LexRank를 이용한 문서요약

허성욱

actto8290@gmail.com

https://github.com/AnalystH/

Contents

- ✓ Abstract
- ✓ Markov Chain
- ✓ PageRank
- ✓ LexRank

✓ Abstract

- **Goal**
  - 문서요약 알고리듬인 LexRank를 이용해 문서요약
  - 요약한 문서를 가지고 실제 포털사이트의 자동 요약시스템과 비교
- **LexRank**
  - Google 검색 알고리듬인 PageRank의 아이디어를 이용
- **PageRank**
  - Markov Chain을 이용해 각 Page에 Rank를 매김으로써, 특정 단어를 검색할 때 Rank 순으로 Page를 출력할 수 있게끔 만들었다.
  - 실제로 Google의 검색 알고리듬은 단순히 PageRank만 이용할 뿐만 아니라 훨씬 더 복잡하다고 한다.

✓ Markov Chain

- **Definition**

  $\{ X_n : n \in T \}$ is a Markov Chain if

  $$P(X_n = x \mid X_0, \dots, X_{n-1}) = P(X_n = x \mid X_{n-1})$$

  for all x and n, where T= {0,1,2, ...}.

- **Transition probabilities**
  - $p_{ij} = P(X_{n+1} = j \mid X_n = i)$ $(p_{ij} \geq 0, \sum_j p_{ij} = 1)$
  - p_{ij} 가 원소인 행렬 P 를 transition matrix라 한다.
- **n-step transition probabilities (n번째 후의 확률)**
  - $p_{ij}(n) = P(X_{m+n} = j \mid X_m = i)$, $(P^n)_{ij} = p_{ij}(n)$
  - Markov Chain의 정의에 의해 위 식이 성립한다. (증명 생략)

✓ Markov Chain

- **Irreducible**
  - 임의의 서로 다른 state가 왔다갔다 할 수 있다.

    (Page로 예를 들면, 네이버에서도 다음으로 갈 수 있고, 다음에서도 네이버로 갈 수 있다.)
- **Ergodic**
  - Recurrent, non-null and aperiodic 하면 그 Chain은 ergodic하다고 부른다.
  - Recurrent : 특정 state에서 출발해서 다시 돌아올 수 있다.
  - Non-null : 유한한 시간 안에 돌아온다.
  - Aperiodic : 주기성이 없다.

  → 즉, 특정 state에서 출발해서 다시 돌아오는데 유한한 시간 안에 오고 주기성이 없으면 ergodic하다.
- **'Unique' Stationary distribution($\boldsymbol{\pi}$)**
  - Irreducible하고 Ergodic한 Markov Chain은 고유한 Stationary distribution을 가진다고 알려져 있다.
  - Stationary distribution 을 π라 하면 $\pi = \pi P$ 인 고유한 π가 존재.

✓ PageRank

➢ **IDEA (전체 웹페이지 갯수를 n이라 가정)**

- Markov Chain 아이디어를 이용. 각 페이지에 걸려있는 링크 갯수를 확률화 시켜서 Transition matrix를 만든다.
- Irreducible 하고 Ergodic하게 '항상' 만들어주기 위해 damping factor(d)라는 걸 집어넣는데, 이는 어떤 페이지에서 다른 페이지로 '점프'할 수 있게 만들어 주는 것이다. 그렇게 함으로써 Stationary distribution을 만들어, 어떤 페이지에서 시작을 했는지에 관계 없이 각 페이지의 중요도를 측정한다.
- Stationary distribution(π)은 $(1\times n)$ 벡터로 만들어지는데, 각 값은 사용자가 페이지에 머무는 시간과 비례한다. 따라서 오래 머무를수록 중요한 페이지라고 볼 수 있는 것이다.
- Stationary distribution을 만들고 숫자가 큰 순서부터 Rank를 매겨 실제 검색어가 들어올 때 Rank가 큰 웹페이지부터 차례로 보여준다.

