LC029 정보검색

2022년 11월 17일

Chapter 19 : Web Search Basics

Web Search Characteristics

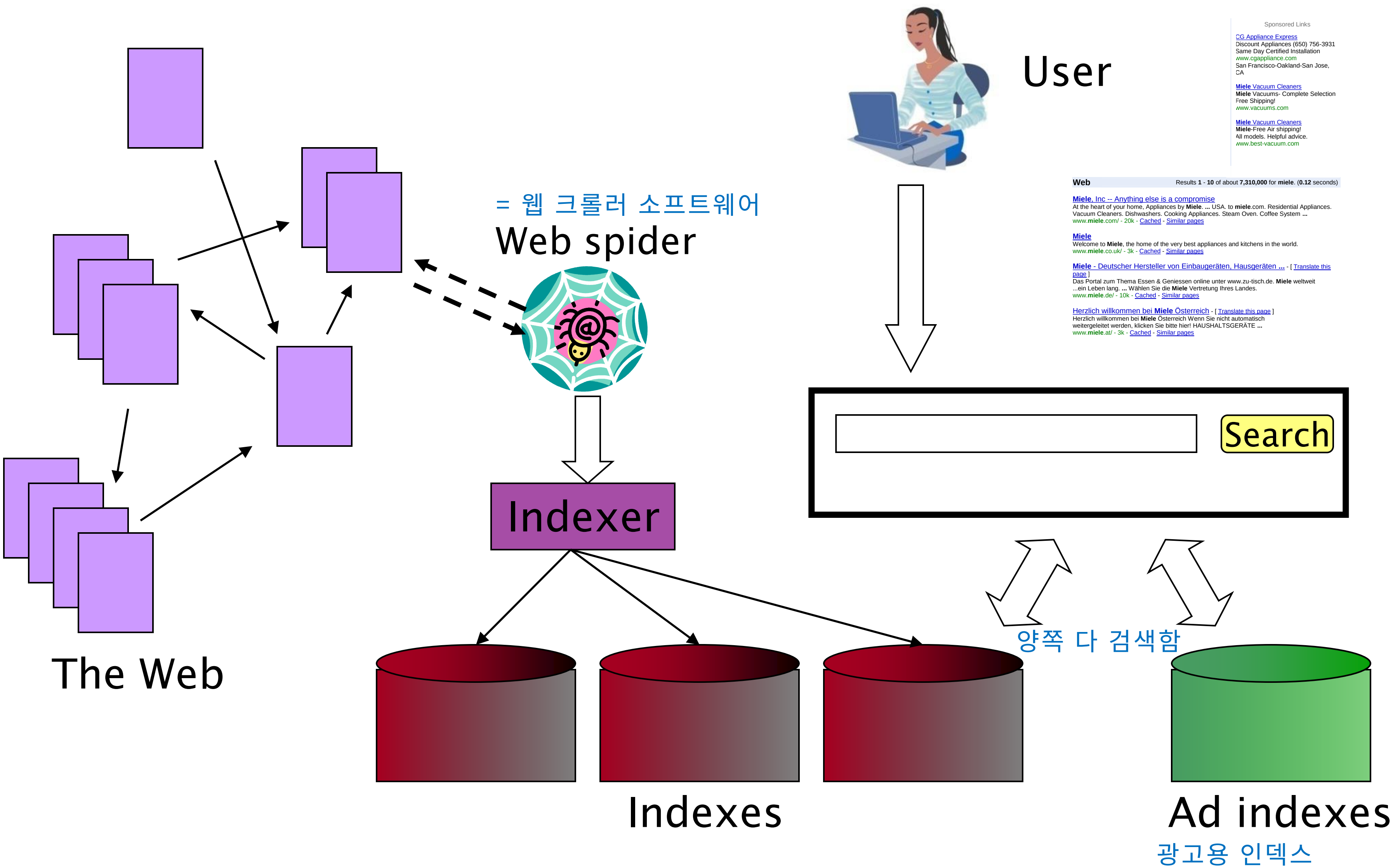
Advertising as Economic Model & Spam

Index Size Estimation

Duplicate Documents

Web Search Characteristics

Web Search Basics



Characteristics of Web document collection

- Creation
  중앙집중화 X : 디자인과 합동(co-ordination) 없음
  - Decentralized : no design and no co-ordination.
    분산되고 민주화된 생성
  - Distributed and democratized content creation.
    생성자의 배경이 다양하다
  - Diversity of backgrounds and motives of its participants.
- Contents
  최신, 구식, 진짜, 가짜가 모두 섞여있고 언어도 다양
  - Includes truth, lies, obsolete information, contradictions.
  - In many languages (many variations in grammar and style) -> require different stemmers and linguistic operations
  - Can be either static or *dynamically generated.*
    static한 문서도 있고 동적으로 생성되는 문서도 있음

Characteristics of Web document collection

dynamic generation of web pages

동적 생성 문서 예시 : 사용자가 조회할 때만 나오는 성적표
→ 요구할 때만 생성되므로 실제로 존재하는 문서가 아님

Characteristics of Web document collection

- Structures
  - Unstructured (text, html, …) 구조 없음
  - Semi-structured (XML, annotated photos) 약간 구조 있음, 반구조화된 문서
  - Structured (Databases) 구조 있음, 구조화된 문서
- Scale 스케일 : 점점 증가
  - Much larger than previous text collections.
  - By 1995, Altavista has crawled and indexed 30 M pages.
  - In 1995, the volume of web pages was doubling every few months.
  - The growth is now slowed down, but still expanding.

