

Probabilistic approaches

보다 공식적으로 용어 가중치를 정의하려는 시도들은 확률 이론에 그 뿌리를 두고 있다. 어떤 것에 대한 확률 개념, 예를 들어 $P(R)$ 로 표기하는 적합성의 확률은 대체로 실험을 통해 공식화 되는데, 이 경우에 실험은 관찰로 만들어지는 과정이다. 실험의 모든 잠재적 결과의 집합을 sample space라 부른다. $P(R)$ 의 경우에, sample space는 $\{relevant, irrelevant\}$ 일 수 있으며, 우리는 무작위 변수 R 은 $0=irrelevant$ 이고 $1 = relevant$ 라면, $\{0,1\}$ 의 값을 취한다고 정의할 수 있다.

이제 무작위로 집단에서부터 한 개의 다큐를 취하는 실험을 정의해 보자: 만일 집단에 있는 적합한 다큐의 수를 안다면, 즉 100개의 다큐가 적합하다고 한다면, 그리고 집단에 있는 다큐의 총 수를 안다면, 즉 100만 개라면, 이것 둘의 지수는 적합성 확률을 $P(R=1) = 1,000,000 = 0.0001$ 로 정의한다. 추가로, $P(D_k)$ 가 sample space $\{0,1\}$ 와 더불어 용어 k 를 포함하고 있는 다큐의 확률이라고 가정한다면, 우리는 $P(R, D_k)$ 를 사용하여 결합확률분포(joint probability distribution)의 결과 값 $\{(0,0), (0,1), (1,0), (1,1)\}$ 을 얻을 수 있고, $P(R/D_k)$ 를 사용하면 조건확률분포(conditional probability distribution)의 결과 값 $\{(0,1)\}$ 을 얻을 수 있다. 따라서 만일 용어 k 를 포함하고 있는 다큐를 대상으로, $P(R=1/D_k=1)$ 는 적합성 확률이다.

표기법 $P(\dots)$ 는 과부하(overload) 걸렸음을 주목하라. 언젠가 우리는 또다른 무작위 변수나 sample space에 대한 얘기할 것이며, 또한 또 다른 척도인 P 에 대해서도 얘기할 것이다. 또한 D 와 T 같은 무작위 변수들이 또 다른 모델에서는 또 다른 sample space를 가질 수 있다는 것을 주목해야 한다. 예를 들어, 확률색인모델에서 D 는 무작위 변수이며, 이것은 “*this is the relevant document*”를 나타내며, 잠재적 결과로서 집단에 있는 다큐들의 identifiers를 갖는다. 그렇지만, 확률검색모델에서 D 는 무작위 변수이며 잠재적 결과로서 모든 잠재적 다큐 descriptions를 갖는데, 이 경우에는 다큐가 용어 k 에 의해 색인되었는지 아닌지를 나타내는 이진 구성요소인 d_k 로 된 벡터들이다.

또 다른 것을 생각해 보자. 변수 A 는 한 사건(잠재적 결과 공간의 부분 집합)을 나타낸다. 똑같이 우리는 결과부터 실수까지 함수인 무작위 변수를 통해 부분집합을 표현할 수도 있으며, 이러한 부분집합은 무작위 변수 A 가 특별한 값을 갖는 도메인이다. 우리는 종종 그 사건이 참인지 확실하게 알지 못한다. 우리는 이러한 사건의 확률 $0 \leq P(A) \leq 1$ 을 알아볼 수 있다. 두 개의 사건 A 와 B 가 동시에 발생할 결합(joint) 사건은 결합확률(joint probability $P(A,B)$)로 나타내며, 조건부 확률(conditional probability $P(A/B)$)은 사건 B 가 발생할 때 사건 A 가 발생할 확률을 나타낸다. 결합과 조건부 확률간의 근본적인 관계는 연쇄법칙(chain rule)에 의해 제공된다:

$$P(A,B) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

1. Binary Independence Model (BIM)

Binary Independence Assumption은 다큐들이 이진 벡터들이라는 것이다. 즉, 다큐 속에 용어의 존재여부만을 기록하는 것이다. 용어들은 적합한 다큐들의 세트에 독립적으로 분산되어 있듯이, 또한 부적합한 다큐의 세트에도 독립적으로 분산되어 있다. 이것들의 표현은 불리안 변수들의 ordered set이다. 즉, 다큐나 쿼리의 표현은 고려대상인 각 용어에 대한 불리안 요소로 된 벡터이다. 좀 더 구체적으로 말해서, 다큐는 벡터 $d = (x_1, \dots, x_m)$ 로 표현된다: 만일 용어 t 가 다큐 d 에 나타나 있으면, $x_t=1$, 그렇지 않다면, $x_t=0$ 으로 표현된다.

많은 다큐들이 이렇게 간단하게 동일한 벡터로 표현될 수 있다. 쿼리도 유사한 방법으로 표현된다. “독립”이란 도큐에 있는 용어들이 서로 독립적인 것으로 여겨져서 이들 용어들 간에는 어떠한 조합도 모델화되지 않는다는 의미이다. 이 가정은 매우 제한적이지만, 많은 상황에서 매우 훌륭한 결과를 제공하는 것으로 여겨지고 있다. 이러한 독립은 서로 의미하는 성질들이 단순화를 위해 독립적인 것으로 취급되는 Naive Bayes classifier의 “naive” 가정과 같다. 이 가정은 다른 용어에서 사용된 차원과 직교하는 차원으로 각 용어를 0이나 1의 값으로 갖는 것으로 고려함으로써, 이러한 표현을 벡터 스페이스 모델의 instance처럼 취급하도록 한다.

<<베이즈 정리(Bayes' theorem) 예제>>

- 경주마 백두산은 통산 100번의 경주를 뛰었다. 그 중 20번의 경주에서 우승했다.

$$=P(\text{백두산}=\text{Win}) = 20/100 = .2$$

- ▪ 그 중 30번은 비가 왔고 나머지는 맑았다.

$$=우천확률: P(\text{Weather}=\text{Rain}) = 30/100 = .3$$

- ▪ ▪ 그 중 백두산은 15번을 이겼다.

$$=조건 확률: P(\text{백두산}=\text{Win}|\text{Weather}=\text{Rain}) = 15/30 = .5$$

$$P(\text{Win}|\text{Rain})= P(\text{Win,Rain})/P(\text{Rain}) = 0.15/0.3 = .5$$

- 그러면 “이겼을 때, 비가 올 확률 $P(\text{Rain}|\text{Win})$ 은 ?”

