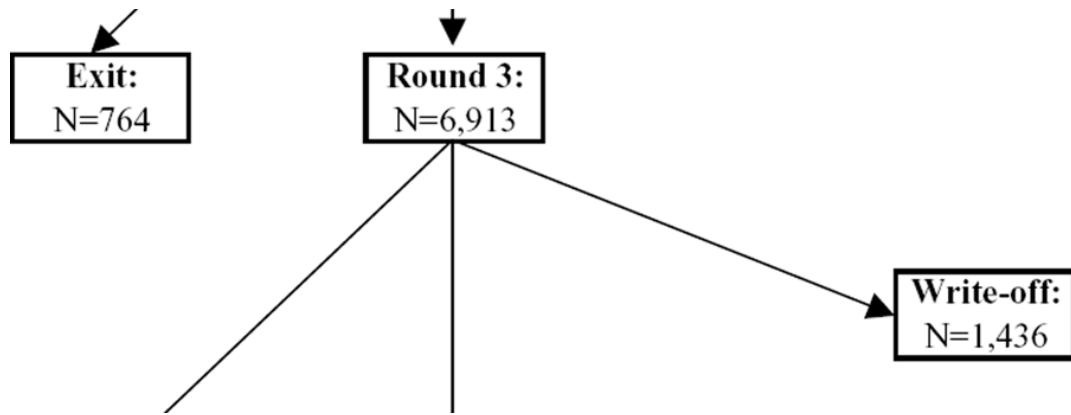
E. Network Measures

F. Competition for Deal Flow and Investment Opportunities

B/M(low book to market ratio): 반비례관계 P/E ratio

04 Journal – Network & VC

II. Sample & Data : 성과 측정



04 Journal – Network & VC

II. Sample & Data

E. Network Measures

Density of ties (% of theoretical max.)			
undirected ties	directed ties		
mean	s.d.	mean	s.d.
3.7	19.0	0.8	9.0
3.7	18.8	0.7	8.6
3.5	18.4	0.7	8.3
3.6	18.7	0.7	8.2
3.5	18.4	0.7	8.1
3.5	18.3	0.7	8.0
4.1	19.9	0.7	8.3
4.0	19.6	0.8	8.8
4.1	19.9	0.8	9.1
4.2	20.0	0.9	9.4
4.5	20.6	1.0	9.8
4.5	20.7	1.0	9.9
4.4	20.4	1.0	10.0
4.2	20.0	1.0	9.9
3.9	19.3	1.0	9.7
2.8	16.5	0.7	8.3
2.2	14.7	0.6	7.4
1.8	13.4	0.5	6.8
1.5	12.2	0.4	6.2
1.4	11.6	0.3	5.7
1.2	11.1	0.3	5.4
1.2	11.0	0.3	5.4
1.2	10.8	0.3	5.3
1.2	11.0	0.3	5.3

Table I. Descriptive Statistics (continued)

	No.	Mean	Std. dev.	Min	Median	Max
Network measures (as of vintage year)						
outdegree (Lead)	3,469	1.203	2.463	0	0.099	22.91
indegree	3,469	1.003	1.671	0	0.210	13.54
degree	3,469	4.237	6.355	0	1.245	41.29
betweenness	3,469	0.285	0.750	0	0.004	7.16
eigenvector	3,469	3.742	5.188	0	1.188	30.96
Competition						
VC inflows in fund's vintage year (\$bn)	3,469	23.842	29.349	2.295	6.474	84.632

04 Journal – Network & VC

II. Sample & Data

E. Network Measures

- Fund의 절반 이상이 신디케이트 펀드이지만 매우 낮은 연결 중심성을 보임
→ 소수의 VC들만 참여하는 배타적이고 안정적인 관계
- Centrality Variation
→ never syndicated VC (1995-1999) 존재 (10.3%, total: 1,820 VCs)

Degree / Closeness / Betweenness

중심성 종류에 따라 미묘하게 달라지는 의미를 비교해봅시다.

04 Journal – Network & VC

I. Network Analysis Methodology

1. Degree Centrality (연결중심성)

- ① 네트워크에서 액터가 관계 맺고 있는 수를 측정한다
 $\sum P_{ij}$ = 신디케이트 관계가 몇 회 있었는가

- ② 더 많은 VC들과 연결 관계를 가지면 우위를 점한다
- 노드가 많을 수록 정보나 deal flow*에 있어서 덜 의존적
 - wider range of contacts, expertise, and pools of capital

*Deal Flow : rate at which they receive business proposals/investment offers

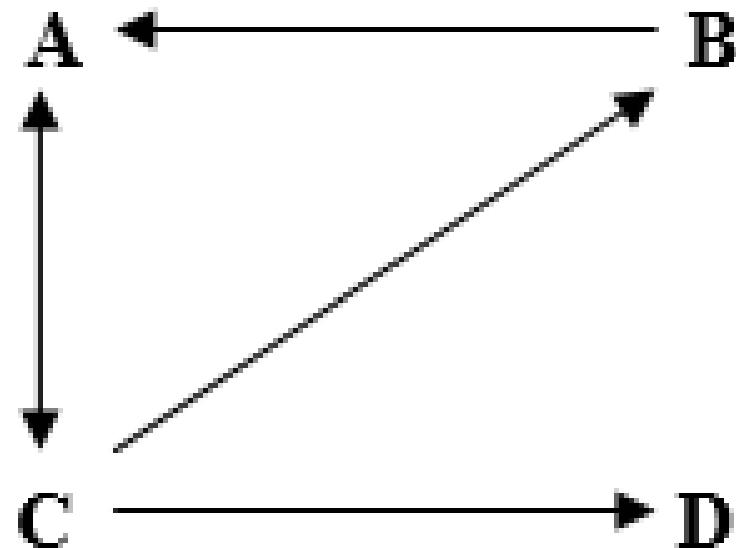
- ③ Normalization
- 시간에 따라 변화하는 업계 특성을 반영
 - 중심성 / (n-1)
- n-1 : maximum possible degree in an n-actor network