Web Information Retrieval

- Web Information Retrieval
  - Web document collections are useful when they are discoverable. 웹문서는 우리가 원하는 것을 찾을 수 있어야 한다. 검색 엔진 매우 중요
  - **Keyword-based search** : Altavista, Excite, Infoseek
    - Based on inverted index and ranking mechanism 이게 지금까지 배운 방식 불린, 벡터 모델
  - **Taxonomy-based search** : Yahoo 크롤러가 가져온 문서를 사람이 직접 카테고리별로 분류함
    - Convenient and intuitive for finding web pages.
    - For accurate and consistent classification, manual classification is needed. 수동 분류를 하기 때문에 키워드 검색보다 정보량이 적고 노동력도 많이 필요함
    - Manual classification entails significant human efforts.
    - 사람마다 주관이 달라 문서가 잘못 분류될 수 있음
    - Mismatch between users and editors of the taxonomy.
    - As the size of taxonomy grows, it is hard for users to find information.

Web Information Retrieval

- First generation of Web Information Retrieval
  - Used classical search techniques we have learned.
  - For indexing, query processing and ranking **at web scale**, they need to harness tens of machines together.
  - The **quality and relevance** of web search results are not satisfactory due to the idiosyncrasies of content creation on the web.

    스팸과 싸우는 테크닉도 필요하다더라
  - **New ranking** and **spam-fighting** techniques have invented to improve the quality of search results.
  - Need to measure the **authoritativeness** of a document based on cues such as which website hosts it.

    문서가 있는 사이트가 얼마나 권위있는 곳이냐

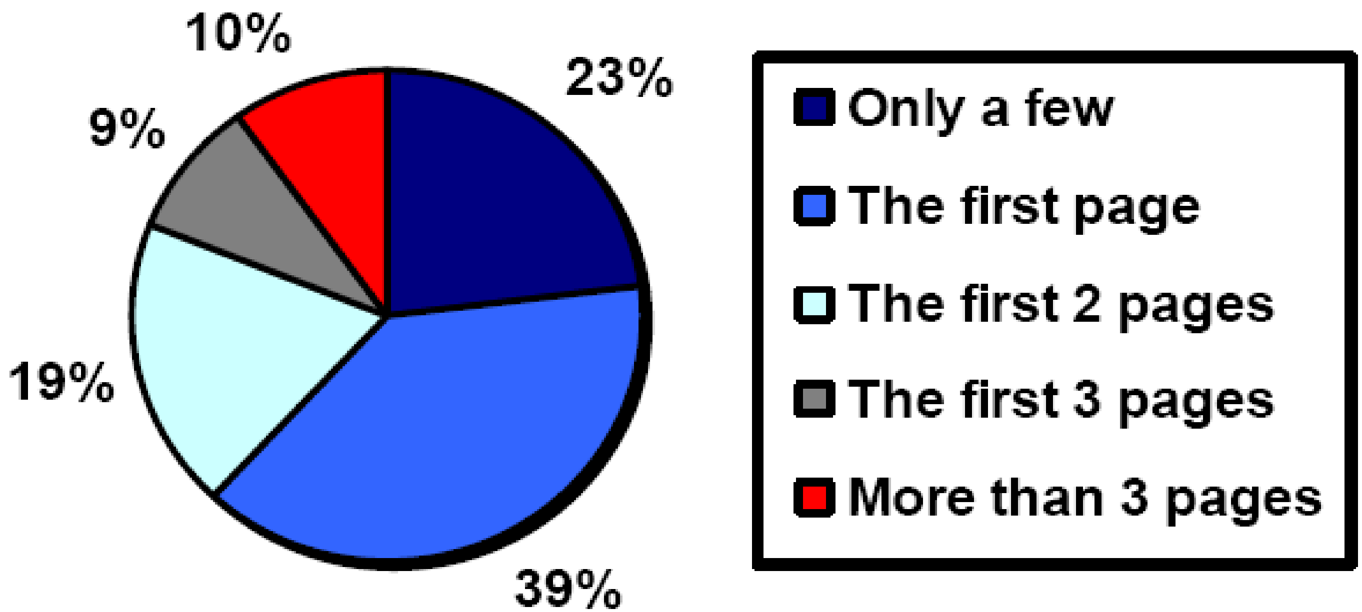
Web Search User Experience

Web Search User Experience

- Traditional IR 정통 정보 검색 : 보통 전문가가 쓰고, 잘 검색할 줄 알고 있음
  - Users were typically professionals.
  - They are trained in the art of phrasing queries.
- Web Search 웹 검색 : 비전문가가 사용, 검색도 단어 몇 개 주고 대충 함
  - Users do not know about the heterogeneity of web content.
  - They do not know the syntax of query language, the art of phrasing queries.
  - Average number of keywords is 2-3.
  - They seldom use operators such as AND, OR and wildcards.