답: Bayes' theorem: 베이즈 정리를 사용하면;

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

$$P(R|W) = \frac{P(W|R) \times P(R)}{P(W)} = \frac{0.5 \times 0.3}{0.2} = 0.75$$

다큐가 적합하다는 확률 $P(R|d,q)$ 는 그 다큐의 용어 벡터의 적합성 확률 $P(R|x,q)$ 로부터 유래된다. Bayes rule을 사용하면, 다음과 같다:

$$P(R|x,q) = \frac{P(x|R,q)*P(R|q)}{P(x|q)}$$

조건: $P(x|R=1,q)$ 와 $P(x|R=0,q)$ 이 적합하거나 부적합한 다큐의 검색 확률이라면.
만일 그렇다면, 그 다큐의 표현은 x 이다. 정확한 확률은 미리 알 수 없으므로 다큐의 집단에 대한 통계로부터 평가치를 사용해야만 한다.

$P(R=1|q)$ 과 $P(R=0|q)$ 은 쿼리 q 에 대한 각각의 적합 또는 부적합 다큐의 이전 확률을 나타낸다. 예를 들어, 만일 우리가 컬렉션에 있는 적합한 다큐의 %를 안다면, 우리는 이러한 확률을 측정하기 위하여 그것을 사용할 수 있다. 다큐가 쿼리에 적합하거나 부적합하므로, 우리는 다음과 같은 공식을 작성할 수 있다:

$$P(R=1|x,q)+P(R=0|x,q)=1$$

▪ Query Terms Weighting

다큐와 쿼리 간의 유사도 함수로서 binary query와 dot product(내적)가 주어질 때, 검색효과를 높이기 위해, 쿼리에 있는 용어들에 가중치를 부여하는 문제가 발생한다.
 p_i 와 q_i 는 적합한 다큐와 부적합한 다큐가 각각 i^{th} 번째 용어를 가질 확률이라고 하자. 다큐의 집단에 대한 통계로부터 평가치를 사용해야만 한다.다큐의 집단에 대한 통계로부터 평가치를 사용해야만 한다. BIM을 처음 제안한 Yu와 Salton은 i^{th} 번째 용어의 가중치는

$$Y_i = \frac{p_i * (1 - q_i)}{(1 - p_i) * q_i} \text{인 증가 함수(increasing function) 이다.}$$

그러므로 만일 Y_i 가 Y_j 보다 높다면, 용어 i 의 가중치는 용어 j 의 가중치보다 높을 것이다. Yu와 Salton은 쿼리에 대한 가중치 할당이 쿼리에 동일한 가중치를 주는 것보다 더 나은 검색 효과를 얻을 수 있다고 하였다. Robertson and Spärck Jones는 그 후에 만일 i 번째 용어에 $\log Y_i$ 의 가중치를 할당한다면, BIA에서 최적의 검색효과를 얻을 수 있다고 주장하였다.

Binary Independence Model was introduced by Yu and Salton. The name Binary Independence Model was coined by Robertson and Spärck Jones.

▪ PRP(Probability Ranking Principle)

만일 각각의 리퀘스트에 대한 참고 검색모델의 응답이 리퀘스트한 이용자에게 시스템에서 사용할 수 있는 데이터를 근거로 확률들을 가능한 한 정확하게 평가한 다음, 최적의 적합성의 확률이 감소하는 순서로 집단의 다큐를 서열화하는 것이라면, 이것은 이용자가 그 같은 데이

터를 근거로 시스템으로부터 얻을 수 있는 최상의 결과일 수 있다.

2. probability indexing model

1960년에 Bill Maron과 Larry Kuhns에 의해 확률색인모델이 정의되었다. Luhn과 달리, 이들은 정보검색시스템을 위한 자동색인을 목표로 삼지 않았다. 아직까지도 수작업 색인이 이 분야에서 지배적이었으므로, 이들은 다큐 D에 적용할 수 있는 다양한 색인어 T를 사용하는(run through) 인간 색인자가 각 용어의 yes/no 결정을 하는 대신에 다큐에 주어진 용어에 확률 $P(T|D)$ 를 할당할 것을 제안하였다. 따라서 모든 다큐는 $P(T|D)$ 에 의해 가중치가 부여된 잠재적 색인어의 집합으로 끝나는데(end up with), 이때 $P(T|D)$ 는 만일 사용자가 다큐 D에 포함된 정보를 원한다면, 그가 T를 사용하여 쿼리를 공식화할 수 있는 확률이다.

Bayes'의 규칙 즉,

$$P(D|T) = \frac{P(T|D)P(D)}{P(T)} \quad (\text{공식 6})$$

을 사용하여, 그들은 $P(D|T)$ 로 다큐의 서열을 제시하였다. 즉, 이용자가 용어 T를 사용하여 쿼리를 작성한다면, 다큐 D는 적합하다. 주목할 것은 오른쪽에 있는 분모(denominator)인 $P(T)$ 는 어떤 주어진 쿼리어에 대해 상수(constant)이며, 결론적으로 다큐는 $P(D|T)$ 의 값에 정비례하는 quantity인 $P(T|D)P(D)$ 에 의해 서열화될 것이다. 이 공식에서, $P(D)$ 는 다큐 D의 연역적(a-priori) 적합성 확률이다.

$P(T|D)$ 가 인간 색인자에 의해 정의되지만, Maron & Kuhns는 다음과 같이 제안하였다: $P(D)$ 는 다큐 이용 통계(statistics on document usage)에 의해 즉, 다큐 이용의 총 수에 의한 다큐 D의 이용 수의 몫(quotient)에 의해 정의될 수 있다. 따라서 their usage of the document prior $P(D)$ 는 서열화에 대한 인기를 끌어올린 첫 번째 것으로 여겨질 수 있으며, 인터넷 탐색에서 중요하게 되었다. 흥미롭게도, $P(T|D)$ 의 평가는 다큐 각각의 사용용으로 처음에 다큐 검색을 위해 입력한 쿼리어를 저장하는 것과 비슷한 방식으로 얻을 수 있다. Maron & Kuhns는 다음과 같이 주장하였다: “그 같은 절차는 물론 극히 비현실적인 것이 될 것이다.” 그러나 사실상 이러한 기법 - 소위 click-through rates를 사용하는 rank optimization - 은 오늘날 웹 탐색엔진에서는 일반화되었다.

