그래프를 이용한 기계 학습

#3 검색 엔진에서는 그래프를 어떻게 활용할까?

신기정
(KAIST AI대학원)

1. 페이지랭크의 배경

2. 페이지랭크의 정의

3. 페이지랭크의 계산

4. 실습: 나무 위키 검색 엔진

1. 페이지랭크의 배경

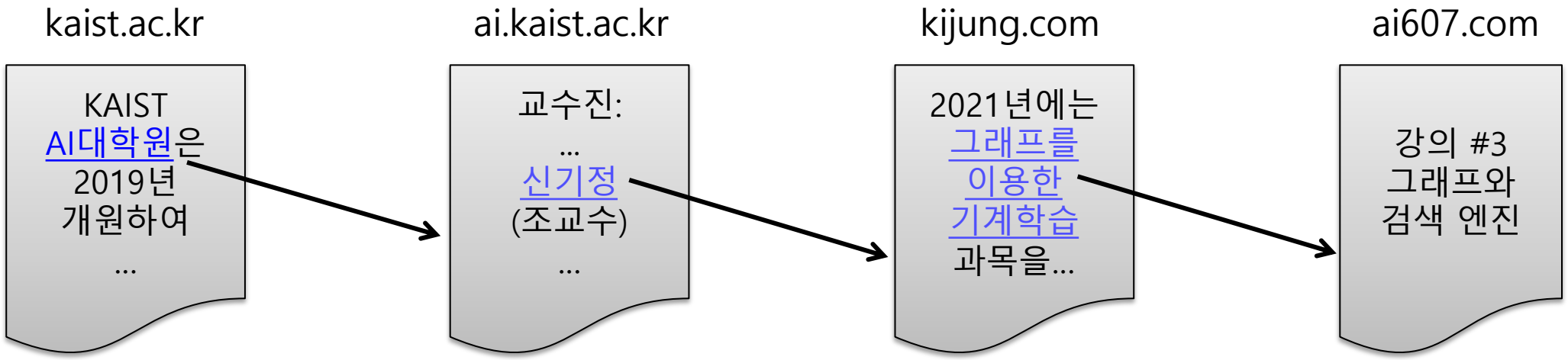
1.1 웹과 그래프

1.2 구글 이전의 검색 엔진

1.1 웹과 그래프

웹은 웹페이지와 하이퍼링크로 구성된 거대한 방향성 있는 그래프입니다

웹페이지는 정점에 해당합니다
웹페이지가 포함하는 하이퍼링크는 해당 웹페이지에서 나가는 간선에 해당합니다
단, 웹페이지는 추가적으로 키워드 정보를 포함하고 있습니다



1.2 구글 이전의 검색 엔진

첫번째 시도는 웹을 거대한 디렉토리로 정리하는 것이었습니다

웹페이지의 수가 증가함에 따라서 카테고리의 수와 깊이도 무한정 커지는 문제가 있습니다
참고로 현재는 수십억 ~ 수백억 개의 웹페이지가 있는 것으로 알려져 있습니다
또한, 카테고리 구분이 모호한 경우가 많아, 저장과 검색에 어려움이 있습니다

1.2 구글 이전의 검색 엔진

두번째 시도는 웹페이지에 포함된 키워드에 의존한 검색 엔진입니다

사용자가 입력한 키워드에 대해, 해당 키워드를 (여러 번) 포함한 웹페이지를 반환합니다

하지만, 이 방법은 악의적인 웹페이지에 취약하다는 단점이 있습니다
예를 들어, 성인 사이트에 '축구'라는 키워드를 (보이지 않도록) 여러 번 포함하게 되면,
'축구'를 검색했을 때 해당 성인 사이트가 결과로 나올 수 있습니다

1.2 구글 이전의 검색 엔진

Q. 사용자 키워드와 관련성이 높고 신뢰할 수 있는 웹페이지를 어떻게 찾을 수 있을까요?

A. 구글의 창업자인 래리 페이지(Larry Page)와 세르게이 브린(Sergey Brin)은 *The PageRank Citation Ranking: Bringing Order to the Web* 라는 제목의 논문을 통해 이 질문에 답합니다

당시 둘은 스탠포드 대학의 박사과정 학생이었습니다

래리 페이지(Larry Page)와 세르게이 브린(Sergey Brin)