How far do people look for results?

When you search something, how many entries do you typically review before clicking one? 검색 결과 몇 개나 보나?

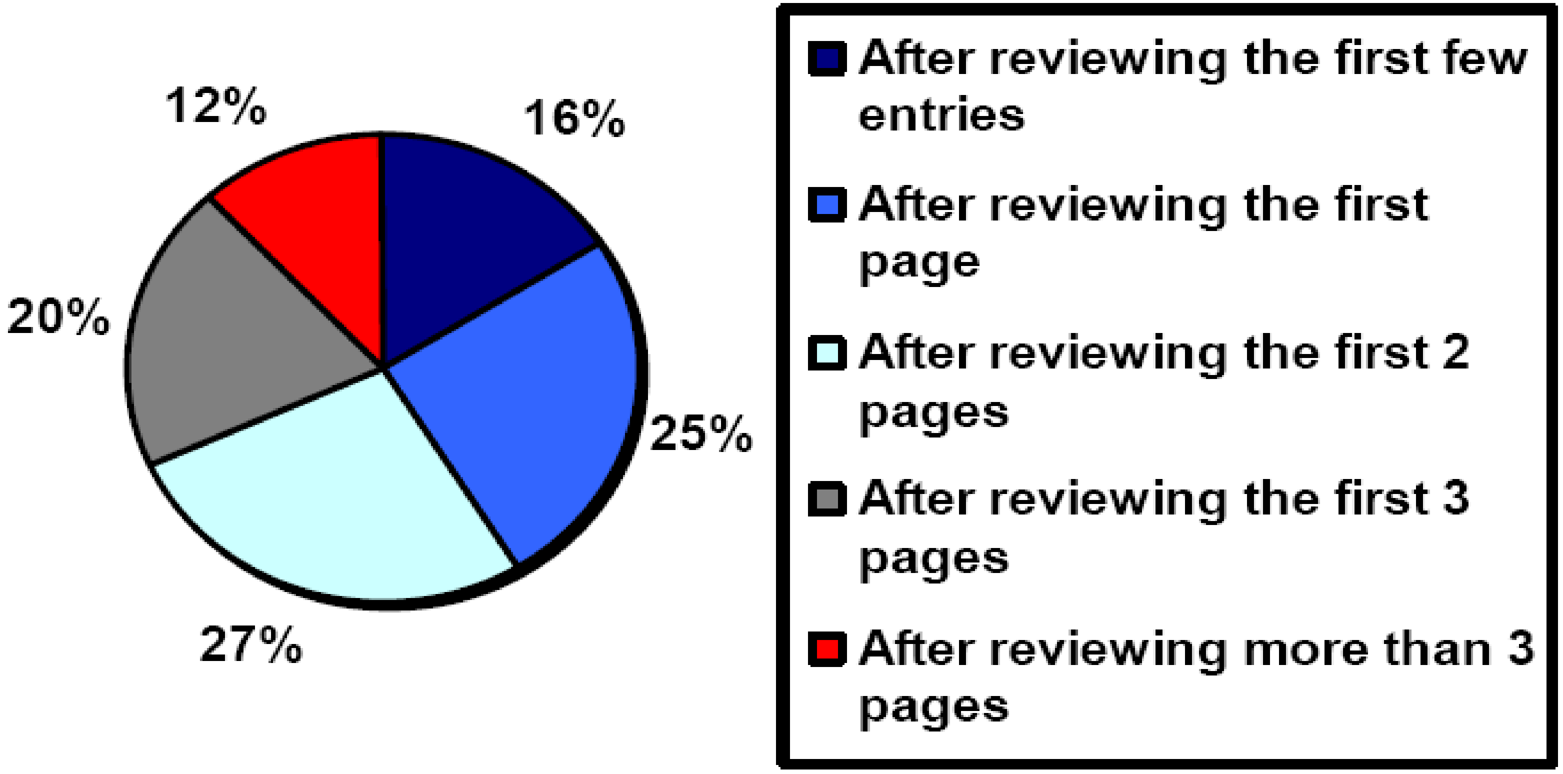


(Source: iprospect.com WhitePaper_2006_SearchEngineUserBehavior.pdf)

How far do people look for results?

If you don't find what you are looking for, when do you revise your search or move on to another search engine?