04 Journal – Network & VC

I. Network Analysis Methodology

1. Degree Centrality (연결중심성)

- 4 Undirected / Directed
Indegree ←
Outdegree →



04 Journal – Network & VC

Network of Biotech VC firms, 1990-1994

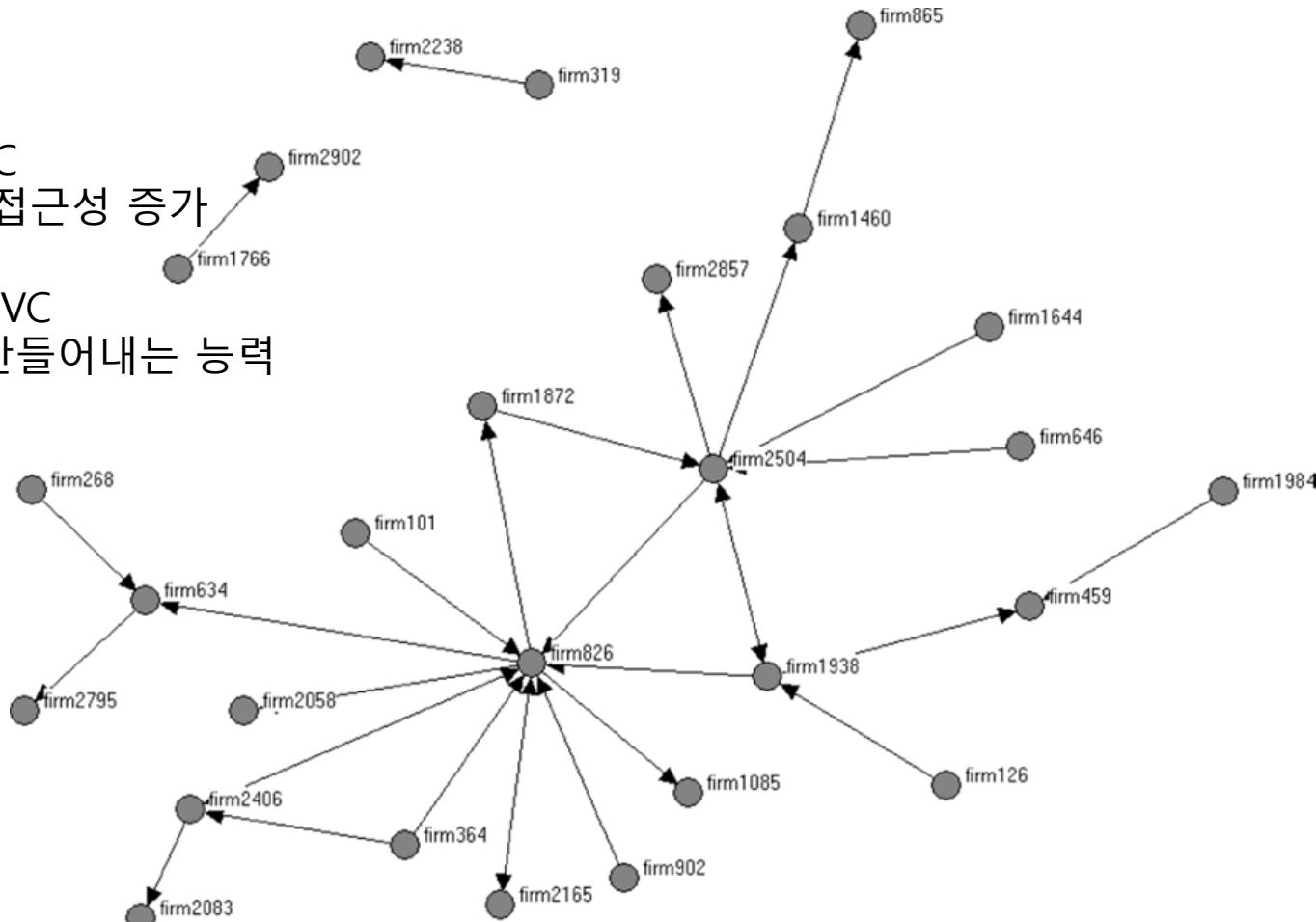
Directed Model

Indegree: 제안 받은 VC

← 정보 및 투자 기회 접근성 증가

Outdegree: 제안 하는 VC

→ 공동 투자 기회를 만들어내는 능력
측정의 수단



04 Journal – Network & VC

I. Network Analysis Methodology

2. Between Centricity (매개중심성)

“Intermediary”

능력자 VC들을 결집시키거나, 직접적으로 관련 없던 투자 기회를 제안하는 매개자 역할의 VC를 측정하는데 활용

→ 측정 결과, 성과에 미치는 영향은 미미한 것으로 나타남

Having the ability to act as a broker between other VCs (betweenness) has a smaller effect, with a one-standard-deviation increase in this centrality measure being associated with only a one percentage point increase in fund performance. This will prove to be true throughout our analysis, suggesting that indirect relationships (those requiring intermediation) play a lesser role in the venture capital market. (p.21)

04 Journal – Network & VC

I. Network Analysis Methodology

3. Closeness

중요한 노드일수록 다른 노드까지 도달하는 경로가 짧을 것이라는 가정

Ex: 사회문화 A도시 vs B도시

→ 액터의 **Quality**

4. Eigenvector Centrality

- 연결된 다른 액터들의 중심성을 **가중치**로 계산
 VC_i 의 아이겐벡터 중심성 = $\sum P_{ij} * evj$
- A노드와 연결된 B노드가 많은 엣지(연결관계)를 가지고 있으면 A노드의 아이겐벡터 중심성이 높아진다

연결 중심성이 높아도 연결된 각 노드의 영향력이 낮으면
(아이겐벡터 중심성이 낮으면) 영향력이 낮다

04 Journal – Network & VC

III. Fund Level Analysis

C. The effect of Firm Networks on Fund performance

Table IV. Network measures X (The Effect of Firm Experience on Fund Performance)

Diagnostics

Adjusted R^2

18.4 %	18.0 %	18.8 %	18.0 %
--------	--------	--------	--------

Test: all coefficients = 0 (F)

44.2 ***	42.7 ***	44.3 ***	42.5 ***
----------	----------	----------	----------

No. of observations

3,105	3,105	3,105	3,105
-------	-------	-------	-------

Table V. Network measures O (The Effect of Firm Networks on Fund Performance)

Network measures

outdegree

0.006 ***

0.002

결정계수 상승

indegree

0.013 ***

0.003

degree

0.003 ***

0.001

betweenness

0.013 **

0.006

eigenvector

0.004 ***

0.001

Diagnostics

Adjusted R^2

18.9 %	19.1 %	19.1 %	18.8 %	19.1 %
--------	--------	--------	--------	--------

Test: all coefficients = 0 (F)

43.4	44.3	44.1	42.8	44.1
------	------	------	------	------

No. of observations

3,105	3,105	3,105	3,105	3,105
-------	-------	-------	-------	-------

04 Journal – Network & VC

III. Fund Level Analysis

C. The effect of Firm Networks on Fund performance

1) Eigen Vector > degree > indegree 순으로 경제적 효과가 큰 것으로 밝혀짐

(To illustrate, a one-standard-deviation increase in these measures is associated with a more than two percentage point increase in exit rates, all else equal.)

2) Thus a VC benefits from having many ties (degree), especially when the ties involve other well-connected VCs (eigenvector), and from being invited into many syndicates (indegree).

3) Outdegree, 즉 펀드를 리드하는 경우에는 “미래의” 호혜성을 기대하는 것이 적합. 상대적으로 경제적 효과 적음

04 Journal – Network & VC

IV. Company Level Analysis

C. Portfolio Company Exit

- 평균 Exit 기간 24분기
- 아이겐벡터가 가장 높은 성과를 보임
"Lead VC의 아이겐벡터 중심성의 std가 1 증가하면 2분기가 단축됨"
 - 영향력 높은 VC들과 관계를 맺고 있는 VC의 포트폴리오 기업들은
 - Exit 속도가 빨라진다.

Outdegree, indegree, degree 중심성이 높은 VC도 약 1분기 가량 단축

* Robust 통계량 : 이상 값에 영향을 적게 받는 통계량

ex: 5명의 마을 주민 월급 평균 13만원 + 빌게이츠가 이사옴 → 이상값(outlier) 발생
→ 평균 대신 중앙값 사용

04 Journal – Network & VC

VII. Conclusion

V. Further Robustness Tests

VI. How do VC Firms Become Networked?

의의

벤처캐피탈 산업에서 조직화된 형태로 네트워크가 가지는 경제적 가치를 분석한 첫 사례

활용 가능성

- 다른 금융 업계의 네트워킹 효과 분석
- 앞서 살펴본 다양한 마케팅 사례에도 적용 가능

한계

VCs likely benefit from personal network ties which we have not so far taken account of.
More broadly, what determines a VC's choice whether or not to network?
What are the costs associated with becoming well-networked?
And how does one form relationships with influential VCs in the network?