3. probabilistic retrieval model

Maron & Kuhns가 적합성의 확률로 서열화하는 것을 소개한 반면에, Stephen Robertson은 이 아이디어를 하나의 법칙(principle)으로 변경시켰다. 그는 probability

ranking principle을 공식화 하였다. 그는 William Cooper에게 다음과 같이 (attributed) 공을 돌렸다:

“각 리퀘스트에 대한 참조 검색 시스템의 response가, 리퀘스트를 한 이용자의 usefulness의 확률을 감소시키는 방식으로 집단에 있는 다큐들을 서열화시키는 것이라면, 그리고 이 같은 목적을 위해 시스템에서 사용하려는 어떠한 데이터라도 그것의 확률 평가는 가능한 한 정확성에 근거해야 한다면, 이용자에 대한 시스템의 전체적인 효과는 그러한 데이터를 기본으로 수 집될 때 최상일 것이다.”

이러한 제안은 다소 사소한 요구조건처럼 보인다. 왜냐하면 정보검색시스템의 목적이 “필요한 정보를 찾도록 이용자를 도와주는 것”이기 때문이다. 그러나 이러한 의미는 Luhn의 similarity principle과 매우 다를 수 있다.

이용자가 한 개의 용어를 포함하고 있는 쿼리를 입력한다고 생각해 보자: 예를 들어, 용어 social. 이용자의 요구를 만족시킬 수 있는 모든 다큐를 알 수 있다면, 아래의 도 5와 같은 Venn diagram에서 보듯이 4 non-overlapping document sets로 그 집단을 나눌 수 있을 것이다. 이 도표는 각각의 비-중복 집합의 크기에 대한 추가적 정보를 포함하고 있다.

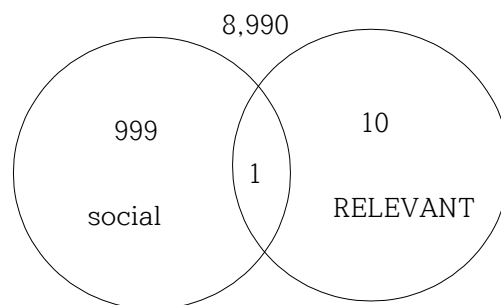


Figure 5: Venn diagram of the collection given the query term **social**

질문대상 집단에는 10,000개의 다큐가 있고 그것들 중에서 1,000개는 단어 “social”을 포함하고 있다고 가정해 보자. 추가로, 쿼리에 적합한 다큐는 11개인데, 그것들 중의 한 개는 단어 “social”을 포함하고 있다고 가정해 보자. 어떤 다큐가 “social”을 색인어로 사용한 다큐 집합으로부터 무작위로 선택된다면, 적합한 다큐가 선택될 확률은 $1/1,000 = 0.0010$ 이다. 그리고 어떤 다큐가 “social”을 색인어로 사용하지 않는 다큐 집합으로부터 무작위로 선택된다면, 적합성 확률은 $10/9,000 = 0.0011$ 로 더 커진다. 이러한 증거를 근거로, 만일 시스템에서 쿼리어 “social”이 색인에 없는 다큐를 반환한다면, 다시 말해서 가장 먼저 그 쿼리와 다른(dissimilar) 다큐를 제공한다면, 최상의 성능을 그 시스템은 발휘할 수 있다. 분명하게 말해서 이러한 전략은 Luhn의 similarity criterion을 위반하고 있다.

Stephen Robertson & Karen Spärck-Jones는 이러한 문제를 해결하고자 자신들의 확률검색모델에서 다큐의 콘텐츠 descriptions가 주어졌을 때 적합성 R의 확률인 $P(R|D)$ 에 의해 다큐를 서열화할 것을 제안하였다. 주목할 것은 D는 여기서 이진 구성요소의 벡터이며, 각

구성요소는 한 개의 용어를 표현하는 반면에, 앞 장에서 D 는 “적합한 다큐”였다는 것이다. 확률검색모델에서, 확률 $P(R|D)$ 는 다음과 같이 해석되어야 한다: 똑같은 D 로 표현할 수 있는 여러개의 다큐, 예를 들어 10개가 있을 수 있다. 만일 이것들 중 9개가 적합하다면, $P(R|D) = 0.9$ 이다. 실제로 이런 일을 하기 위하여, 우리는 확률 오즈(odds) $P(R|D)/P(\bar{R}|D)$ 에 Bayes' rule을 사용할 것이다(\bar{R} 은 부적합 다큐를 나타낸다). 오즈는 계산할 때 우리로 하여금 아직까지 적합성 확률로 서열화를 제공하고 있는 $P(D)$ 을 무시하도록 한다. 따라서 우리는 적합성에 제공된 용어들 간의 독립성을 가정해 보자.

$$\frac{P(R|D)}{P(\bar{R}|D)} = \frac{P(D|R)P(R)}{P(D|\bar{R})P(\bar{R})} = \frac{\prod_k P(D_k|R)P(R)}{\prod_k P(D_k|\bar{R})P(\bar{R})} \quad (\text{공식 7})$$

여기서, D_k 는 다큐 벡터에 있는 k 번째의 구성요소(용어)를 나타낸다. 용어들의 확률들은 위에 있는 적합한 다큐의 예제 즉, 도 5에서 “social”의 적합성 확률은 $1/11$ 이라는 방식으로 정의된다.

좀 더 편리하게 확률검색을 실행하기 위하여, three order preserving transformations 가 사용된다:

- 1) 다큐는 오즈 그 자체적인 것보다는 logarithmic odds의 합계에 의해 서열화 시킨다.
- 2) 적합성 $P(R)/P(\bar{R})$ 의 priori odds는 무시한다.
- 3) 모든 다큐 점수로부터 $\sum_k \log(P(D_k = 1|R)/P(D_k = 1|\bar{R}))$ 즉, 빈(empty) 다큐의 점수를 뺀다.