원하는 정보를 못찾았을 때 얼마나 빨리 재검색하나



User Query Needs

- 검색하는 사람의 검색 목적
  Web search queries are categorized into three groups.
  - Informational 정보 필요 `Low hemoglobin`
    - Want to learn about something (40% ~ 65%).
    - **Try to assimilate information from multiple web pages.**
  - Navigational 목적지가 있음 `Sungshin Women's University`
    - Want to go to that page (15% ~ 25%).
    - **Measure of user satisfaction is precision at 1.**
  - Transactional 상거래, 이미지 다운로드 등 데이터의 이동
    - Want to do something (20% ~ 35%).
      - Purchasing a product `Apple ipod`
      - Downloading a file `Mars surface images`
      - Accessing a service `Road condition of Seoul`
    - **Search engine should return results listing those services.**

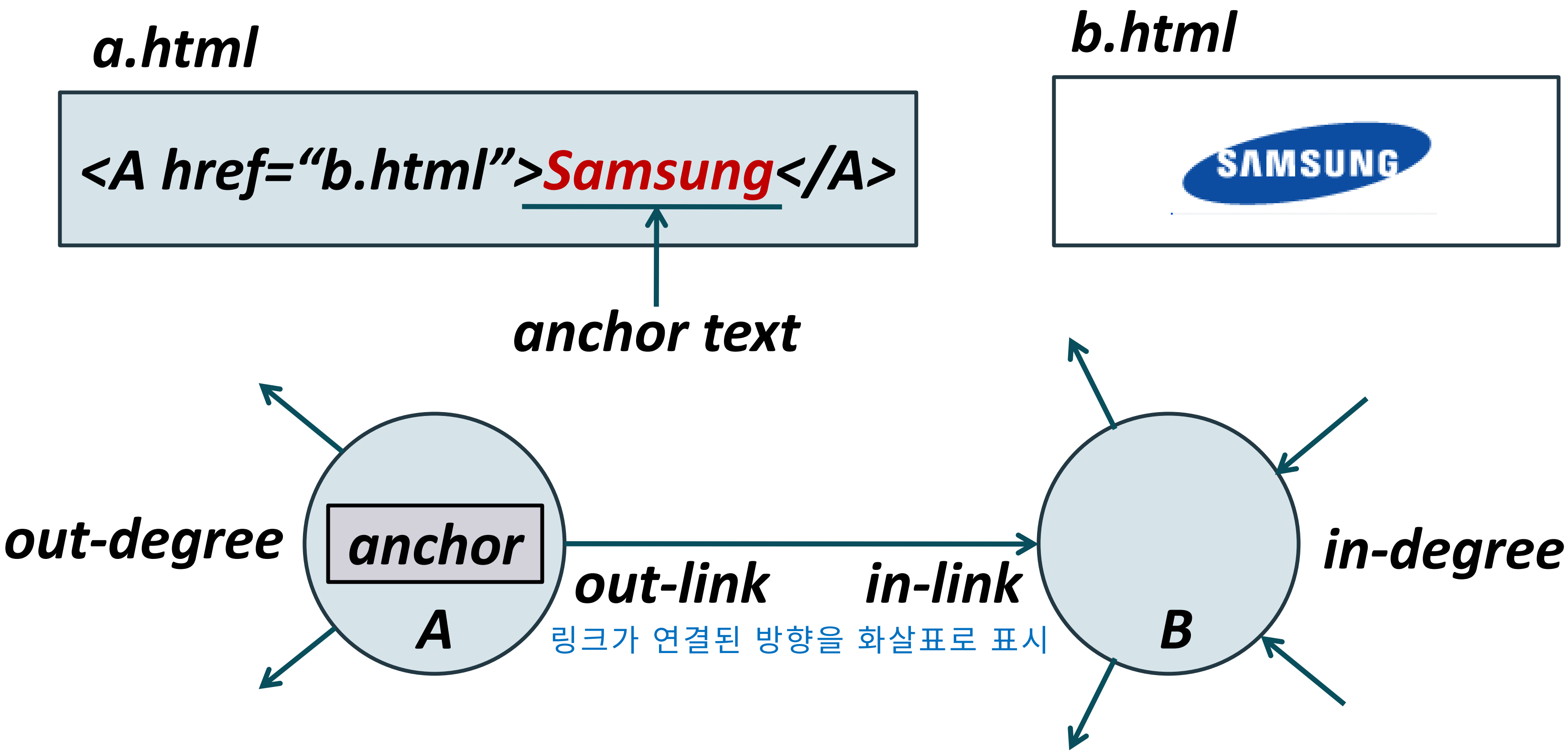
User Query Needs

- Other than that, users want to: 예 : 렌트카 회사들 뭐 있나 보기
  - 특정 부류가 모여있는 허브를 찾고 싶음
  - Find a good hub. `Car rental Brasil`
  - Do an exploratory search to see what is going on.