✓ PageRank

➢ Formula

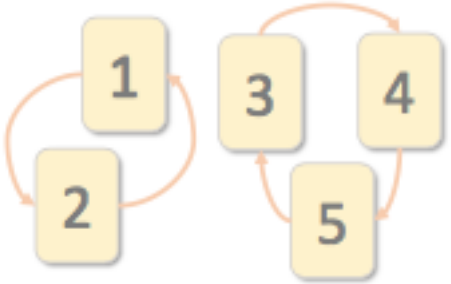
- $PR(i) = \frac{d}{n} + (1-d)\sum_{j=1}^{n} \frac{I_{ij}}{N_j} PR(j)$ (단, $PR(i) = \frac{n}{1-d}\pi_i$) , d는 보통 0.15를 사용한다고 한다.

→ 프로그램을 이용해서 계산을 해야하기 때문에, 계산하기 편하게 행렬로 바꾸면

$$P = \left(\frac{d}{n}E + (1-d)IN^{-1}\right)P \quad (\text{단}, E\text{는 모든 원소가 } 1\text{인 } n\times n \text{ 행렬}, \sum_{i=1}^{n} P_i = 1)$$

✓ PageRank

➢ Example



$$I = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, N = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathrm{P} = \left(\frac{d}{n}E + (1-d)IN^{-1}\right) = \frac{0.15}{5}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} + 0.85\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0.03 & 0.88 & 0.03 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.88 \\ 0.03 & 0.03 & 0.88 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.88 & 0.03 \end{pmatrix}$$

Stationary distribution$(\pi) = (0.2, 0.2, 0.2, 0.2, 0.2)^T$

상식적으로도, 페이지마다 링크가 각각 1개씩 걸려있으니 중요도도 같을 것

✓ LexRank

➢ IDEA

- 기본적인 아이디어는 PageRank랑 똑같다. 다만 PageRank에서는 각 웹사이트에 걸려있는 링크수를 가지고 Transition matrix를 만들었다면, LexRank에서는 문장안에 있는 단어 유사도를 가지고 Transition matrix를 만들고 문장의 중요도를 파악한다.
- 문장의 중요도를 계산한 상태에서, 실제 몇 줄로 요약할 것인지 입력을 하면 Rank가 높은 문장 순서대로 출력을 해주면 된다.

➢ **단어 유사도 측정**

- $$\text{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x}\text{tf}_{w,y}(\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x}(\text{tf}_{x_i,x}\text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(\text{tf}_{y_i,y}\text{idf}_{y_i})^2}}$$
- $tf_{w,s}$: s문장 안에 있는 w단어의 빈도

  $idf_w = \log\left(\frac{N}{n_w}\right)$, N : 총 문서의 갯수 n_w : w 단어가 들어가 있는 문서의 갯수

✓ LexRank

➢ **Transition Matrix**

- $\text{idf-modified-cosine}(x, y)$ 를 계산해서 Transition Matrix를 만드는데, 문제점이 있다. 행렬의 각 열의 합이 1이 된다는 보장이 없다는 것이다. Markov Chain을 가지고 Transition Matrix를 만들면 열의 합이 1이 돼야 한다. 따라서 각 열을 합이 1이 되게끔, 열의 합으로 나누어 주는 게 필요하다.

- 즉, $\frac{\text{idf-modified-cosine}(u, v)}{\sum_{z \in adj[v]} \text{idf-modified-cosine}(z, v)}$ 을 계산해서 Transition Matrix를 만든다. 그 다음에

  PageRank에서 damping factor(d)를 넣었던거와 마찬가지로 행렬을 만들고, 그 행렬을 통해 Stationary distribution을 만들고 문장의 중요도를 계산한다.

➢ **Formula**

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{\text{idf-modified-cosine}(u, v)}{\sum_{z \in adj[v]} \text{idf-modified-cosine}(z, v)} p(v)$$

감사합니다.