1. 페이지랭크의 정의

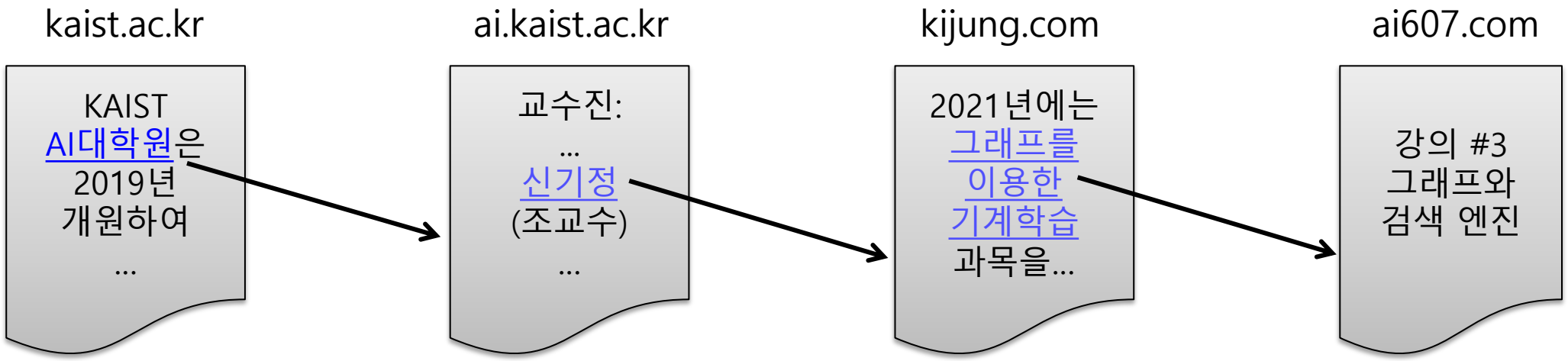
1.1 페이지랭크의 정의: 투표 관점

1.2 페이지랭크의 정의: 임의 보행 관점

2.1 페이지랭크의 정의: 투표 관점

페이지랭크의 핵심 아이디어는 투표입니다
즉, 투표를 통해 사용자 키워드와 관련성이 높고 신뢰할 수 있는 웹페이지를 찾습니다

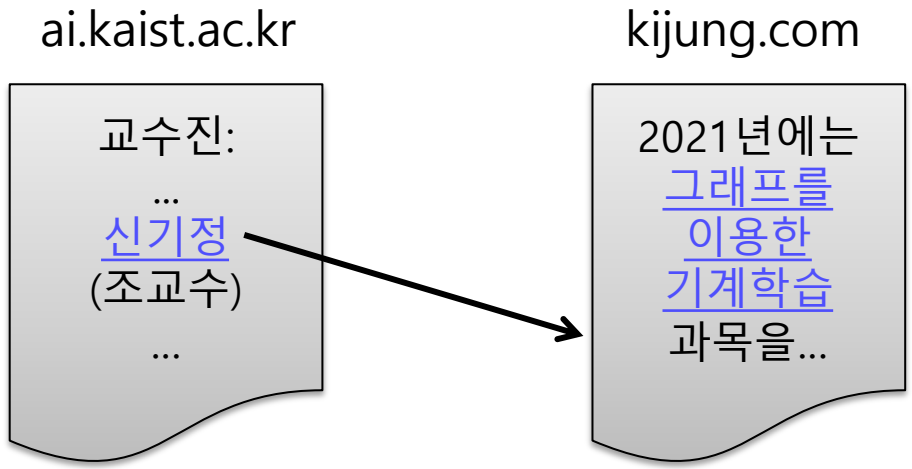
투표의 주체는 바로 웹페이지입니다
웹페이지는 하이퍼링크를 통해 투표를 합니다



2.1 페이지랭크의 정의: 투표 관점

페이지랭크의 핵심 아이디어는 투표입니다
즉, 투표를 통해 사용자 키워드와 관련성이 높고 신뢰할 수 있는 웹페이지를 찾습니다

투표의 주체는 바로 웹페이지입니다
웹페이지는 하이퍼링크를 통해 투표를 합니다



사용자 키워드를 포함한 웹페이지들을 고려합시다
웹페이지 u가 v로의 하이퍼링크를 포함한다면?
u의 작성자가 판단하기에 v가 관련성이 높고 신뢰할 수 있다는 것을 의미합니다
즉, u가 v에게 투표했다고 할 수 생각할 수 있습니다