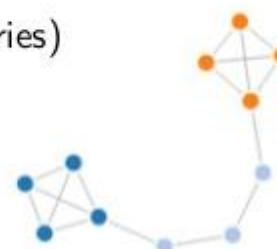
05 NetworkX

NetworkX : Python Library

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

Features

- ① 무방향성, 방향성, 다중그래프 등의 데이터 구조
- ② Nodes can be "anything" (e.g. text, images, XML records)
- ③ Edges can hold arbitrary data (e.g. weights, time-series)
- ④ 표준적인 그래프 알고리즘.
- ⑤ BSD 라이센스
- ⑥ Well tested: more than 1800 unit tests
- ⑦ PYTHON LIBRARY!



05 NetworkX

Hello Network

다운로드:

<https://pypi.python.org/pypi/networkx/>

pip install networkx

(Just using Anaconda)

```
>>>import networkx as nx  
>>>import matplotlib.pyplot as plt  
>>>import numpy as np
```

05 NetworkX

Hello Network

네트워크 만들기:

```
grp= nx.Graph()
```

네트워크 데이터 불러오기(GML Format):

```
grp_loaded= nx.read_gml('설문조사 결과.gml')
```

05 NetworkX

GML Format

```
1  graph [  
2      directed 1  
3      node [  
4          id 0  
5          label "0"  
6      ]  
7      node [  
8          id 1  
9          label "1"  
10     ]  
11     node [  
12         id 2  
13         label "2"  
14     ]
```

```
159    edge [  
160        source 5  
161        target 19  
162    ]  
163    edge [  
164        source 6  
165        target 18  
166    ]  
167    edge [  
168        source 6  
169        target 1  
170    ]
```

05 NetworkX

Hello Network

노드 추가:

```
grp.add_node('h', author= 'team_5')
grp.add_nodes_from(['a', 'b', 'c', 'd', 'e', 'f', 'g'])
```

엣지 추가:

```
grp.add_edge('a', 'b', weight= 0.5, color= 'red')
edges_to_add= [('b', 'c'), ('c', 'd'), ('b', 'd'), ('b', 'g'), ('e', 'b'), ('e', 'f'),
('e', 'g'), ('g', 'c'), ('g', 'd')]
grp.add_edges_from(edges_to_add)
```

Can add information about nodes and edges

05 NetworkX

Hello Network

*우리가 가지고 있는 설문조사 데이터는 csv 형식입니다.
설문조사 데이터를 활용할 때는 NetworkX에 들어갈 수 있는 list
형식으로 바꿔 줍시다!



Data_Structure.py

05 NetworkX

Hello Network

```
import csv
import numpy as np
import os
os.chdir('C:\\\\Studying\\\\GrowthHackers\\\\0905 Network')

def data_making(filename):
    with open(filename, 'rt', newline= '') as f:
        data= csv.reader(f, delimiter= ',')
        #open csv file. 'data' is iterating type, but not a list.
        next(data, None)      #Deleting Index
        node= []
        q1_edge= []
        q2_edge= []
        for row in data:
            answer_list= [val.split(', ') for val in row[1:]]
            node.append(row[0])
            q1_edge.extend([(row[0], i) for i in answer_list[0]])
            q2_edge.extend([(row[0], i) for i in answer_list[1]])
    return (node, q1_edge, q2_edge)
```

05 NetworkX

Hello Network

```
['Source_ID', 'Q1_Target_Id', 'Q2_Target_Id']  
[ '0', '1, 16, 17', '7, 18, 19' ]  
[ '1', '12, 16, 17', '0, 6, 12' ]  
[ '2', '1, 12, 17', '7, 18, 4' ]  
[ '3', '8, 5, 16, 11', '18, 0, 12, 11' ]  
[ '4', '8, 16, 13', '7, 15, 16, 11' ]
```

인덱스가 존재함

1번 node가 선택한 node들.
row[1]이 Q1, row[2]이 Q2
'Str Type'

05 NetworkX

Hello Network

```

import csv
import numpy as np
import os
os.chdir('C:\\\\Studying\\\\GrowthHackers\\\\0905 Network')

def data_making(filename):
    with open(filename, 'rt', newline= '') as f:
        data= csv.reader(f, delimiter= ',')
        #open csv file. 'data' is iterating type, but not a list.
        next(data, None)      #Deleting Index
        node= []
        q1_edge= []
        q2_edge= []
        for row in data:
            answer_list= [val.split(', ') for val in row[1:]]
            node.append(row[0])
            q1_edge.extend([(row[0], i) for i in answer_list[0]])
            q2_edge.extend([(row[0], i) for i in answer_list[1]])
    return (node, q1_edge, q2_edge)

```

₩n이 나오지 않도록

csv 모듈을 이용해서 바로 list로 변환

인덱스는 data의 첫 번째 값임. 그 다음 값부터 iterate시키기

05 NetworkX

Hello Network

```

import csv
import numpy as np
import os
os.chdir('C:\\\\Studying\\\\GrowthHackers\\\\0905 Network')

def data_making(filename):
    with open(filename, 'rt', newline= '') as f:
        data= csv.reader(f, delimiter= ',')
        #open csv file. 'data' is iterating type, but not a list.
        next(data, None)      #Deleting Index
        node= []
        q1_edge= []
        q2_edge= []
        for row in data:
            answer_list= [val.split(',') for val in row[1:]]           Str Type의 선택 결과 데이터들을 리스트로 변환
            node.append(row[0])
            q1_edge.extend([(row[0], i) for i in answer_list[0]])
            q2_edge.extend([(row[0], i) for i in answer_list[1]])
    return (node, q1_edge, q2_edge)