이런 방식으로 수백만개의 용어일 수 있는 모든 용어의 합계에는 그 다큐에서 제시하고 있는 용어들의 단지 non-zero 값만을 포함시킨다.

$$matching-score(D) = \sum_{k \in matching\ terms} \log \frac{P(D_k = 1|R)P(D_k = 0|\bar{R})}{P(D_k = 1|\bar{R})P(D_k = 0|R)} \quad (\text{공식 8})$$

실제로, 쿼리에 없는 용어들은 공식 8에서도 무시된다. 확률검색모델을 충분히 이용하기 위해서는 두 가지가 필요하다: 예를 들어, 적합한 다큐들과 긴(long) 쿼리들. 적합한 다큐는 $P(D_k|R)$ 즉, 적합한 용어 k 를 포함하고 있는 다큐의 확률을 계산하는데 필요하다. 이 모델에서는 다큐에 용어의 존재와 부재만을 구별하기 때문에, 그리고 결과적으로 다큐 점수의 뚜렷한(distinct) 값들의 수(number)가 짧은 쿼리에서는 낮기 때문에, 긴 쿼리가 필요하다. 예를 들어, 한 단어의 쿼리에서 뚜렷한 확률의 수는 2(다큐가 그 단어를 포함하고 있느냐 아니냐)이며, 두 단어 쿼리에서는 4(다큐가 두 단어 모두를 포함하는 경우, 첫 번째 단어만 포함하는 경우, 두 번째 단어만 포함하는 경우, 그리고 둘 다 없는 경우) 이다. 따라서 3단어의 쿼리의 수는 8 이다. 분명하게 말해서 웹 탐색용으로, 사전에 전혀 적절하지 않은 것으로 알려진 다큐용으로, 그리고 전형적으로 짧은 쿼리용으로, 이 모델은 맞지 않는다. 그렇지만 이 모델은 예를 들어 spam filter용으로는 도움을 준다. spam filters와 관련해서는 시간이 지나면서 적

합한(no spam or 'ham') 또는 부적합한(spam) 다큐에 대한 많은 예가 누적되고 있다. 입수되는 이메일이 스팸인지 아닌지를 결정하기 위하여, 이메일의 full text가 단지 몇 개의 용어 대신에 사용될 수 있다.

4. 2-Poisson model

Bookstein과 Swanson은 다큐의 색인어를 구별할 목적으로 일련의 통계 규칙을 개발하였다. 이들은 다큐에 있는 용어의 발생 빈도수인 tf 는 다음과 같은 2가지의 Poisson 분산을 혼합하여 모델화할 수 있다고 제안하였다. 이 때 X 는 빈도수의 무작위 변수이다.

$$P(X=tf) = \lambda \frac{e^{-\mu} (\mu_1)^{tf}}{tf!} + (1-\lambda) \frac{e^{-2\mu} (\mu_2)^{tf}}{tf!}$$

λ : 정해진 시간 안에 어떤 사건이 일어날 횟수에 대한 기대값

푸아송 분포(Poisson分布, 영어: Poisson distribution)는 확률론에서 단위 시간 안에 어떤 사건이 몇 번 발생할 것인지를 표현하는 이산 확률 분포이다. 다시 말해서, 푸아송 확률이란 고정된 단위 시간(또는 공간) 안에서 어떤 사건이 무작위로 몇 번 발생할 것인지를 표현하는 이산 확률 분포이다. 만일 X 가 주어진 시간에서 발생한 사건의 숫자라면, 그리고 주어진 시간당 사건의 평균수가 λ (람다)라면, 주어진 시간에 사건 x 를 관찰할 확률은 다음과 같이 구한다:

$$P(X = x) = e^{-\lambda} \frac{\lambda^x}{x!}$$

$$x = 0, 1, 2, 3, 4, \dots$$

▲ Examples

- 1) 분당 톨 게이트에 도착하는 자동차의 수.
- 2) 한 페이지에 있는 오타 typo)의 수.

▲ Note

e 는 수학적 자연상수인, $e \approx 2.718282$ 이다. e 의 파워를 계산하기 위하여 계산기에 있는 e^x 버튼을 사용한다.

만일 X 의 확률이 이러한 방식으로 분산되어 진다면, 우리는 $X \sim \text{Po}(\lambda)$ 라고 쓴다. 이때, λ 는 분산의 파라미터이다. 그리고 우리는 X 가 파라미터 λ 와 함께 푸아송 분산에 따른다고 말한다.

<예제 1>

『어떤 도서관에서 신간의 대출이 시간당 평균 1.8권씩 무작위로 이루어지고 있다고 가정해 보자. 그럼 이 도서관에서 주어진 시간에 4권의 신간이 대출될 확률은 무엇이나?』

▪ X 는 특정한 시간에 대출되는 신간의 수 이다.

1) 대출은 무작위로 발생 한다

2) 평균비율은 1.8권이다.

따라서 $X \sim \text{Po}(1.8)$ 라고 쓴다.

이제 주어진 시간에 정확하게 4권의 신간이 대출될 확률을 다음과 같은 공식을 사용하여 계산할 수 있다:

$$P(X = 4) = e^{-1.8} \frac{1.8^4}{4!} = 0.0723$$

<예제 2>

『이 도서관에서 주어진 시간에 2권 이상의 신간이 대출될 확률은 무엇이나?』

$$P(X \geq 2) = P(X = 2) + P(X = 3) + \dots$$

i.e. 계산할 확률은 무한 수이다.

다음과 같이 구한다:

$$\begin{aligned} P(X \geq 2) &= P(X = 2) + P(X = 3) + \dots \\ &= 1 - P(X < 2) \\ &= 1 - (P(X = 0) + P(X = 1)) \\ &= 1 - \left(e^{-1.8} \frac{1.8^0}{0!} + e^{-1.8} \frac{1.8^1}{1!} \right) \\ &= 1 - (0.16529 + 0.29753) \\ &= 0.537 \end{aligned}$$

5. Okapi BM25 (BM: Best Matching)

정보검색에서, Okapi BM25은 특정한 탐색쿼리에 적합성을 근거로 문서를 매칭시켜서 서열화시키는 탐색엔진에서 사용하는 ranking function 이다. 이것은 Stephen E. Robertson,

Karen Spärck Jones 등에 의해 1970년대와 80년대에 개발된 확률적 검색 프레임워크에 근거하고 있다.