2.1 페이지랭크의 정의: 투표 관점

즉, 들어오는 간선이 많을 수록 신뢰할 수 있다는 뜻입니다

논문을 고를 때도 마찬가지입니다
사람들은 많이 인용된 논문을 더 많이 신뢰합니다

2.1 페이지랭크의 정의: 투표 관점

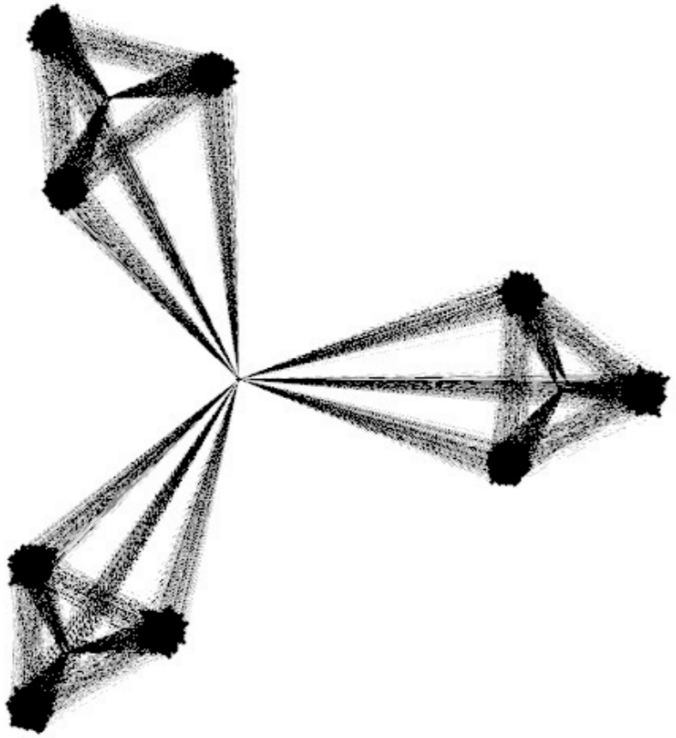
즉, 들어오는 간선이 많을 수록 신뢰할 수 있다는 뜻입니다

Q. 그런데 들어오는 간선의 수를 세는 것만으로 충분할까요?

A. 아닙니다. 악용될 소지가 있습니다

웹페이지를 여러 개 만들어서 간선의 수를 부풀릴 수 있습니다
즉, 관련성과 신뢰도가 높아 보이도록 조작할 수 있습니다

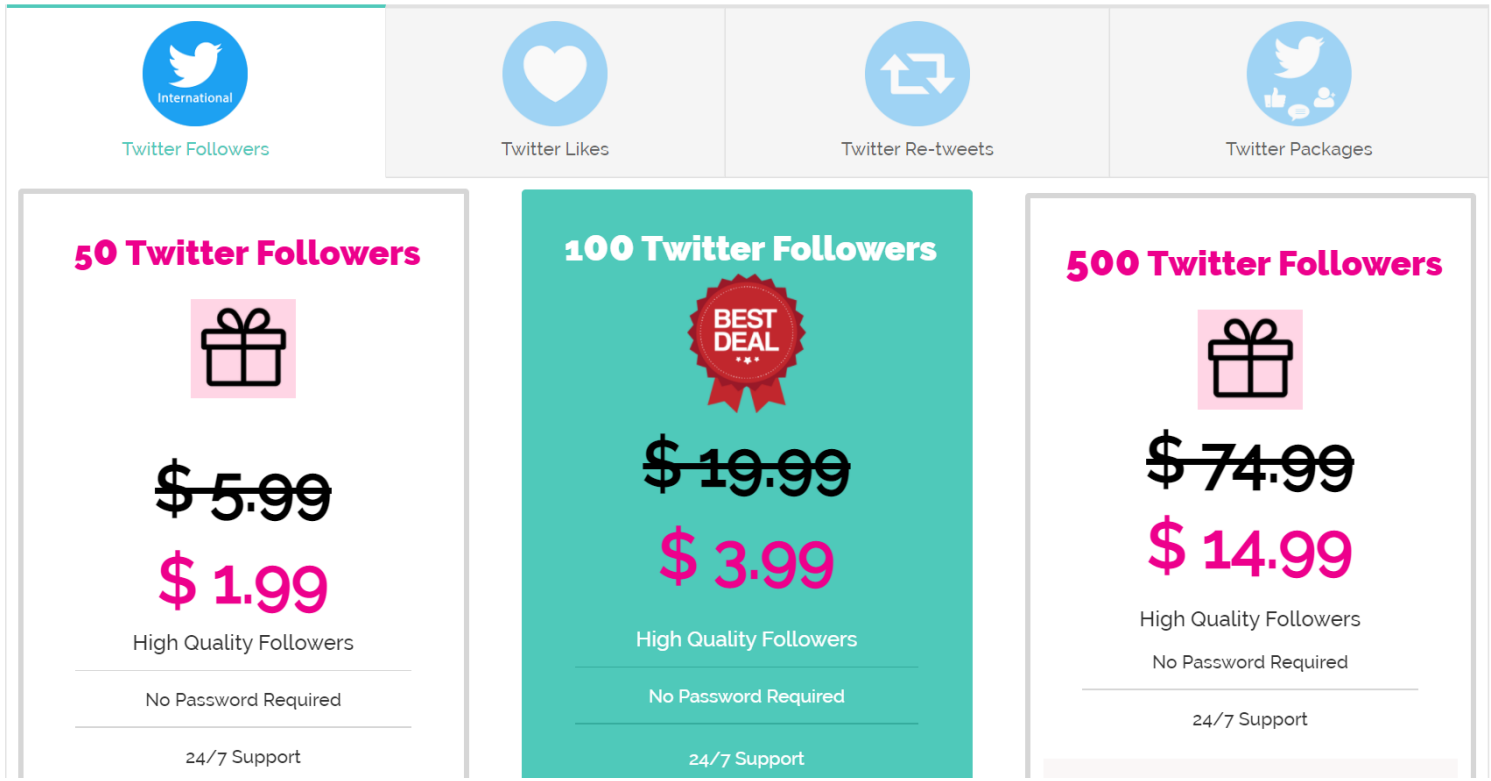
오른쪽 그림은 제가 연구를 통해 찾아낸 웹 그래프의 일부입니다
완벽한 대칭성 등, 인위적으로 만들어진 것으로 의심됩니다



이미지 출처: http://www.cs.cmu.edu/~kijungs/papers/kcoreKAIS2018.pdf

2.1 페이지랭크의 정의: 투표 관점

이런 식의 악용은 온라인 소셜 네트워크에서도 흔히 발견됩니다



이미지 출처: http://www.cs.cmu.edu/~kijungs/papers/kcoreKAIS2018.pdf

2.1 페이지랭크의 정의: 투표 관점

Q. 이런 악용을 막으려면 어떻게 해야 할까요?

A. 이런 악용에 의한 효과를 줄이기 위해, 페이지랭크에서는 가중 투표를 합니다
즉, 관련성이 높고 신뢰할 수 있는 웹사이트의 투표를 더 중요하게 간주합니다
반면, 그렇지 않은 웹사이트들의 투표는 덜 중요하게 간주합니다
악용이 없는 경우에도 사용할 수 있는 합리적인 투표 방법입니다

2.1 페이지랭크의 정의: 투표 관점

Q. 이런 악용을 막으려면 어떻게 해야 할까요?

A. 이런 악용에 의한 효과를 줄이기 위해, 페이지랭크에서는 가중 투표를 합니다
즉, 관련성이 높고 신뢰할 수 있는 웹사이트의 투표를 더 중요하게 간주합니다
반면, 그렇지 않은 웹사이트들의 투표는 덜 중요하게 간주합니다
악용이 없는 경우에도 사용할 수 있는 합리적인 투표 방법입니다