```

row[0](선택 노드), i(피선택 노드)를 튜플로 연결

05 NetworkX

Hello Network

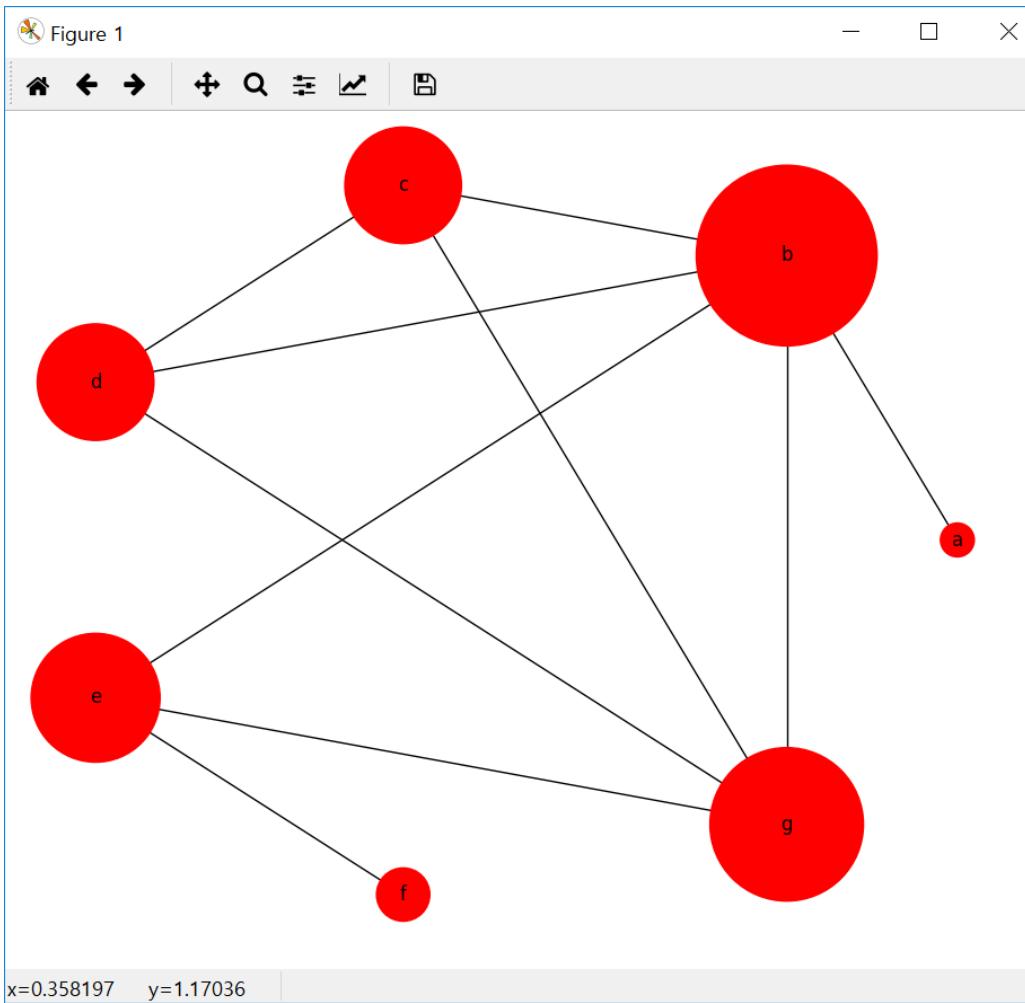
```
pos= nx.shell_layout(grp) # take a location of each nodes
```

```
nx.draw(grp, pos, node_size= [(i*500)**2 for i in nx.pagerank(grp).values()])
nx.draw_networkx_labels(grp, pos)
plt.show()
```

*여기서부터는 예시 데이터를 활용하여 메서드를 알아봅니다. 노드 추가/엣지 추가
파트에서 보았던 데이터를 말이죠!

05 NetworkX

Hello Network



05 NetworkX

Degree Distribution

```
print('Degree: ', grp.degree())      #graph의 차수 도출
```

#degree 함수, density 함수 같은 경우는 그래프 외의 input이 없고, 이 경우 grp.degree()로 써 주어도 좋습니다. 그러나 대부분의 경우 nx.function(grp, **args)로 써 주어야 합니다.

```
deg_sum= float(sum(grp.degree().values()))
```

```
print('Average Node Degree: ', deg_sum/nx.number_of_nodes(grp))  
#평균 차수
```

```
print('Density: ', nx.density(grp)) #Density
```

05 NetworkX

Degree Distribution

```
grpmtx= nx.to_numpy_matrix(grp)
print('Matrix: ', grpmtx)
#graph를 나타내는 numpy matrix 도출
#특히 information 중 weight가 고려됩니다.
#Matrix: [[ 0.  0.5  0.  0.  0.  0.  0. ]
#          [ 0.5  0.  1.  1.  1.  0.  1. ]
#          [ 0.  1.  0.  1.  0.  0.  1. ]
#          [ 0.  1.  1.  0.  0.  0.  1. ]
#          [ 0.  1.  0.  0.  0.  1.  1. ]
#          [ 0.  0.  0.  0.  1.  0.  0. ]
#          [ 0.  1.  1.  1.  1.  0.  0. ]]
```

05 NetworkX

Centrality

```
print('Betweenness: ', nx.betweenness_centrality(grp))  
#Betweenness: {'a': 0.0, 'b': 0.4666, 'c': 0.0, 'd': 0.0, 'e': 0.3333, 'f': 0.0, 'g': 0.1333}
```

```
print('Closeness: ', nx.closeness_centrality(grp))  
#Closeness: {'a': 0.5, 'b': 0.8571, 'c': 0.6, 'd': 0.6, 'e': 0.6666, 'f': 0.4286, 'g': 0.75}
```

```
print('Degree: ', nx.degree_centrality(grp))  
#Degree: {'a': 0.1666, 'b': 0.8333, 'c': 0.5, 'd': 0.5, 'e': 0.5, 'f': 0.1666, 'g': 0.6666}
```

05 NetworkX

Centrality

```
[eigvalue, eigvec]= np.linalg.eig(grpmatx)
```

#Eigenvalues and Eigenvectors, Numpy의 eig 함수는 output으로 eigvalues(N*1 array type), eigvector(N*N array type)를 출력

```
print(eigvalue)
print(eigvec)
print('Eigenvalue_based: ', nx.eigenvector_centrality(grp))
# Eigenvalue_based:
{'a': 0.0757, 'b': 0.5105, 'c': 0.4266, 'd': 0.4266, 'e': 0.3291, 'f': 0.09755, 'g': 0.5019}
```

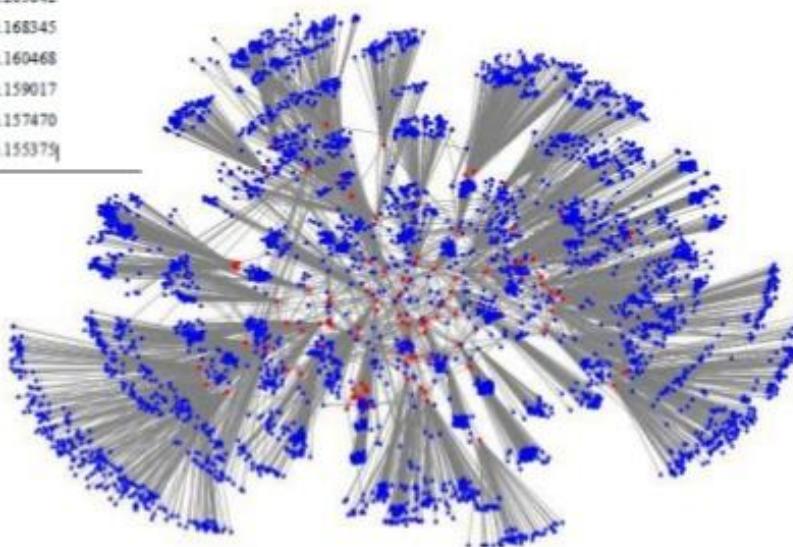
05 NetworkX

Do more with NetworkX

소셜 네트워크 분석

HITS를 이용한 예: Top 10 users with high authorities and hubs on Twitter

Top #	User	Authority	User	Hub
1	Oprah	0.339594	7696000420	0.377777
2	BarackObama	0.294960	ghduluthistic	0.374830
3	ericschmidt	0.249806	ameliaaa	0.228877
4	techable	0.242290	4freeonline	0.212488
5	spunk	0.230091	jegejay102	0.205642
6	DalaLama	0.104141	jocou_siloen	0.168345
7	ev	0.103572	damishazis	0.160468
8	TechCrunch	0.100271	bisafezone	0.159017
9	google	0.099207	musicalgenius5D	0.157470
10	cubrk	0.098969	CharityCause	0.155379