BM25와 이것의 새로운 변종인 BM25F는 웹 탐색과 같은 문서검색에서 최신의 TF-IDF-like retrieval functions을 사용하고 있다.

BM25는 문서에 있는 쿼리어들 간의 상호 연관성(상대적 근접성)과 관계없이, 각 문서에 나타나는 쿼리를 근거로 문서의 세트를 서열화하는 a bag-of-words retrieval 함수(function) 이다. 이것은 단일 함수가 아니라, 비록 서로 다른 요소와 파라미터를 사용하더라도 실제로 scoring functions의 한 가족이다. 가장 잘 알려진 사례들 중의 하나는 다음과 같다:

keywords q_1, \dots, q_n 를 포함하고 있는 쿼리 Q에서, 문서 D의 BM25 score는 다음과 같이 계산한다:

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

- ▲ $f(q_i, D)$ 은 문서 D에 있는 q_i 's term frequency이다;
- ▲ $|D|$ 는 단어로 된 문서 D의 길이 이다;
- ▲ avgdl은 문서에서 발췌한 텍스트 집단에 있는 평균문서길이 이다.
- ▲ k_1 and b 는 free parameters이며, $k_1 \in [1.2, 2.0]$ 그리고 $b = 0.75$ 처럼, 대체적으로 advanced optimization이 없을 때 선택된다.
- ▲ $\text{IDF}(q_i)$ 는 쿼리용어 q_i 의IDF (inverse document frequency) weight 이다. 이것은 대체로 다음과 같이 계산한다:

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

- ▲ N 은 집단에 있는 문서의 총 수이다.
- ▲ $n(q_i)$ 은 q_i 을 포함하고 있는 문서의 수이다.

6. Bayesian network models

- 첨부자료: 04_basianNetworkModel.

7. language models

▲ 개념

언어 모델링(Language Modeling)은 자연어 안에서 문법, 구문, 단어 등에 대한 어떤 규칙성을 찾아내고 그 규칙성을 이용하기 위한 노력이다. 이런 방법을 통해 얻어진 언어모델(LM: Language Model)은 오랫동안 음성 인식이나 기계 번역, 문자 인식, 철자 교정 등 다양한 응용분야에서 시스템의 정확도를 높이고 수행 시간을 줄이는 데 유용한 방법으로 각광을 받아왔다.

언어모델은 크게 지식 기반 모델(Knowledge-based Model)과 통계적 모델(Statistical Model)로 나눌 수 있다.

1) 지식 기반 모델은 정규 문법(RG: Regular Grammar)나 문맥 자유 문법(CFG: Context-Free Grammar)을 만들고, 이러한 문법 구조에 어긋난 구조를 탐색 공간에서 제거함으로써 탐색 범위를 줄이고 인식률을 높이는 방식이다. 그러나 지식 기반 모델은 문법 구조를 만들기가 까다롭고 대용량의 어휘를 수용하기 어려울 뿐만 아니라 언어의 비문법성에 의해 규칙정의가 어렵기 때문에 범용 언어 모델이나 새로운 영역에 대한 언어 모델을 구성할 때 많은 시간과 노력을 요구하게 된다. 따라서 이 모델링 방법은 주로 특정적이고 협소한 분야의 자연언어처리 분야에서 일부 사용되고 있을 뿐 대규모의 데이터를 처리해야 하는 분야에서는 적용되기 어려운 접근법이다.

2) 통계적 모델은 대량의 말뭉치(Corpus)에서 언어 규칙을 확률로 나타내고 확률값을 통해서 탐색 영역을 제한하는 방법이다. 그래서 통계적 언어모델은 음성인식에서 정확성뿐만 아니라 탐색 공간을 급격히 줄이는 효과를 보여준다. 통계적 언어모델의 목적은 주어진 인식영역에 맞는 단어열 s 의 확률을 예측하는 것으로 단어열 s 는 w_1, w_2, \dots, w_i 로 이루어진 단어열이라고 가정하면 단어열 s 의 확률 $P(s)$ 는 다음과 같다:

$$p(w_1, w_2, \dots, w_i) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2) \dots p(w_i|w_1 \dots w_{i-1})$$

그러나 일반적인 통계적 언어모델링은 어떤 단어열에서 i 번째 단어 w_i 가 나타날 확률을 구하기 위해 w_1 에서 w_{i-1} 까지 $i-1$ 개의 단어열이 나타날 확률을 구해야 하는 번거로움이 있다. 그래서 단어의 확률은 이전 단어에 의존적이라는 Markov 가정에 의해 i 번째 단어가 나타날 확률을 구하기 위해 이전의 $N-1$ 개의 단어의 확률만을 적용하는 N-Gram 모델이 일반적이다. 특정한 문장 s 에 대한 N-gram 모델의 식은 다음과 같다:

$$P(s) \cong P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n p(w_i | w_{i-m+1}, \dots, w_{i-1})$$

w_i 는 문장에서 i 번째 단어이고 m 은 연관되어 사용된 단어 수를 정의한다. bigram 모델($m=2$)은 앞의 단어에 이어 뒤의 단어가 나타날 확률을 사용해서 측정한다. 반면 unigram 모델($m=1$)은 오직 개별 단어의 확률로 측정한다. 음성인식과 기계번역과 같은 응용분야에서는 단어순서가 중요하기 때문에 일반적으로 대용량의 학습데이터가 가능하다면 trigram($m=3$) 모델이, 소량의 학습데이터가 가능하다면 bigram이 사용되어 왔다. 정보검색에서 단어순서의 역할은 덜 명백하다고 생각되었고 이에 따라 가장 친숙하고 기본적인 언어모델인 unigram 모델이 광범위하게 사용되고 있다.