Q. 잠깐, 관련성과 신뢰성은 저희가 투표를 통해 측정하려는 것 아니었나요?
출력을 입력으로 사용하자는 이야기처럼 들리는데요?

A. 그렇습니다. 재귀(Recursion), 즉 연립방정식 풀이를 통해 가능합니다.

2.1 페이지랭크의 정의: 투표 관점

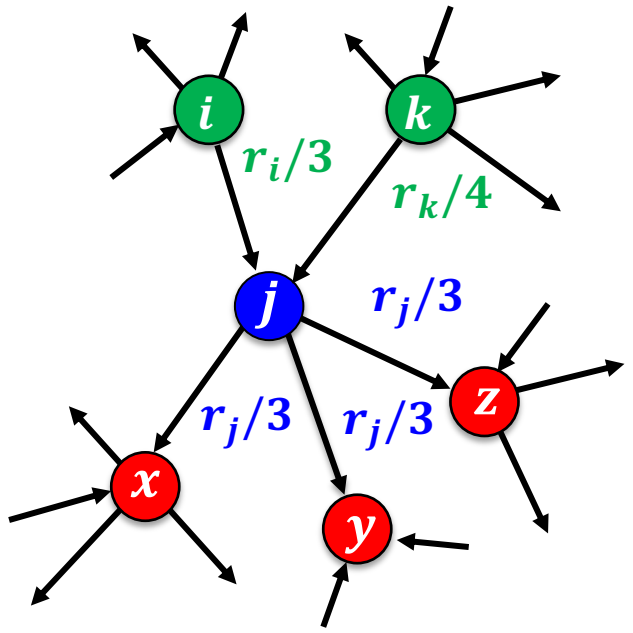
측정하려는 웹페이지의 관련성 및 신뢰도를 페이지랭크 점수라고 부릅시다

각 웹페이지는 각각의 나가는 이웃에게

$\frac{\text{자신의 페이지랭크 점수}}{\text{나가는 이웃의 수}}$ 만큼의 가중치로 투표를 합니다

오른쪽 예시에서 웹페이지 j는 웹페이지 x, y, z에게 각각 가중치 $\frac{r_j}{3}$ 으로 투표를 합니다

r_j는 웹페이지 j의 페이지랭크 점수를 의미합니다



이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch5.pdf

2.1 페이지랭크의 정의: 투표 관점

측정하려는 웹페이지의 관련성 및 신뢰도를 페이지랭크 점수라고 부릅시다

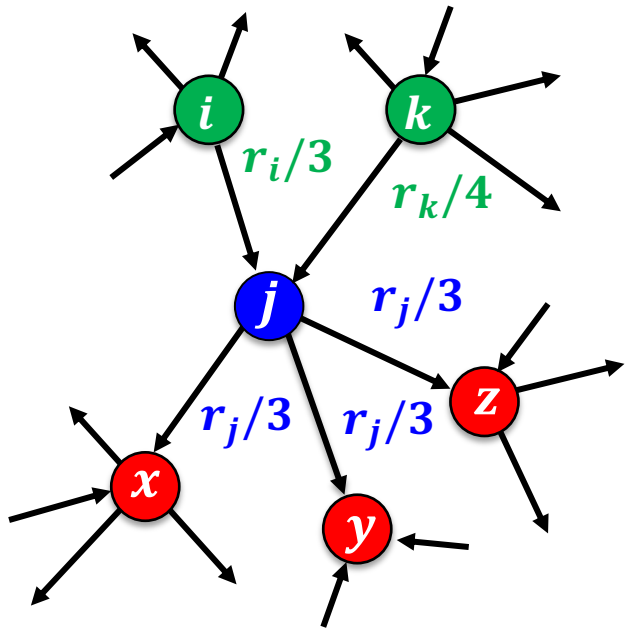
각 웹페이지의 페이지랭크 점수는 받은 투표의 가중치 합으로 정의됩니다

오른쪽 예시에서 웹페이지 j의 페이지랭크 점수 r_j는 다음과 같습니다

$$r_j = r_i / 3 + r_k / 4$$

페이지랭크 점수의 정의는 다음과 같습니다

$$r_j = \sum_{i \in N_{in}(j)} \frac{r_i}{d_{out}(i)}$$



이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch5.pdf

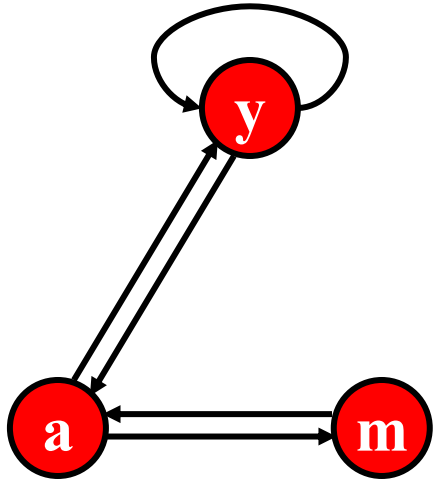
2.1 페이지랭크의 정의: 투표 관점

측정하려는 웹페이지의 관련성 및 신뢰도를 페이지랭크 점수라고 부릅시다

오른쪽 예시에서의 정점 별 페이지랭크 식은 다음과 같습니다

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

변수 3개 식이 3개이므로 연립방정식을 통해 풀 수 있습니다
구체적인 계산은 뒤에서 자세히 설명합니다



2.2 페이지랭크의 정의: 임의 보행 관점

페이지랭크는 임의 보행(Random Walk)의 관점에서도 정의할 수 있습니다

임의 보행을 통해 웹을 서핑하는 웹서퍼를 가정합시다
즉, 웹서퍼는 현재 웹페이지에 있는 하이퍼링크 중 하나를
균일한 확률로 클릭하는 방식으로 웹을 서핑합니다

웹서퍼가 t번째 방문한 웹페이지가 웹페이지 i일 확률을 $\boldsymbol{p_i(t)}$라고 합시다
그러면 $\boldsymbol{p(t)}$는 길이가 웹페이지 수와 같은 확률분포 벡터가 됩니다