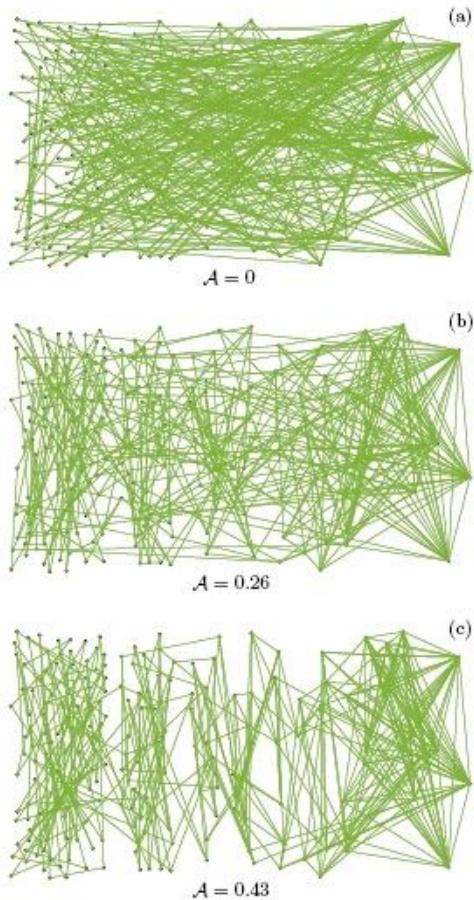
05 NetworkX

■ Node 간의 상관관계: Assortativity

```
print('Assortativity: ', nx.degree_assortativity_coefficient(grp))
# degree가 유사한 node들 간에 연결되는 정도.
# Assortativity: -0.333333333333
# 유사성이 높을 수록, degree가 높은 노드가 degree가 높은 노드들끼
리 연결되어 있음. 0보다 크면 Assortative, 0보다 작으면 Dassortative.
```

05 NetworkX

■ Node 간의 상관관계: Assortativity



$$r = \frac{M^{-1} \sum_i j_i k_i - [M^{-1} \sum_i \frac{1}{2}(j_i + k_i)]^2}{M^{-1} \sum_i \frac{1}{2}(j_i^2 + k_i^2) - [M^{-1} \sum_i \frac{1}{2}(j_i + k_i)]^2}$$

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$$

Copyright © Gilbut, Inc. All rights reserved.

05 NetworkX

결집계수: Clustering Coefficient

```
print('Clustering for one Node: ', nx.clustering(grp))
print('Average Clustering Coefficient: ', nx.average_clustering(grp))

# e의 경우, 자신과 연결된 세 개의 edges 중에서 두 개를 선택할 때 두
개의 엣지로 연결된 노드들과 삼각형을 이룰 확률은 1/3입니다.

# Clustering for one Node: {'a': 0.0, 'b': 0.4, 'cd
```

```
print('Transitivity: ', nx.transitivity(grp))
# (possible Triangles)/(Triads)
# Transitivity: 0.6
```

05 NetworkX

결집계수: Clustering Coefficient

(vii) 결집계수: 결집계수는 다음과 같다. 어느 하나의 노드 i 를 잡았을 때 이 노드의 도수를 k_i 라 하자. 이때 k_i 개의 이웃 노드들끼리 얼마나 잘 연결되어 있는가를 나타내는 양이다. 즉

$$C_i = \frac{2E_i}{k_i(k_i - 1)}. \quad (2.16)$$

여기서 E_i 는 노드 i 의 k_i 의 이웃들 사이에 연결된 링크의 수를 의미하고, $k_i(k_i - 1)/2$ 는 그들간의 가능한 모든 연결선수이다.

$$C = \frac{3 \times \text{number of triangles}}{\text{number of connected triplets of vertices}} = \frac{\text{number of closed triplets}}{\text{number of connected triplets of vertices}}.$$

05 NetworkX

거리와 관련된 지표

```

print("Communicability: ", nx.communicability(grp)['b'])
# 두 노드 간의 거리를 나타내는 지표
# 두 노드 간에 가능한 경로를 모두 고려하여 거리를 계산. 이 때 짧은 경
# 로는 높은 weight를, 긴 경로는 낮은 weight를 가짐
# Communicability: {'a': 2.8527, 'b': 8.8266, 'c': 6.6356, 'd': 6.6356, 'e':
# 5.2216, 'f': 1.4140, 'g': 7.7705}
print('Average Path Length: ', nx.average_shortest_path_length(grp))
# Average Path Length: 1.6667

```

$$G_{pq} = \sum_{k=0}^{\infty} \frac{(A^k)_{pq}}{k!} = e^A.$$

05 NetworkX

견고성(Robustness)/ Connectivity

```
print('Connectivity value: ', nx.node_connectivity(grp))
print('Connectivity for all Nodes: ', nx.all_pairs_node_connectivity(grp))
# 몇 개의 edge를 끊을 때 노드들 간의 연결이 끊어질까?
# Connectivity value: 1
# Connectivity for all Nodes: {... 'b': {'a': 1, 'c': 3, 'd': 3, 'e': 2, 'f': 1, 'g': 4}, ...}
```

05 NetworkX

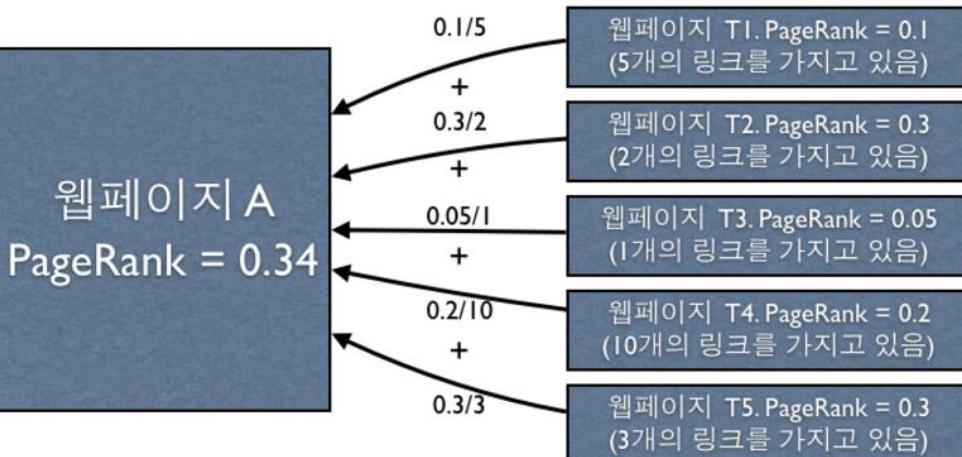
■ Pagerank, HITS Algorithm

```
print('Pagerank: ', nx.pagerank(grp, alpha= 0.85, max_iter= 100,  
tol= 1e-06))  
# Google의 검색 결과 중심성 분석 알고리즘  
# 많은 링크에서 인용될 수록, 중요한 노드로부터 인용되었을 수록 중요한 문서라는 전제  
# Pagerank: {'a': 0.04322, 'b0.2307, 'c': 0.1488, 'd': 0.1488, 'e': 0.1645,  
'f': 0.0680, 'g0.1959}
```

```
print('Hits: ', nx.hits(grp))  
# Hyperlink-Induced Topic Search  
# Hub로부터 직접 인용된 문서를 기반으로 Root 클러스터와 Base 클러스터를 구분
```