통계적 언어모델이 모두 그렇듯이 N-Gram 통계언어 모델링은 모든 가능한 문장의 확률적 분포를

추정하기 위해 언어 모델을 사용하기 때문에, 우선적으로 전체 컬렉션을 대표할 만한 샘플 데이터를 선정하고 그 데이터를 학습함으로써 언어모델을 추정하는 것이 선행되어야 한다. 따라서 통계적 언어모델의 좋은 성능을 위해서는 대용량의 학습 데이터가 필요하고, 만약 빈약한 학습데이터에 의해 빈도수가 '0'인 단어가 질의어에 포함될 경우 전체 질의어의 확률분포 값이 '0'이 되는 심각한 오류가 발생하게 된다. 즉 미등록어(OOV: Out-Of-Vocabulary) 발생시에는 그 대처가 불가능하다는 것이 통계적 언어모델링의 가장 큰 단점이다. 하지만 지난 30여 년 동안 언어모델링의 다양한 확장 기법에 대한 연구들을 통해 미등록어 발생의 단점들을 보완함으로써 음성인식, 기계 번역, 자동 교정 등과 같은 대부분의 언어모델링 응용분야에서 성공적인 결과를 보이고 있다. 이중에 통계적 언어모델인 N-gram이 가장 성공적이고 간편한 언어모델링 방법으로 알려져 있다. 이제 언어모델이 정보검색 분야에서는 어떻게 적용되는지에 대하여 간략하게 살펴보고자 한다.

▲ 언어모델과 정보검색

1998년의 Ponte와 Croft의 연구 이후, 언어모델링 연구 분야에서 정보검색은 음성인식, 기계번역과 같은 주요한 응용분야의 하나가 되었고 정보검색 연구 분야에서 언어모델은 주목해야할 검색모델의 하나로 자리 잡고 있다. 언어모델 적용은 여러 가지 이유에서 이점을 지닌다. 예를 들면, 언어모델을 사용하여 정보검색시스템을 구축하는 것은 시스템 설계와 실험 성과에 대한 이유를 확률적인 방법을 사용하여 명확한 수치 데이터로 제시해 줄 수 있다는 것이다. 또 다른 큰 이점은 지난 30여 년 동안 자연언어처리 분야에서 실험되어 축적되어온 다양한 언어모델과 완화기법의 결합에 대한 연구결과물들을 응용할 수 있다는 것이다.

정보검색에서 언어모델 적용의 기본적인 가정은 이용자가 생각하고 있는 하나의 '이상적인' 문헌이 존재하고, 이용자는 그 문헌을 요구하기 위해 질의어 텍스트로 표현한다는 것이다. 이와 같은 이용자의 질의생성과정을 모델링 하는 것이 전통적인 LMIR의 기본 개념이 된다. 즉, 문헌으로부터 질의를 생성한다는 것이다. 다시 말해 LMIR은 컬렉션 내의 각각의 문헌 강를 위한 언어모델 M_d 를 구축한다는 것이고, 그리고 어떻게 이런 각각의 문헌 모델로부터 질의 Q 가 생성될 수 있는가에 따라 문헌들을 순위화하는 것이다. 이것은 $P(Q|M_d)$ 로 표현된다.

정보검색을 위한 언어모델들은 $P(Q|M_d)$ 을 어떻게 정의하고 계산하느냐에 따라 그 접근방법이 달라진다.

Ponte와 Croft는 질의를 단일어로 취급하고 $P(Q|M_d)$ 의 근사치를 구하기 위해 질의어를 생성해낼 확률과 질의어에 포함되지 않은 단어를 생성하지 않을 확률의 곱으로 계산했다. 질의에서 같은 단어의 중복 출현은 고려되지 않았고 이를 위해 제시된 식은 다음과 같다.

$$P(Q|M_d) = \prod_{w \in Q} P(w|M_d) \prod_{w \notin Q} (1.0 - P(w|M_d))$$

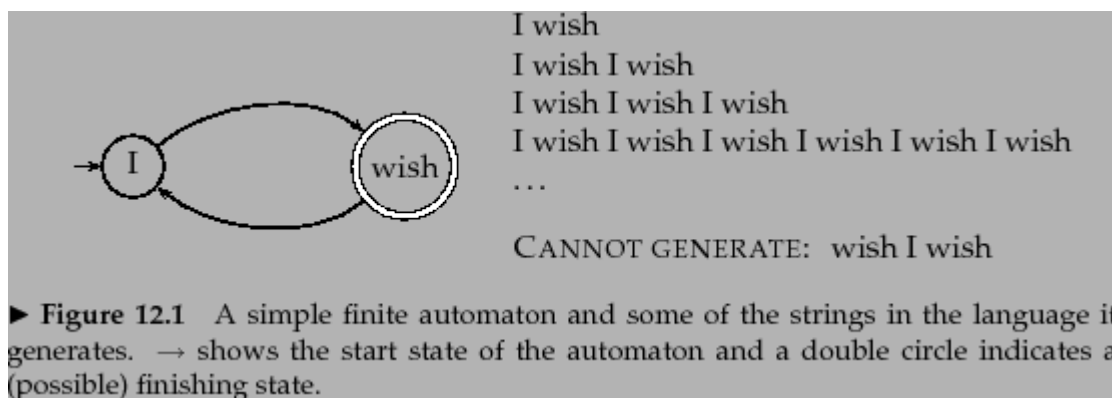
여기에서 $P(w|M_d)$ 는 비모수적 방법에 의해 계산된 것으로, w 와 위험요인을 포함한 문헌에서 w 의 평균 확률을 사용하였고, 전혀 출현하지 않은 단어를 위해서는 컬렉션에서 w 의 전체 확률을 대신 사용하였다. Ponte와 Croft의 모델은 정보검색분야에 적용된 초기 언어모델로 최근의 많은 연구들에서 기본적인 또는 전통적인 언어모델로 간주되고 있다.

전통적인 LMIR은 확률적인 계산방법에 의한 전형적인 수확모델을 따르기 때문에 비교적 단순하고 이해하기 쉬운 체제이다. 또한 LMIR은 각 문헌의 모델링 과정이 곧 색인작성 과정이 되며 이렇게 추정