그러면 아래 식이 성립합니다

$$\boldsymbol{p_j(t+1) = \sum_{i \in N_{in}(j)} \frac{p_i(t)}{d_{out}(i)}}$$

2.2 페이지랭크의 정의: 임의 보행 관점

페이지랭크는 임의 보행(Random Walk)의 관점에서도 정의할 수 있습니다

웹서퍼가 이 과정을 무한히 반복하고 나면, 즉 t가 무한히 커지면,
확률 분포는 $\boldsymbol{p(t)}$는 수렴하게 됩니다

다시 말해 $\boldsymbol{p(t) = p(t+1) = p}$이 성립하게 됩니다
수렴한 확률 분포 $\boldsymbol{p}$는 정상 분포(Stationary Distribution)이라고 부릅니다
그러면 앞서 소개한 수식을 아래와 같이 바꿀 수 있습니다

$$\boldsymbol{p_j(t+1) = \sum_{i \in N_{in}(j)} \frac{p_i(t)}{d_{out}(i)}} \quad \Rightarrow \quad \boldsymbol{p_j = \sum_{i \in N_{in}(j)} \frac{p_i}{d_{out}(i)}}$$

잠깐, 이 수식 좀 익숙하지 않나요?

2.2 페이지랭크의 정의: 임의 보행 관점

투표 관점에서 정의한 페이지 랭크 점수는 임의 보행 관점에서의 정상 분포와 동일합니다

투표 관점에서 정의한 페이지랭크 점수 $\boldsymbol{r}$

$$r_j = \sum_{i \in N_{in}(j)} \frac{r_i}{d_{out}(i)}$$

임의 보행 관점에서 정의한 정상 분포 $\boldsymbol{p}$

$$p_j = \sum_{i \in N_{in}(j)} \frac{p_i}{d_{out}(i)}$$

3. 페이지랭크의 계산

3.1 페이지랭크의 계산: 반복곱

3.2 문제점과 해결 방법

3.1 페이지랭크의 계산: 반복곱

페이지랭크 점수의 계산에는 반복곱(Power Iteration)을 사용합니다

반복곱은 다음 세 단계로 구성됩니다

(1) 각 웹페이지 i의 페이지랭크 점수 $\boldsymbol{r_i^{(0)}}$ 를 동일하게 $\frac{1}{\text{웹페이지의 수}}$ 로 초기화합니다

(2) 아래 식을 이용하여 각 웹페이지의 페이지랭크 점수를 갱신합니다

$$\boldsymbol{r_j^{(t+1)}} = \sum_{i \in N_{in}(j)} \frac{\boldsymbol{r_i^{(t)}}}{\boldsymbol{d_{out}(i)}}$$

(3) 페이지랭크 점수가 수렴하였으면 종료합니다. 아닌 경우 (2)로 돌아갑니다

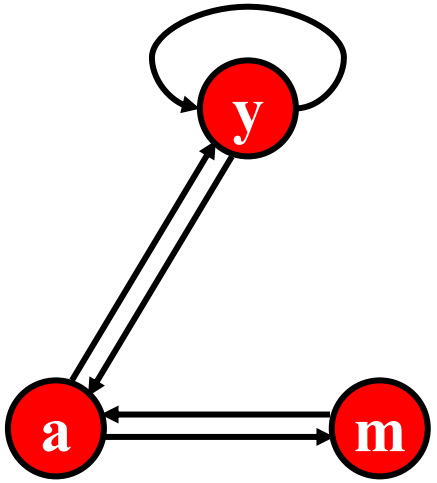
3.1 페이지랭크의 계산: 반복곱

페이지랭크 점수의 계산에는 반복곱(Power Iteration)을 사용합니다

반복곱 예시:

$$\begin{matrix} r_y \\ r_a \\ r_m \end{matrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

반복: 0



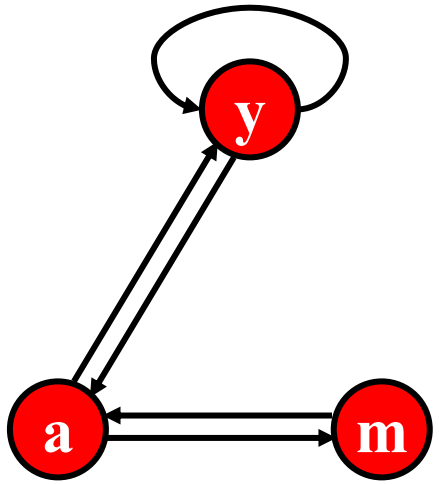
$$r_j = \sum_{i \in N_{in}(j)} \frac{r_i}{d_{out}(i)}$$

3.1 페이지랭크의 계산: 반복곱

페이지랭크 점수의 계산에는 반복곱(Power Iteration)을 사용합니다

반복곱 예시:

r_y	1/3	1/3	5/12	9/24		6/15
r_a =	1/3	3/6	1/3	11/24	…	6/15
r_m	1/3	1/6	3/12	1/6		3/15
반복:	0	1	2	3		



$$r_j = \sum_{i \in N_{in}(j)} \frac{r_i}{d_{out}(i)}$$

3.2 문제점과 해결책

앞선 예시에서는 반복곱이 잘 동작하는 것을 알겠습니다 그런데…

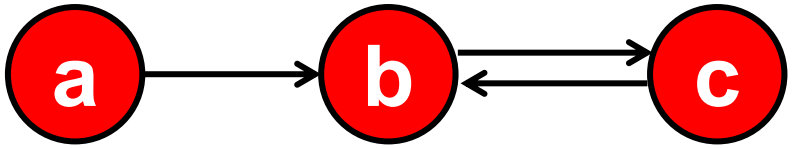
Q1. 반복곱이 항상 수렴하는 것을 보장할 수 있나요?

Q2. 반복곱이 "합리적인" 점수로 수렴하는 것을 보장할 수 있나요?