05 NetworkX

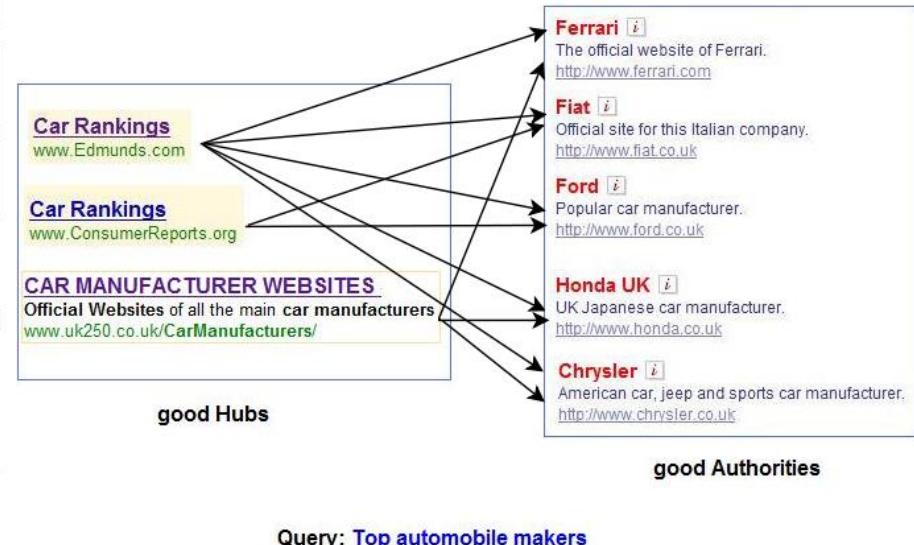
Pagerank, HITS Algorithm



PageRank 알고리즘을 그림으로 표현한 것. Dampen Factor가 있기 때문에 이것과 똑같지는 않지만, 간단하게 표현하면 위와 같다.

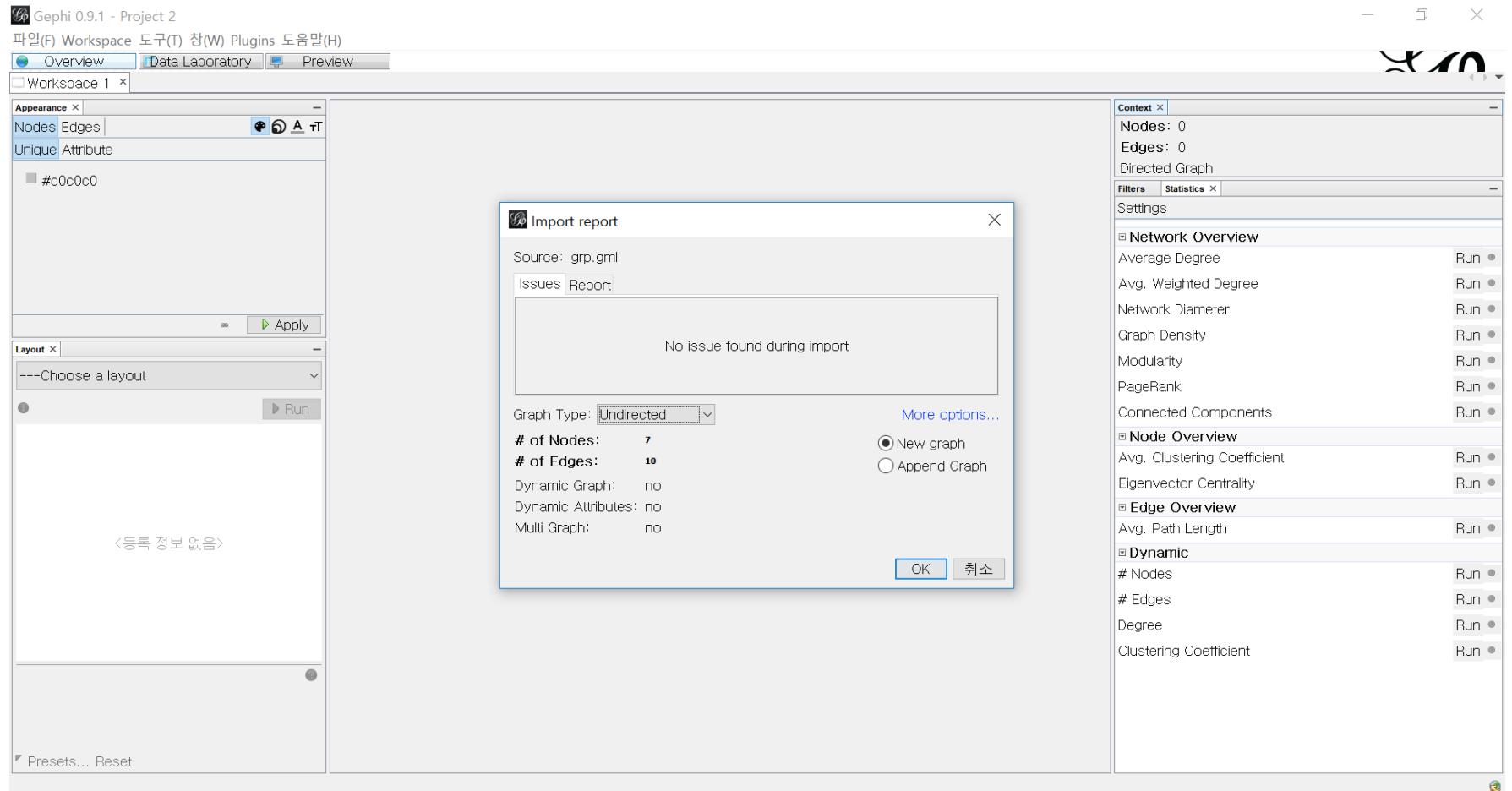
위 그림에서 웹 페이지 A를 가리키는 페이지는 T1, T2, T3, T4, T5의 다섯 개가 있고, 이들을 정규화해서 합한 값이 0.34이므로, A의 '페이지 랭크'는 0.34가 된다. 이 페이지랭크 값은 A가 가리키는 다른 페이지의 PageRank를 계산하는 데 쓰일 것이다. 그럼 T1의 페이지 랭크는 어떻게 구했나? 마찬가지로 T1을 가리키는 다른 페이지들의 PageRank값으로부터 구한다. 이렇게 해서 파고 내려가면 무한히 가게 될 것 같은데, '제한 조건'을 걸면 언젠가는 계산이 끝이 난다. 이러한 방법으로 계산하는 것을 컴퓨터 과학에서는 'recursive(재귀적)'이라고 한다. 즉, PageRank는 재귀 호출 알고리즘이다.

One of the interesting points that he brought up was that the human process should go is more complex than just compare a list of query words against a list of documents and return the matches. Suppose we want to buy a car and type in a general query phrase like "the best automobile makers in the last 4 years", perhaps with the intention to get back a list of top car brands and their official web sites. When you ask this question to your friends, you expect them to be able to understand that automobile means car, vehicle, and that automobile is a general concept that includes vans, trucks, and other type of cars. When you ask this question to a computer that is running a text based ranking algorithm, things might be very different. That computer will count all occurrences of the given words in a given set of documents, but will not do intelligent rephrasing for you. The list of top pages we get back, while algorithmically correct, might be very different than what expected. One problem is that most official web sites are not enough self descriptive. They might not advertise themselves the way general public perceives them. Top companies like Hyundai, Toyota, might not even use the terms "automobile makers" on their web sites. They might use the term "car manufacturer" instead, or just describe their products and their business.