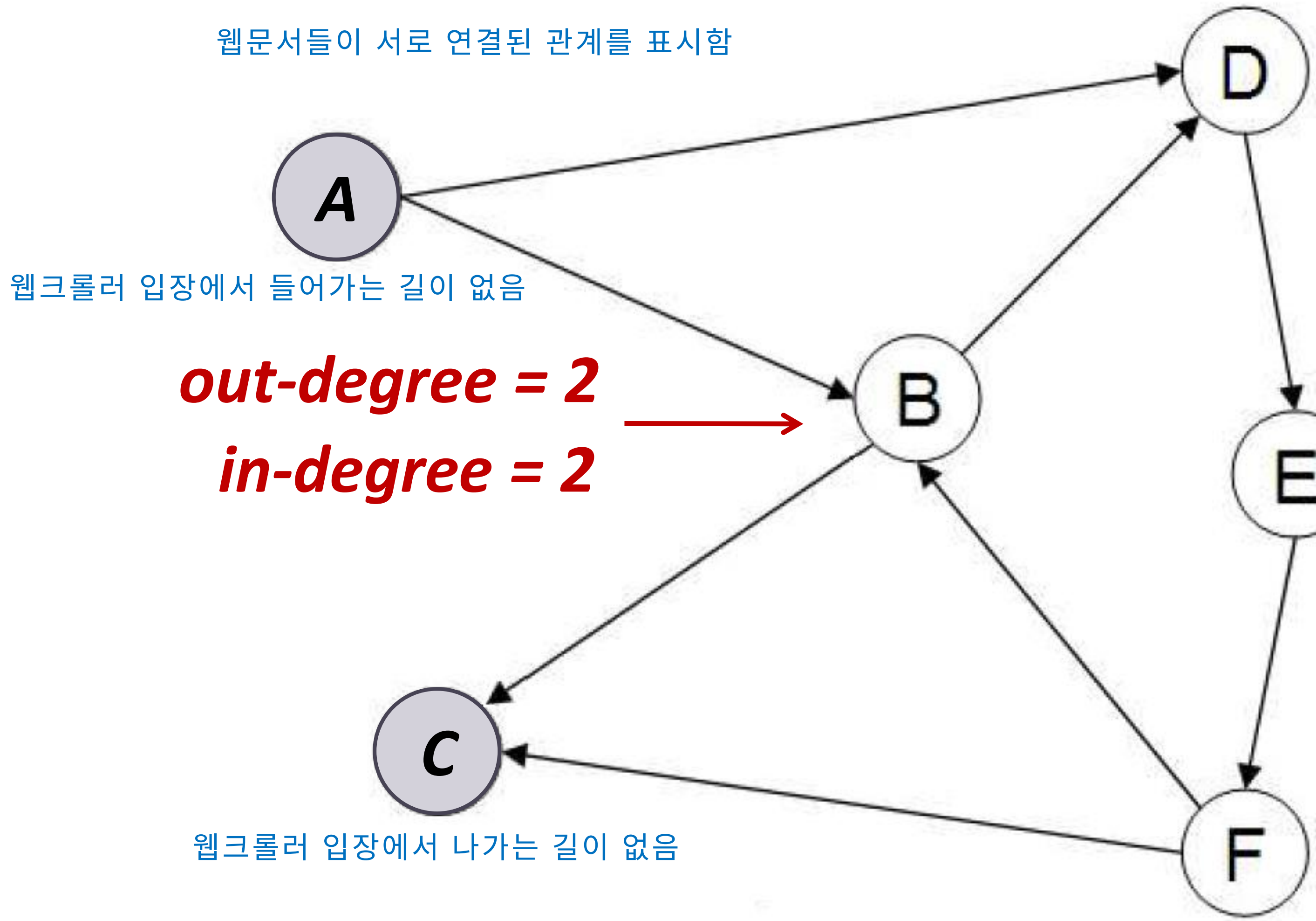
Web Graph

Web Graph

- Web documents have hyperlinks between them as a directed graph.



Web Graph



웹문서들이 서로 연결된 관계를 표시함
D
A
웹크롤러 입장에서 들어가는 길이 없음
out-degree = 2
in-degree = 2
B
E
C
F
웹크롤러 입장에서 나가는 길이 없음

Web Graph

링크는 랜덤하게 분포하지 않는다

- The links are not randomly distributed.
  웹페이지 링크 분포는 푸아송비를 따르지 않는다
  - The distribution of the number of links *into* a web page does not follow the Poisson distribution.
  지수
  - The distribution follows **power law**:
    The number of web pages with in-degree $i \propto \dfrac{1}{i^{\alpha}}$ where α is typically 2.1.