3.2 문제점과 해결책

Q1. 반복곱이 항상 수렴하는 것을 보장할 수 있나요?

정답은 '아니오' 입니다



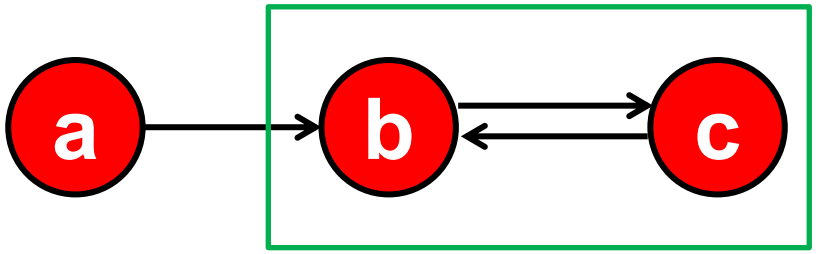
오른쪽 예시에서 페이지랭크 점수는 수렴하지 않습니다

r_a		1/3	0	0	0
r_b	=	1/3	2/3	1/3	2/3	…
r_c		1/3	1/3	2/3	1/3
반복:		0	1	2	3	

3.2 문제점과 해결책

Q1. 반복곱이 항상 수렴하는 것을 보장할 수 있나요?

정답은 '아니오' 입니다



스파이더 트랩

오른쪽 예시에서 페이지랭크 점수는 수렴하지 않습니다

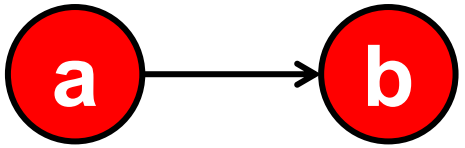
r_a	1/3	0	0	0	
r_b =	1/3	2/3	1/3	2/3	…
r_c	1/3	1/3	2/3	1/3	
반복:	0	1	2	3	

들어오는 간선은 있지만 나가는 간선은 없는 정점 집합인
스파이더 트랩(Spider Trap)에 의한 문제입니다

3.2 문제점과 해결책

Q2. 반복곱이 "합리적인" 점수로 수렴하는 것을 보장할 수 있나요?

정답은 '아니오' 입니다



오른쪽 예시에서 페이지랭크 점수는 0으로 수렴합니다

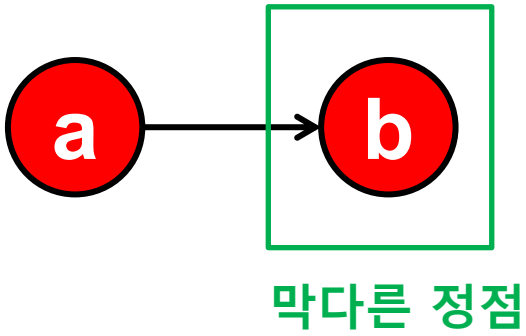
r_a		1/2	0	0	0	
r_b	=	1/2	1/2	0	0	…
반복:		0	1	2	3	

들어오는 간선은 있지만 나가는 간선은 없는
막다른 정점(Dead End)에 의한 문제입니다

3.2 문제점과 해결책

Q2. 반복곱이 "합리적인" 점수로 수렴하는 것을 보장할 수 있나요?

정답은 '아니오' 입니다



오른쪽 예시에서 페이지랭크 점수는 0으로 수렴합니다

r_a		1/2	0	0	0	
r_b	=	1/2	1/2	0	0	…
반복:		0	1	2	3	

들어오는 간선은 있지만 나가는 간선은 없는
막다른 정점(Dead End)에 의한 문제입니다

3.2 문제점과 해결책

문제 해결을 위해 순간이동(Teleport)을 도입합니다

임의 보행 관점에서, 웹을 서핑하는 웹서퍼의 행동을 다음과 같이 수정합니다
(1) 현재 웹페이지에 하이퍼링크가 없다면, 임의의 웹페이지로 순간이동 합니다
(2) 현재 웹페이지에 하이퍼링크가 있다면, 앞면이 나올 확률이 α인 동전을 던집니다
(3) 앞면이라면, 하이퍼링크 중 하나를 균일한 확률로 선택해 클릭합니다
(4) 뒷면이라면, 임의의 웹페이지로 순간이동 합니다

(1)과 (4)의 임의의 웹페이지는 전체 웹페이지들 중에 하나를 균일확률로 선택합니다
순간이동에 의해서 스파이더 트랩이나 막다른 정점에 갇히는 일이 없어졌습니다
α를 감폭 비율(Damping Factor)이라고 부르며 값으로 보통 0.8 정도를 사용합니다