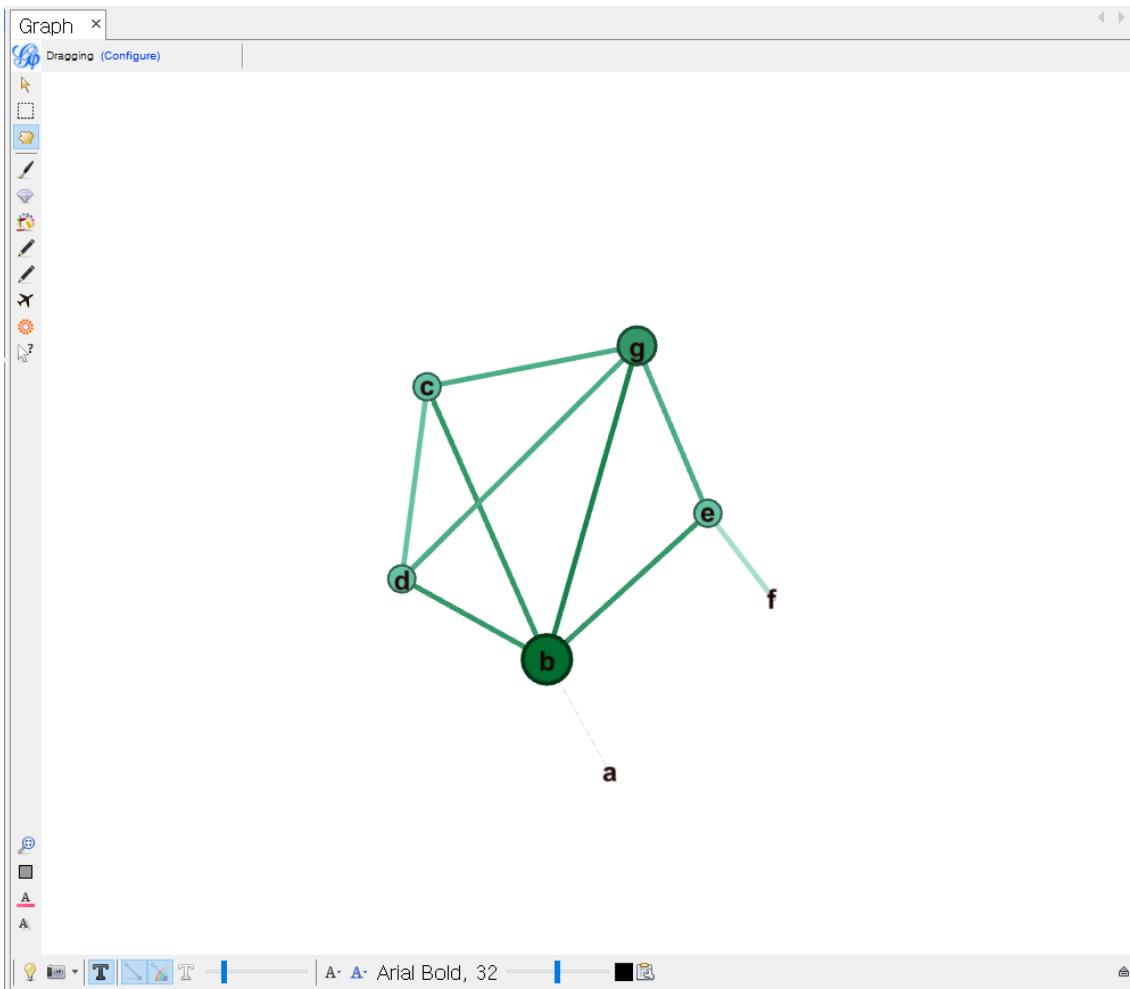
05 Gephi

Load GML File



05 Gephi

Load GML File

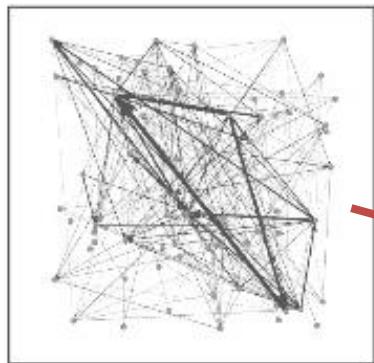


05 Gephi

Layout

You should now see a graph

We imported the "Les Misérables" dataset¹. This is a coappearance network of characters in the novel "Les Misérables" from Victor Hugo.

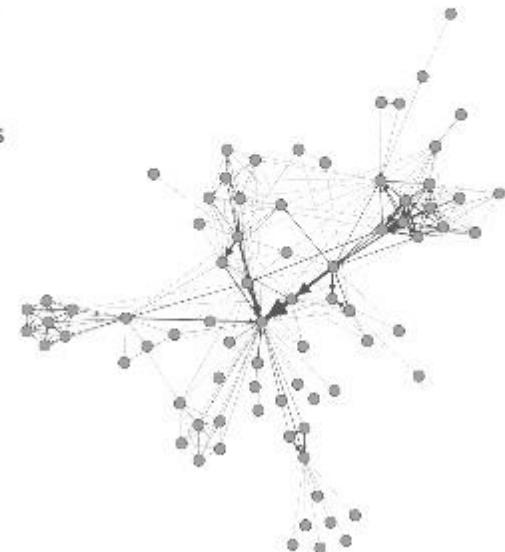


Node position is random at first, so you may see a slightly different re

ForceAtlas layout

Home-brew layout of Gephi, it is made to spatialize Small-World / Scale-free networks. It is focused on quality (meaning "being useful to explore real data") to allow a rigorous interpretation of the graph (e.g. in SNA) with the fewest biases possible, and a good readability even if it is slow.

Author:	Mathieu Jacomy
Date:	2007
Kind:	Force-directed
Complexity:	$O(N^2)$
Graph size:	1 to 10 000 nodes
Use edge weight:	Yes



¹D. E. Knuth, The Stanford GraphBase: A Platform for Combinatorial Computing, Addison Wesley, Reading, MA (1993).

05 Gephi

Analysis

Gephi 0.9.1 - Project 2

파일(F) Workspace 도구(T) 창(W) Plugins 도움말(H)

Overview Data Laboratory Preview

Workspace 2

Appearance

- Nodes Edges
- Unique Attribute
- #1f0300

Graph

- Dragging (Configure)
- Nodes
- Edges
- Attributes
- Groups
- Network
- Project
- File
- Help

Layout

- Choose a layout
- Run

<등록 정보 없음>

HTML Report

Parameters:

Network Interpretation: undirected

Results:

Average Clustering Coefficient: 0.680
 Total triangles: 5
 The Average Clustering Coefficient is the mean value of individual coefficients.

