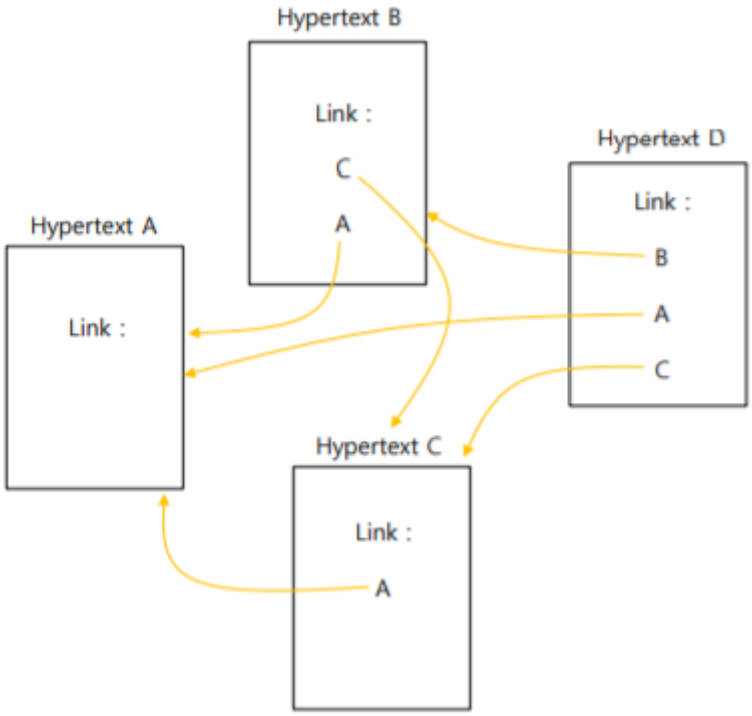
출처가 명시되지 않은 모든 자료(이미지 등)는 조성현 강사님의 강의 기반

문서 요약

1. PageRank

> *In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext.*



예시 출처 : Wikipidia

위와 같이 4개의 문서 A, B, C, D가 있다. 각 링크 연결 상태가 노란 선으로 표시되어 있다. A는 3개 페이지로부터, B는 1개 페이지로부터, C는 2개 페이지로부터, D는 0개 페이지로부터 링크가 걸려 있다. A가 제일 중요해 보인다.

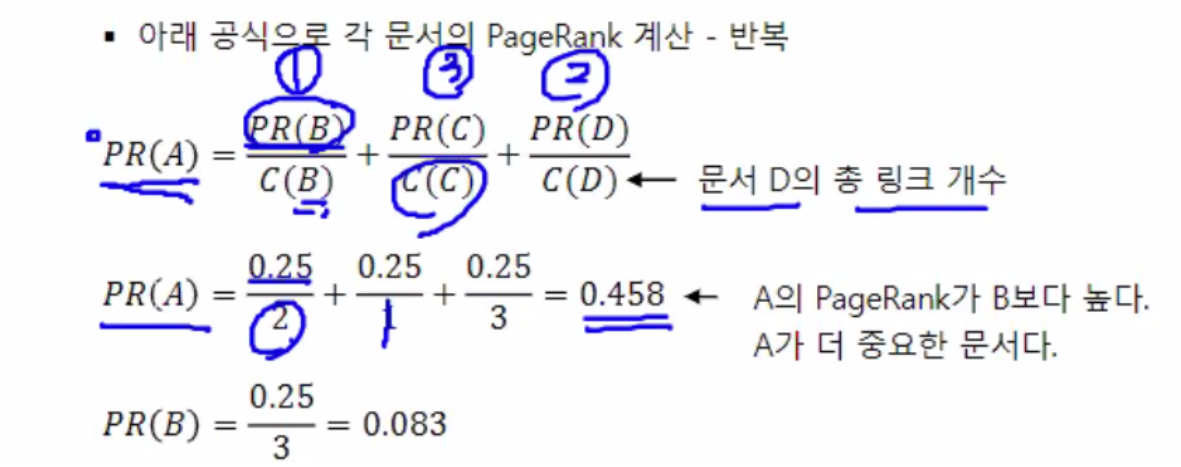
이 아이디어를 어떻게 수치화하여 나타낼 수 있을까?

1) 각 페이지의 PageRank 초기화

문서의 개수를 N이라 할 때, 각 페이지 랭크를 $\frac{1}{N}$로 초기화한다.