3.2 문제점과 해결책

순간이동 도입은 페이지랭크 점수 계산을 다음과 같이 바꿉니다

(1) 각 막다른 정점에서 (자신을 포함) 모든 다른 정점으로 가는 간선을 추가합니다

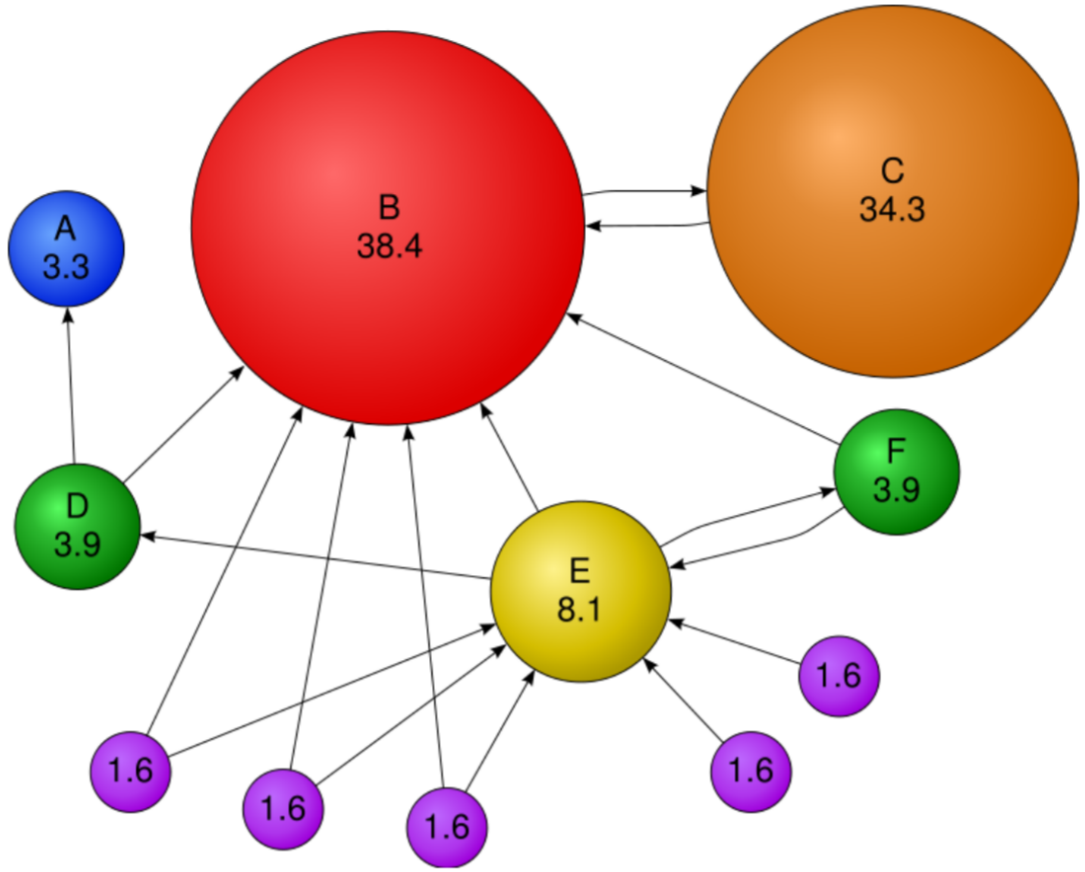
(2) 아래 수식을 사용하여 반복곱을 수행합니다

$$r_j = \sum_{i \in N_{in}(j)} \left(\alpha \frac{r_i}{d_{out}(i)} \right) + (1-\alpha) \frac{1}{|V|}$$

$|V|$는 전체 웹페이지의 수를 의미합니다
파란색 부분은 하이퍼링크를 따라 정점 j에 도착할 확률을 의미합니다
빨간색 부분은 순간이동을 통해 정점 j에 도착할 확률을 의미합니다

3.2 문제점과 해결책

수정된 페이지랭크 점수 예시



이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch5.pdf

4. (실습) 나무위키 검색 엔진

4.1 나무위키 데이터 소개

4.2 데이터 불러오기

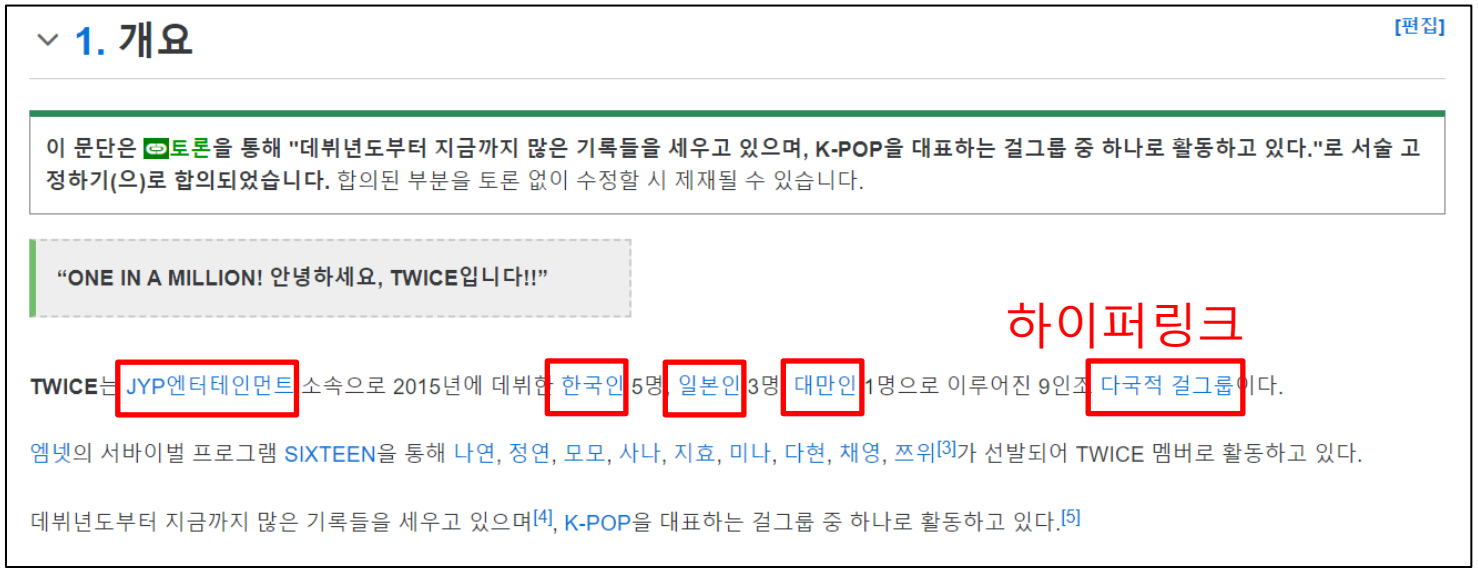
4.3 검색 1단계: 부분그래프 구성

4.4 검색 2단계: 페이지랭크 점수 측정

4.1 나무위키 데이터 소개

나무위키란?