Clustering Coefficient Distribution

Algorithm:

Matthieu Latapy, *Main-memory Triangle Computations for Very Large (Sparse (Power-Law)) Graphs*, in Theoretical Computer Science (TCS) 407 (1-3), pages 458-473, 2008

Context

- Nodes: 7
- Edges: 10
- Undirected Graph

Filters Statistics

Settings

Network Overview

- Average Degree Run
- Avg. Weighted Degree Run
- Network Diameter Run
- Graph Density Run
- Modularity Run
- PageRank Run
- Connected Components Run

Node Overview

- Avg. Clustering Coefficient 0.68 Run
- Eigenvector Centrality Run

Edge Overview

- Avg. Path Length Run

Dynamic

- # Nodes Run
- # Edges Run
- Degree Run
- Clustering Coefficient Run

05 Mission

1. 설문조사.gml 파일을 열고 pyplot 그래프로 나타내 보세요. **설문조사 결과는 Directed Graph입니다! nx.DiGraph()
2. Eigenvector Centrality, Hits Algorithm으로 네트워크의 Hub를 찾아내 보세요. 두 분석의 결과가 같나요?
3. 네트워크를 비방향 그래프로 바꿔 보세요.(nx.Graph()로 바꾸면 됩니다.) Eigenvector Centrality가 제일 높은 노드의 Communicability를 도출해 보세요. 결과로부터 어떤 정보를 얻을 수 있나요?
4. Gephi를 통해 시각화하고, Force Atlas Layout을 적용시켜 보세요.

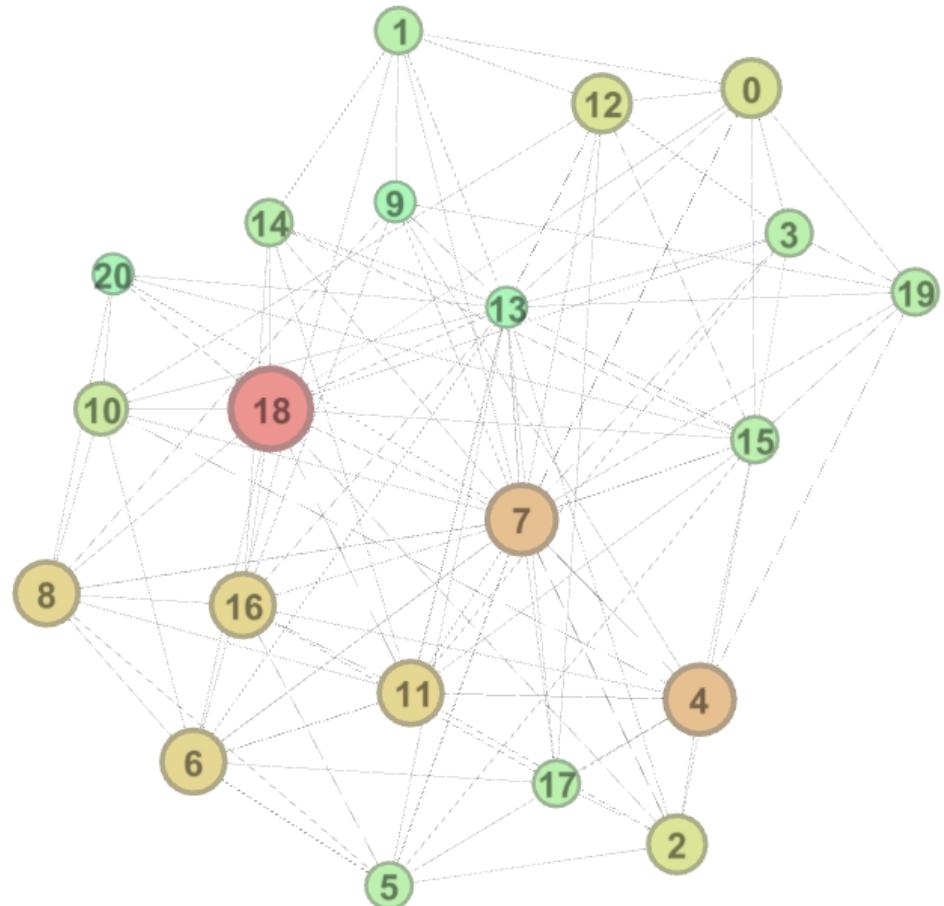
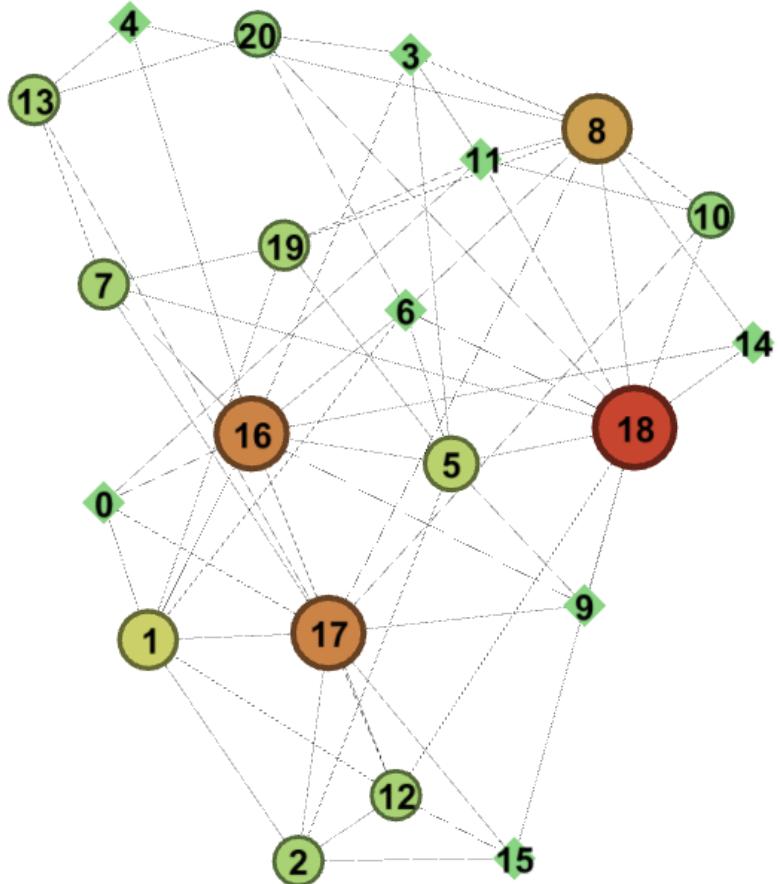
05 Answer

1. 생략
2. [('18', 0.4862), ('17', 0.4189), ('16', 0.3074)]
[('6', 0.0973), ('9', 0.0877), ('12', 0.0753)]
[('18', 0.1853), ('16', 0.1590), ('17', 0.1322)]
3. [('18', 80.9672), ('8', 67.8125), ('16', 67.3215)]

05 Answer

1. 생략
2. [('18', 0.3536), ('7', 0.3100), ('4', 0.2908)]
[('7', 0.1671), ('13', 0.1558), ('15', 0.0942)]
[('18', 0.0704), ('11', 0.0595), ('2', 0.0592)]
3. [('7', 4323.2773), ('13', 3939.5768), ('18', 3150.8676)]

05 Answer



Reference

- <https://www.slideshare.net/koorukuroo/20140830-pycon2014-networkx>
- <http://networkx.readthedocs.io/en/latest/>
- <http://www.kateto.net/wp-content/uploads/2012/12/COMM645%20-%20Gephi%20Handout.pdf>
#Gephi Handout and reference

THANK YOU !