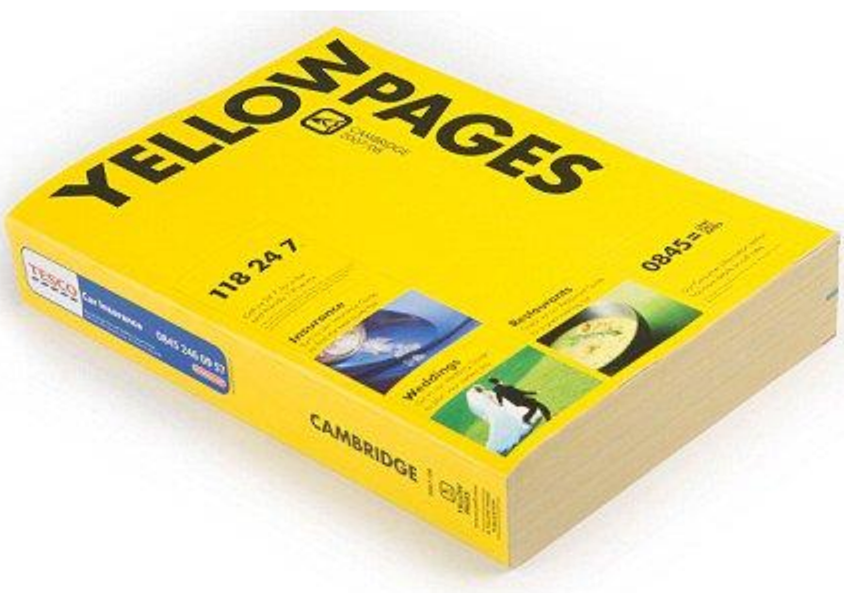
when i = 1, 1/1$^{2.1}$ = 1.00	들어가는 링크 개수에 따라 그런 문서가 존재하는 비율
when i = 2, 1/2$^{2.1}$ = 0.23	
when i = 3, 1/3$^{2.1}$ = 0.10	
when i = 4, 1/4$^{2.1}$ = 0.05	
when i = 5, 1/5$^{2.1}$ = 0.03	

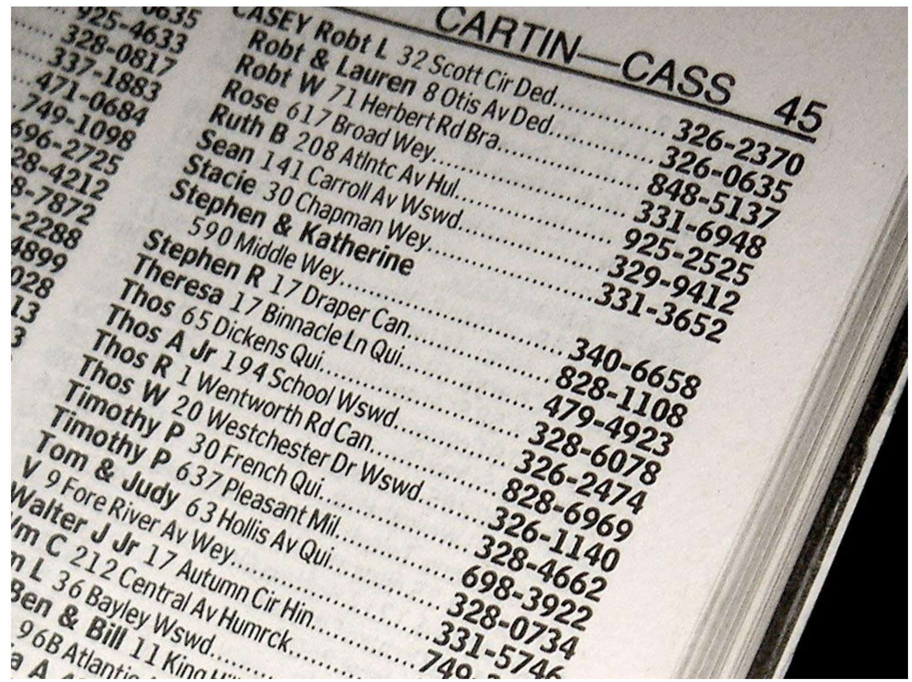
Advertising as Economic Model

광고에 대한 내용임

Paid Inclusion Model

- The Yellow Pages 전화번호부
  - 돈을 내면 글자를 굵고 크게 써준다
  - Companies pay for larger / darker fonts.
    - It is fair because they pay for that.





Paid Inclusion Model

- The Yellow Pages
  - Companies pay for larger / darker fonts.
    - It is fair because they pay for that.

Paid Inclusion Model

- The Yellow Pages

  꼼수 : 사전순 정렬을 이용해 전화번호부 앞면을 차지하고 이름도 요상하게 지음(무료)
  - Some companies use some tricky method to list their name early in the list.
    - **aaaaaaCheapestPhoneCompanyaaaaa**
    - They do not pay for that.
  - Web search engine followed the Yellow Pages' paid inclusion model.