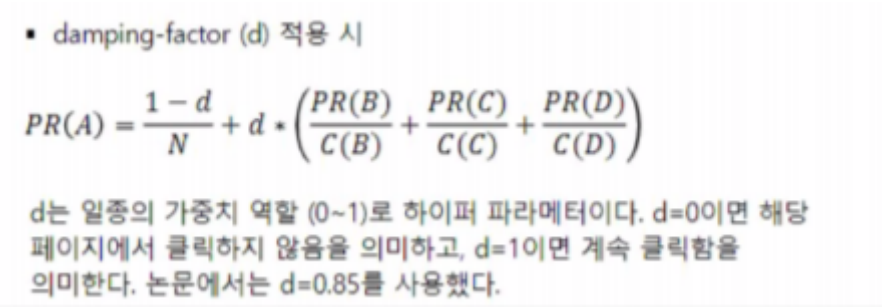
2) 각 페이지의 PageRank 계산 및 반복

각 문서마다 다음과 같은 방식으로 `PageRank`를 계산한다.



A 문서를 예로 들어 확인하자. A로 유입되는 링크가 있는 문서는 B, C, D이다. B, C, D 각각의 총 링크 개수를 분모에, B, C, D 각각의 `PageRank`를 분자에 놓고 더한 값으로 `PageRank` 값을 업데이트한다.

3) damping-factor 적용



- `d`가 0이면 해당 페이지에서 클릭하지 않음을 의미. 가중치 0으로 설정.

예컨대 링크가 걸려 있어도 링크에 관심이 없으면 클릭 안 함. 그래서 클릭 안 해서 `d`가 0이면, `1/N`만 남는다.

- d=1이면 계속해서 클릭.

이러한 클릭의 빈도 등을 조절해 주는 역할을 하는 파라미터가 `d`이다. 논문의 저자는 0.85를 적용했다.

`PageRank` 알고리즘에서는 중요한 문서로부터 얼마나 많은 링크가 걸려 있는지가 중요하다. 연결의 *질*을 따진다.

2. TextRank

Google의 PageRank 알고리즘을 차용해서 만든 문서 요약 알고리즘이다. 하나의 문서 안에 여러 문장들이 있을 때, 문장들 간에 연계, 유사도가 있을 것이다. 중요한 문장, 유사한 문장일수록 자주 등장할 것이라는 아이디어를 기반으로 한다.

1) 각 문장의 TextRank 초기화

PageRank와 동일하게, 문장 개수의 역수로 초기화한다.

2) 문장 간 유사도 측정

`i` 번째 문장과 `j` 번째 문장 간 유사도 계산 과정을 반복한다.

$$Similarity(S_i, S_j) = \frac{|\{W_k | W_k \in S_i \ \& W_k \ \in S_j\}}{log(|S_i|) + log(|S_j)}$$

- 분모: 각 문장 단어 개수의 절댓값에 로그를 취한 것을 더한 값.
- 분자: 두 문장 모두에 등장하는 단어의 개수의 절댓값을 취한 값.

3) TextRank 계산 및 반복

- 아래 공식으로 각 문장의 TextRank 계산 - 반복

weight (similarity between Si and Sj)

$$TR(S_i) = \underbrace{(1-d)}_{N} + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} TR(S_j)$$ ← Weighted graph

(if d = 1)