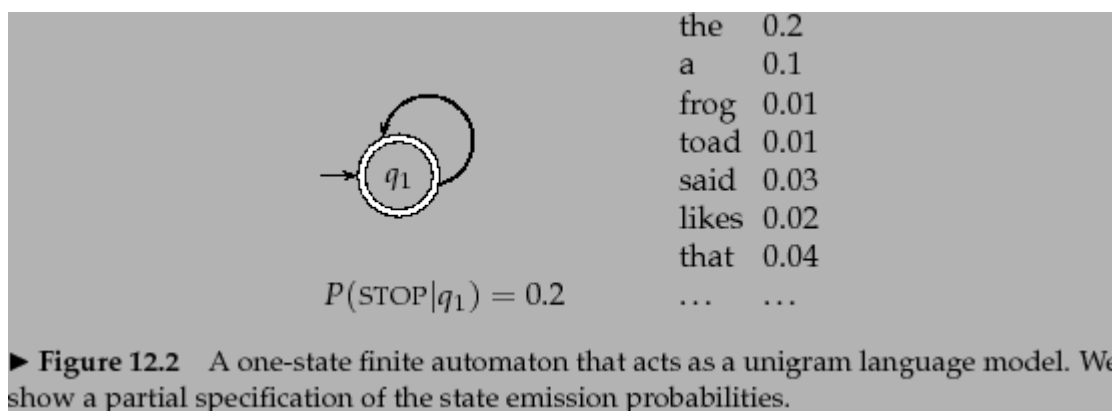
된 문헌모델에서 질의 생성확률을 추정하는 것으로 색인작성과 검색모델이 통합된 접근법이라는 장점을 가진다. 하지만 LMIR이 대량의 훈련 데이터(Training Data)가 필요하다는 점, 적합성 피드백(Relevance Feedback)이나 질의 확장(Query Expansion)을 직접적으로 적용하기 어렵다는 점, 또한 불리언 연산자나 구, 구조적인 질의 처리가 어렵다는 단점을 지니고 있다.

▲ Sample:

<Finite automata and language models>



쿼리를 생산하는 문서모델이란 무엇일까? 공식적인 언어모델의 일종인 전통적인 언어생산 모델은 문자열(string)을 인식하거나 생산하는데 사용할 수 있다. 예를 들어, 도 12.1의 finite automaton에서는 샘플을 포함하고 있는 문자열을 생산할 수 있다. 생산가능한 문자열의 풀-세트를 automation의 *language*라 부른다.



만일 각 노드가 다른 용어를 생산하는 것과 관련해서 확률분포를 가지고 있다면, 우리는 language model을 갖고 있다. language model의 개념은 본질적으로 확률적이다. language model은 어휘(vocabulary)에서 발췌한 문자열의 확률 척도를 구하는 함수이다. 예를 들어, alphabet Σ 에 대한 language model의 공식은 다음과 같다:

$$\sum_{s \in \Sigma^*} P(s) = 1$$

한 가지 가장 간단한 언어모델은 probabilistic finite automaton과 대등하다. probabilistic finite automaton은 도 12.2에서처럼, 다른 용어를 생산과 관련해서 단일 확률 분포를 갖는 단일 노드를 가지며, 그 공식은 $\sum_{t \in V} P(t) = 1$ 이다.

각 단어가 생산된 다음에, 우리는 다른 단어를 생산하기 위하여 거기서 멈출 것인지 또는 looping할 것인지를 결정한다. 또한 이 모델에서는 최종 상태에서 멈출 확률을 필요로 한다. 이 같은 모델은 단어의 순서 별로 확률분포를 설정한다. 또한 이것은 그것의 분포에 따라 텍스트를 생산하는 모델을 제공하기도 한다.

▲ example 1

word sequence의 확률을 찾기 위하여, 각 단어를 생산한 다음에 멈출 확률이나 순환한 확률과 함께, 우리는 단지 이 모델에서 순차적으로 각 단어에 매긴 확률을 곱하면 된다.

$$\begin{aligned} P(\text{frog said that toad likes frog}) &= (0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.01) \\ &\quad \times (0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.2) \\ &\approx \underline{\underline{0.0000000000001573}} \end{aligned}$$

보다시피, 특별한 문자열/문서의 확률은 대체로 매우 작은 수이다! 여기서 우리는 두 번째 frog 단어를 생산한 다음에 멈추었다. 첫 번째 줄의 숫자는 term emission probabilities 이고, 두 번째 줄은 각 단어를 생산한 다음에 계속하거나 멈출 확률들이다. 분명한 stop 확률은 finite automation이 잘 구성된 언어모델이 되기 위하여 필요하다.

그럼에도 불구하고, 대부분의 경우에, 우리는 STOP 그리고 (1-STOP) 확률을 포함시키는 것을 생략할 것이다. 데이터 세트인 두 모델을 비교하기 위하여, 우리는 그것들의 *likelihood ratio*를 계산할 것이다. *likelihood ratio*는 한 모델에서 이루어진 데이터의 확률을 다른 모델에서 이루어진 데이터의 확률로 단지 나눈 결과이다. 왜냐하면 stop 확률이 고정되어 있다면, 문자열을 생산하는 두 개의 언어 모델들의 likelihood를 비교하여 얻는 결과인 likelihood ratio는 변하지 않을 것이기 때문이다. 따라서 문서의 순위도 변하지 않을 것이다. 그럼에도 불구하고 공식적으로 그 숫자들은 더 이상은 참된 확률이 아니고 단지 확률에 비례하는 것이다.

▲ example 2

Model M_1		Model M_2	
the	0.2	the	0.15
a	0.1	a	0.12
frog	0.01	frog	0.0002
toad	0.01	toad	0.0001
said	0.03	said	0.03
likes	0.02	likes	0.04
that	0.04	that	0.04
dog	0.005	dog	0.01
cat	0.003	cat	0.015
monkey	0.001	monkey	0.002
...

Figure 12.3: Partial specification of two unigram language models.

도 12.3처럼, 우리가 두 개의 언어모델 M_1 과 M_2 를 가지고 있다고 가정해 보자. 각각은 m_1 probability에서 보여주고 있는 것처럼, 용어들의 순서에 따라 확률 값(estimate)을 제공하고 있다. 용어들의 순서에 더 높은 확률을 제공하고 있는 언어모델은 더 쉽게 용어 순(term sequence)을 생산할 수 있다. 이제 우리는 우리의 계산에서 STOP 확률을 생략할 것이다

For the sequence shown, we get:

$$\begin{array}{l}
 \text{(J)} \quad \begin{array}{ccccccccc}
 s & \text{frog} & \text{said} & \text{that} & \text{toad} & & \text{likes} & \text{that} & \text{dog} \\
 M_1 & 0.01 & 0.03 & 0.04 & 0.01 & & 0.02 & 0.04 & 0.005 \\
 M_2 & 0.0002 & 0.03 & 0.04 & 0.0001 & & 0.04 & 0.04 & 0.01
 \end{array} \\
 P(s|M_1) = 0.0000000000000048 \\
 P(s|M_2) = 0.000000000000000384
 \end{array}$$

이제 우리는 $P(s|M_1) > P(s|M_2)$ 라는 것을 알게 된다. 우리는 확률의 생산과 관련해서 공식을 제시하고 있다. 그러나 이것은 확률적 어플리케이션에서는 일반적이더라도, 실제로는 log probabilities의 합을 가지고 작업하는 것이 아마도 최상일 것이다.