CheapestPhoneCompany
: 폰 회사가 뭐가 있는지 모르는 사람이 "가장 싼 폰 회사" 알려주세요 했는데 실제로 있는 회사였음(비쌈)

Advertisement

- Early Stage
  브랜드 자체를 배너 광고
  - **Banner advertisement** at popular websites (MSN, CNN, …)
  - The purpose of the advertisement was to convey a positive feeling about their **brand**.
    배너 노출 횟수에 따른 요금
  - The advertisement was priced on CPM (cost per mille).
- Later
  - As the goal of the advertisement was to induce a **transaction**, the pricing model was changed to CPC (cost per click). 광고의 목표는 실제 트랜잭션이 일어나게 하는 것.
    클릭 횟수에 따른 요금

Sponsored Search : GOTO

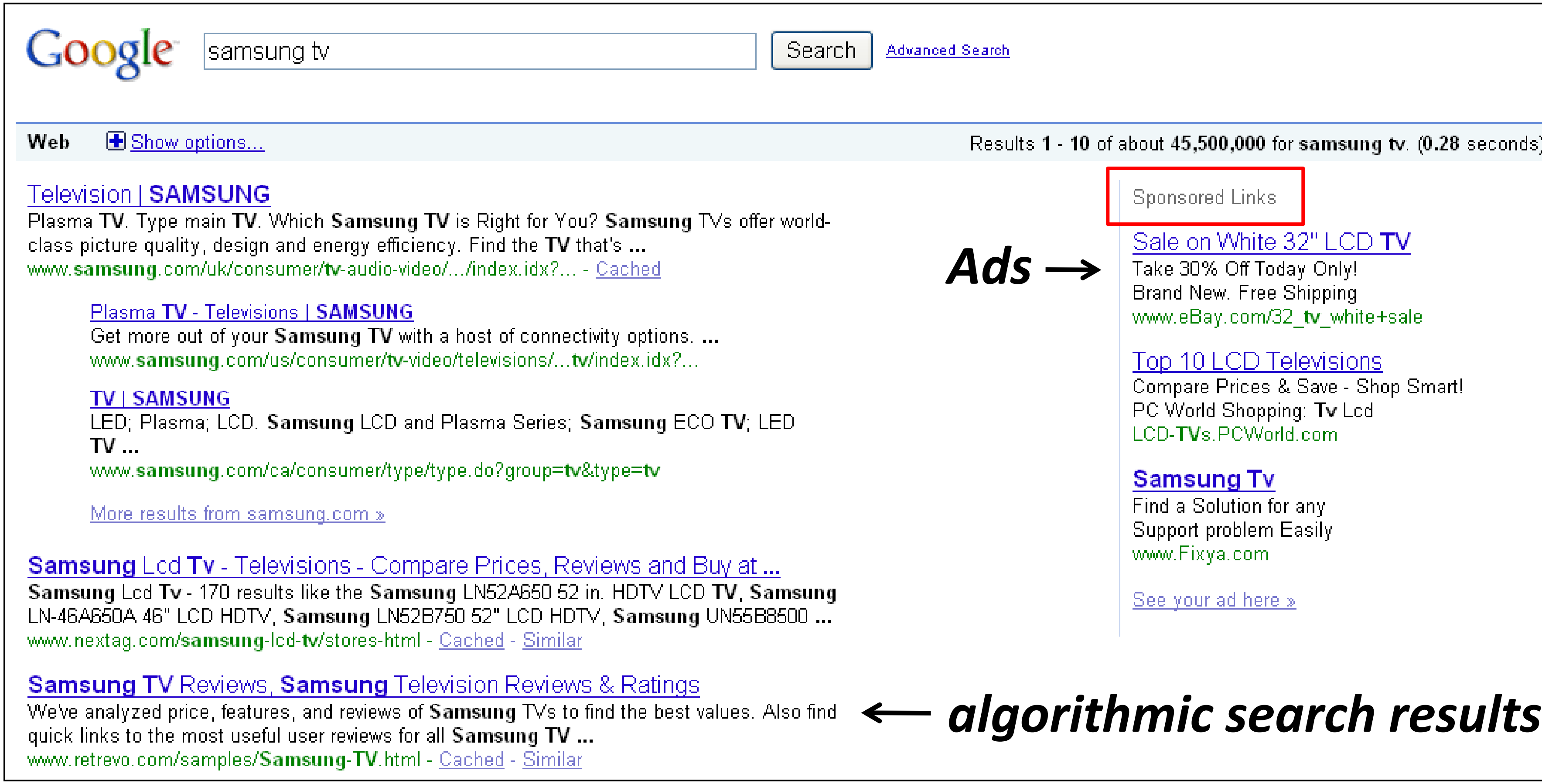
- Goto was not a search engine in the traditional sense.
  돈을 많이 낼수록 검색 결과 상위에 나옴
  - Your search ranking depended on how much you paid.
    사용자가 클릭하면 GOTO가 수수료 받음
  - When the user clicked on a result page, Goto was compensated by the corresponding advertiser.
- Goto users are actively expressing an interest and intent related to the query term.
  - A user typing **iphone** is more likely to buy it than simply browsing news about **iphone**. 사용자도 어차피 뭘 사려고 검색하는거다
  - Auction for keywords: ***casino*** was expensive!
    키워드 경매 : 이 검색어에 대해 상위노출 얼마까지 낼 수 있냐