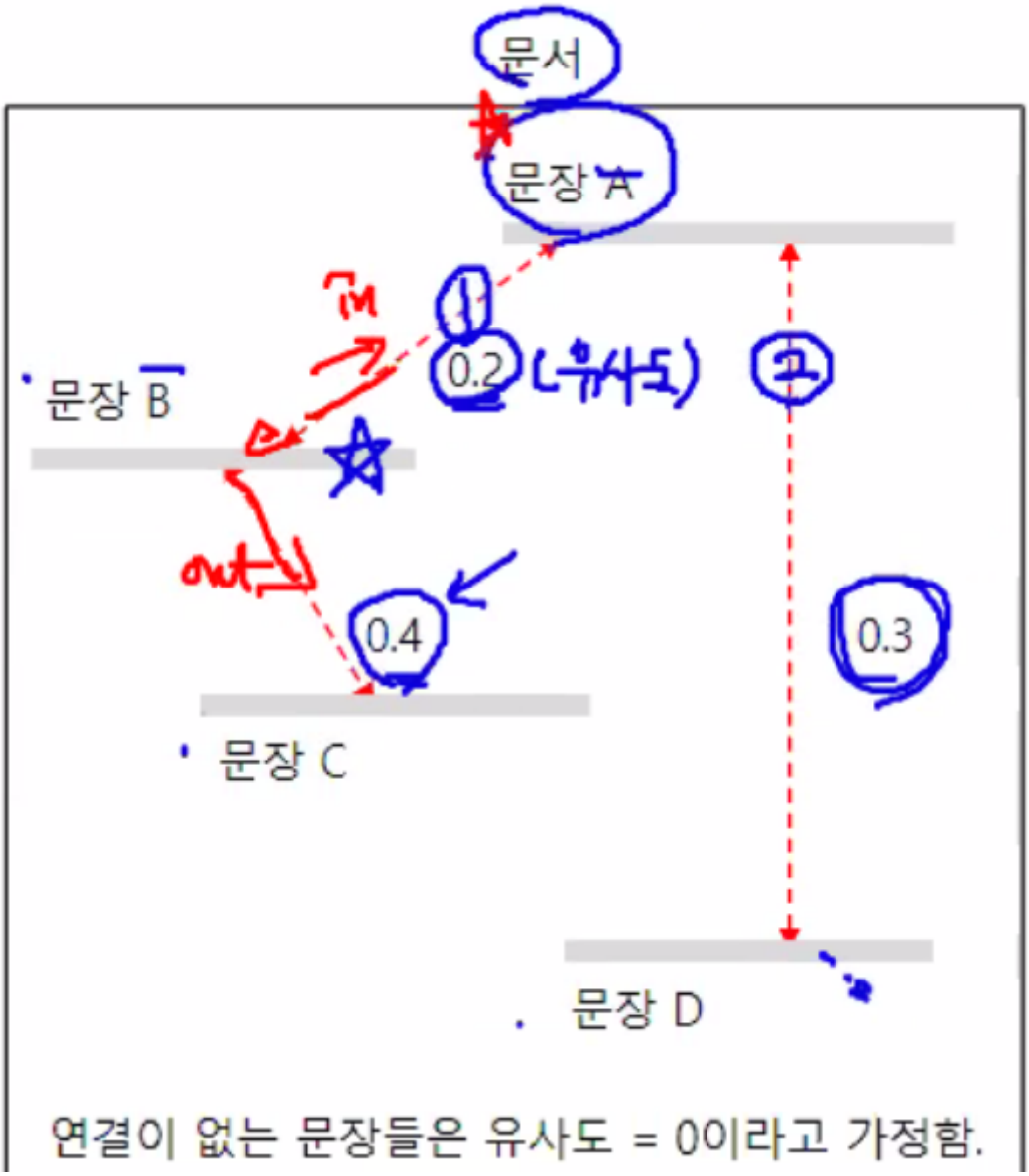
$$TR(A) = \frac{0.2}{0.2+0.4} * 0.25 + \frac{0.3}{0.3} * 0.25 = 0.33$$

$$TR(B) = \frac{0.2}{0.2+0.3} * 0.25 + \frac{0.4}{0.4} * 0.25 = 0.35$$

B의 TextRank가 A보다 높다. B가 A보다 더 중요한 문장이다. 직관적으로 타당한가 ?

The resulting graph is highly connected, with a weight associated with each edge, indicating the strength of the connections established between various sentence pairs in the text. The text is therefore represented as a weighted graph, and consequently we are using the weighted graph-based ranking formula introduced in Section 2.2.

- `in` : 관련 있는 문장.
- `out` : 관련 있는 문장과 관련이 있는*(?)* 또 다른 문장.



$TR(A)$를 계산하는 방법을 보는 것이 더 잘 이해된다.

그림 상으로 보면 문장 A는 B, D와 관련 있다. 따라서 각 문장과의 유사도를 분자에 놓는다. B 입장에서는 A와도 관련 있고, C와도 관련 있다. 따라서 A와 B, B와 C의 유사도를 더한 것을 B 부분의 분자에 놓는다.

문장 A의 경우 0.2, 0.3 문장이 주목하고, B의 경우 0.2, 0.4 문장이 주목한다. 직관적으로 B가 A보다 더 중요한 것처럼 보이는데, 실제로 그렇다.

위와 같은 방법에 의해 모든 문장에 대해 TR 지수를 계산한 후, *가장 높은* 것들을 뽑아 내면 그 문서를 대표하는 문장이라고 본다.

실습

Gensim 라이브러리에서 `summarize` 함수를 불러 와 사용한다. 설정한 `ratio` 인자에 따라 요약되는 문장의 길이가 달라진다.

```
from gensim.summarization import summarize

summary = summarize(text, ratio=0.1)
```

> *참고*
>
> 위 실습에서의 url은 가짜 논문을 만들어 주는 사이트이다.

요약 결과를 확인해 보면, 생각보다 *실망(?)*스럽다. 많이 나오고, 유사한 데이터가 많이 나온다. 결국 많은 양의 데이터를 가지고 학습해야 함과, 어떻게 보면 이것이 인공지능의 한계라는 점을 알려주는 것 같기도?