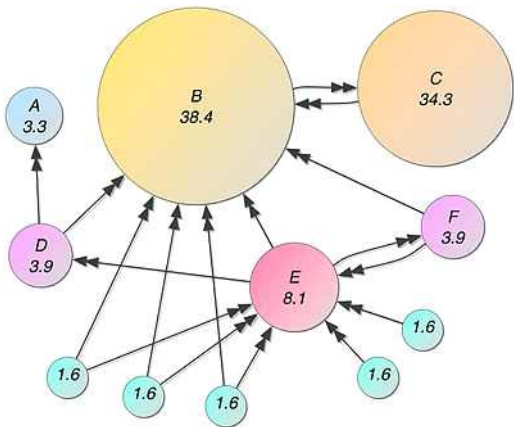
8. Google's page rank model

Sergey Brin & Lawrence Page가 웹 서치 엔진 Google을 개발했을 때, 이들은 다른 탐색엔진이 두 가지 특징으로 차별화하였다. 간단한 no-nonsense search interface와 탐색결과를 서열화하는 신속하고도 다양한 방법. 과거의 모델처럼 쿼리에 밀접하게 매치되는 다큐를 반환하는 대신에, 이들이 목표로 삼은 것은 매우 질 높은 다큐를 반환시키는 것이었다. 다시 말해서, 신뢰하는 사이트에서 생산한 다큐들. 구글은 웹의 하이퍼 링크 구조를 사용하여 PageRank라 부르는 페이지의 품질을 결정한다. 웹에서 많은 장소로부터 링크되어지는 웹 페이지는 아마도 찾아볼 가치가 있는 것이다. 이것들을 high quality 페이지임에 틀림없다. 만

일 페이지가 다른 고품질 페이지로부터 연결되어진다면, 이것은 추가로 찾아볼 가치가 있다는 것을 의미한다.

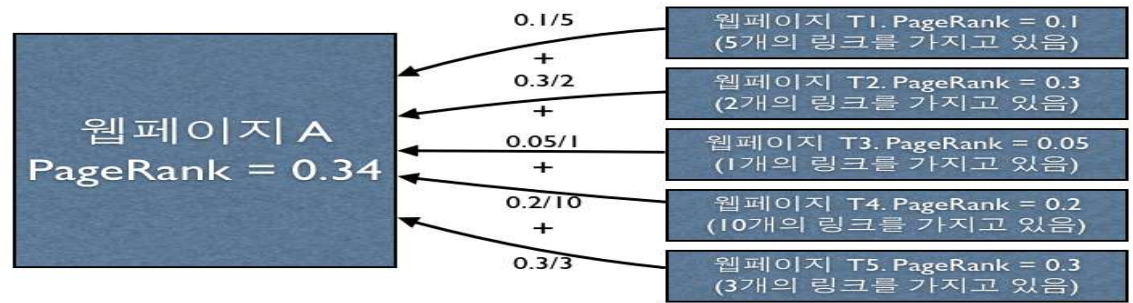
페이지 랭크는 링크분석 알고리즘이며, 하이퍼링크된 다큐 집합들의 각 요소에 대해 상대적 중요성을 측정하기 위해 숫자로 된 가중치를 할당한다. 어떤 특정한 요수 E에 할당된 가중치는 E의 페이지 랭크로 참조되며 PR(E)로 표시한다.

<예 1>



위의 간단한 네트워크의 수학적 페이지 랭크들은 %로 표현되어 있다. 페이지 C는 비록 C에 비해 링크의 수가 적더라도, 페이지 E보다 높은 페이지 랭크를 가지고 있다; C에 연결된 한 개의 링크는 높은 값을 가지고 있는 중요한 페이지에서 왔기 때문이다. 만일 아무 페이지에서 탐색을 시작한 탐색자가 현재 방문 중인 페이지로부터 아무 링크나 선택할 가능성이 85% 라면, 그리고 모든 웹에서 무작위로 선택된 페이지로 점프할 가능성이 15% 라면, 그는 그 때엔 페이지 E 8.1%에 도달할 것이다.(임의 페이지로 점프할 가능성 15%는 damping factor 85%와 같다).

<예 2>



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 또 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.

이제 d, 즉 Damping Factor에 대해 생각해 보자. 위 수식을 다시 한 번 보자. 위키피디아에 따르면, 올바른 수식은 아래와 같다:

$$PR(A) = (1-d)/N + d (PR(T1)/C(T1) + \cdots + PR(Tn)/C(Tn))$$

- 이렇게 하면 전체 페이지의 PageRank를 합산한 값이 1이 된다.

d 값은 0과 1 사이에서 정해지는데, d값이 커져서 1이 되면 앞의 (1-d)는 0이 되고, 뒤 수식의 합이 그대로 A의 PageRank가 된다. 이것이 바로 위 그림에서 가정한 상황이다. 반대로 d값이 작아져서 0이 되면, 뒤 수식의 합은 0이 되고, A의 PageRank는 1이 된다. d가 0이면 모든 페이지의 PageRank는 1이 되므로 아무 의미가 없어진다. 그래서 d는 실험을 통해 0과 1 사이의 어떤 값에서 정해지는데, 논문에서는 보통 0.85로 설정해놓았다고 되어 있다. 논문에 따르면 damping factor란 ‘어떤 마구잡이로 웹서핑을 하는 사람이 그 페이지에 만족을 못하고 다른 페이지로 가는 링크를 클릭할 확률’이다. 즉, damping factor가 1이면, 무한히 링크를 클릭한다는 뜻이고, 0이면 처음 방문한 페이지에서 무조건 멈추고 더 이상 클릭하지 않는다는 뜻이다. 0.85이면, 85%의 확률로 다른 페이지를 클릭해볼 것이라는 뜻이다. 이 경우 15%의 확률에 걸리는 순간 클릭을 멈추고 그 페이지를 살펴본다.

END