나무위키는 서브컬쳐에 특화된 위키피디아 사이트입니다
문서들이 하이퍼링크를 통해 연결되어 있습니다



나무위키 문서 예시 (TWICE)

4.1 나무위키 데이터 소개

다음의 나무위키 데이터를 수집하여 제공합니다

가수와 관련된 내용으로 한정하였습니다

1.5만 개의 정점, 17만 개의 간선, 27만개의 키워드로 구성되어 있습니다

4.1 나무위키 데이터 소개

구체적인 파일 이름과 형식은 다음과 같습니다

vertices.txt: 문서 목록입니다.
 - 각 줄이 "[문서 식별자]₩n" 형태입니다
vertex2name.txt: 문서의 제목 목록입니다.
 - 각 줄이 "[문서 식별자] ||| [제목]₩n" 형태입니다
edge.txt: 하이퍼링크 목록입니다.
 - 각 줄이 "[나가는 문서 식별자] [들어오는 문서 식별자]₩n" 형태입니다
keyword.txt: 키워드 목록입니다.
 - 각 줄이 "[키워드 식별자] ||| [키워드] ₩n" 형태입니다
v2k.txt: 문서 별 포함된 키워드 목록입니다.
 - 각 줄이 "[문서 식별자] ||| [키워드1 식별자] [키워드2 식별자] … ₩n" 형태입니다
k2v.txt: 키워드 별 해당 키워드가 포함된 문서 목록입니다
 - 각 줄이 "[키워드 식별자] ||| [문서1 식별자] [문서2 식별자] … ₩n" 형태입니다

4.2 데이터 불러오기

필요한 라이브러리를 읽어옵니다

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import os
import os.path as osp
import sys
```

4.2 데이터 불러오기

파일에 저장된 데이터를 읽어옵니다

```
# 실습에 필요한 데이터셋을 읽어서 저장합니다.
print("###### Read Graphs ######")
path_v2n = osp.abspath(osp.join(os.getcwd(), 'drive/MyDrive/data/wiki/vertex2name.txt'))
path_edges = osp.abspath(osp.join(os.getcwd(), 'drive/MyDrive/data/wiki/edges.txt'))
path_keyword = osp.abspath(osp.join(os.getcwd(), 'drive/MyDrive/data/wiki/keyword.txt'))
path_k2v = osp.abspath(osp.join(os.getcwd(), 'drive/MyDrive/data/wiki/k2v.txt'))

G = nx.DiGraph()
f = open(path_edges)
for line in f:
    v1, v2 = map(int, line.split())
    G.add_edge(v1, v2)
```

4.2 데이터 불러오기

파일에 저장된 데이터를 읽어옵니다

```
print("###### Read keyword ######")
keywords = {}
f = open(path_keyword)
for line in f:
    num, k = line.split(" ||| ")
    k = k.rstrip()
    num = int(num)
    keywords[k] = num
```

4.2 데이터 불러오기

파일에 저장된 데이터를 읽어옵니다

```
print("###### Read keyword to vertex ######")
k2v={}
f = open(path_k2v)
for line in f:
    k, v = line.split(" ||| ")
    k = int(k)
    v = v.rstrip()
    v = v.split()
    v = list(map(int, v))
    k2v[k] = v
```

4.2 데이터 불러오기

파일에 저장된 데이터를 읽어옵니다

```
print("###### Read vertex to name ######")
v2n = {}
f = open(path_v2n)
for line in f:
    v, n = line.split(" ||| ")
    v = int(v)
    n = n.rstrip()
    v2n[v] = n
```

4.3 검색 1단계: 부분그래프 구성

주어진 검색어가 포함된 문서들로 구성된 부분그래프(Subgraph)를 구성합니다

```
# 검색어로 사용할 키워드를 입력으로 받아서, 그 키워드를 포함한
# 문서들로 이루어진 부분 그래프(subgraph) H를 추출합니다.

print("###### Mapping Subgraphs for each keyword ######")
search = "걸스데이"
print("Search : %s" % search)
key = keywords[search]
sub_vertices = k2v[key]

H = G.subgraph(sub_vertices)

print(len(H.nodes))
```

```
###### Mapping Subgraphs for each keyword ######
Search : 걸스데이
301
```

5.4 검색 2단계: 페이지랭크 점수 측정

구성된 부분그래프(Subgraph)에서 페이지랭크를 수행하여 문서 별 점수를 계산합니다
문서들을 페이지랭크 점수 역순으로 정렬하여 출력합니다

```
# subgraph H에 대해서 pagerank 알고리즘을 시행합니다.
print("###### PageRank Algorithm ######")
pr = nx.pagerank(H, alpha = 0.9)
res = [key for (key, value) in sorted(pr.items(), key=lambda x:x[1], reverse=True)]
for item in res:
    print(v2n[item])
```

```
###### PageRank Algorithm ######
걸스데이
민아(걸스데이)
유라(걸스데이)
소진(걸스데이)
달샤벳
```

3강 정리

1. 페이지랭크의 배경

- 디렉토리, 키워드 기반 검색 엔진의 한계

2. 페이지랭크의 정의

- 투표 관점: 하이퍼링크를 통한 가중 투표
- 임의 보행 관점: 웹서퍼가 각 웹페이지를 방문할 확률

3. 페이지랭크의 계산

- 반복곱
- 스파이더 트랩 및 막다른 정점 문제를 해결하기 위한 순간 이동

4. (실습) 나무위키 검색 엔진