Sponsored Search : GOTO

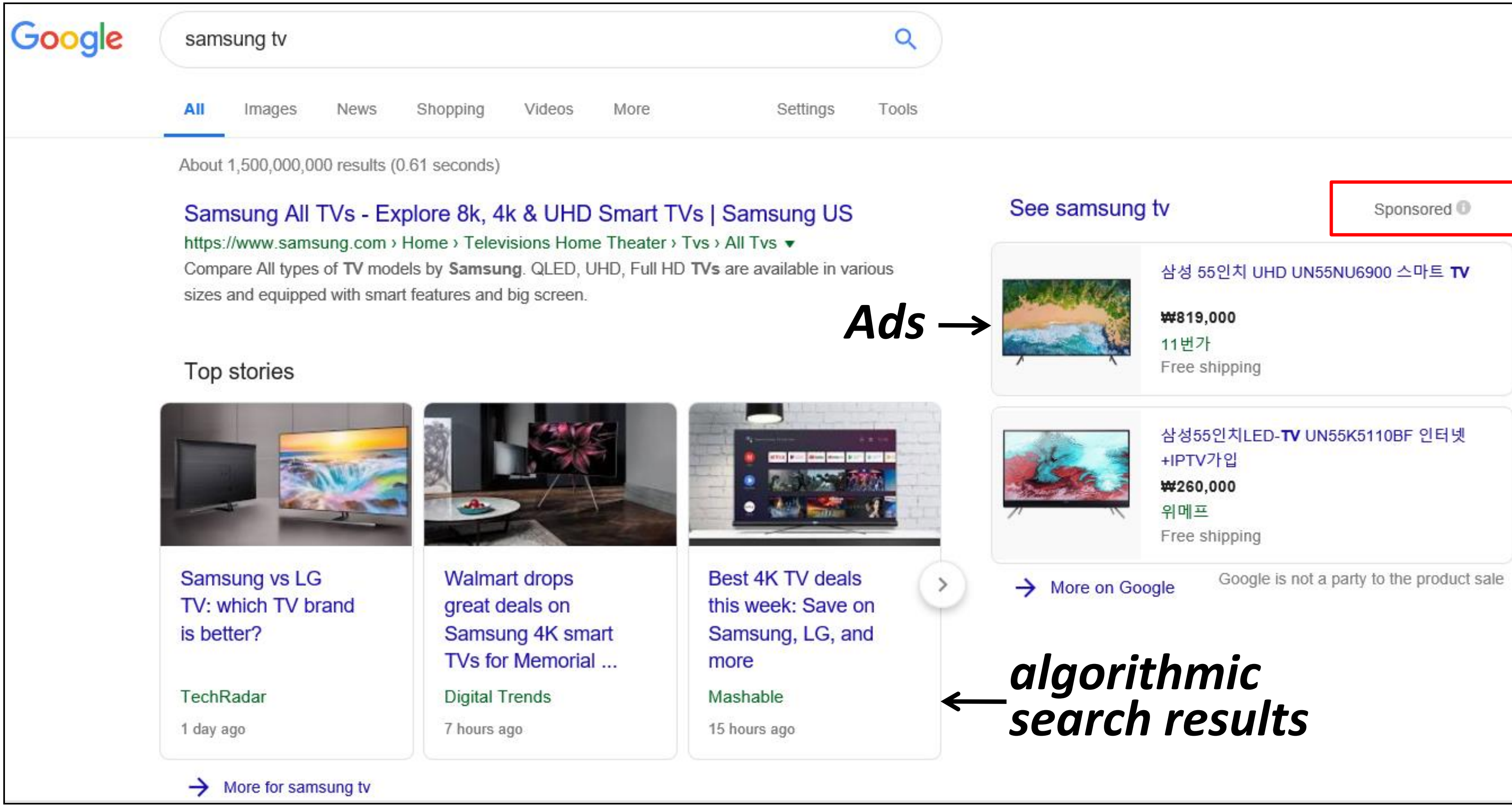
- Goto was morphed into Overture, and finally acquired by Yahoo!
  - Meanwhile Goto/Overture’s annual revenues were nearing $1 billion.
- Google added paid-placement ads to the side, independent of search results.
  - Pure search engine + **Sponsored search** engine

    구글도 검색 결과에 광고 섞임

Sponsored Links



Sponsored Links



Sponsored Links

Spam

Trouble with Sponsored Links

- To rank highly, it costs money.

광고를 걸지 않고 알고리즘이 내 글을 위에 올려주게 하기

- An alternative is ***Search Engine Optimization (SEO)****:*
  - Tuning your web page to rank highly in the **algorithmic search results** for selected keywords.
  - It is an alternative to paying for placement, thus, intrinsically a marketing function.
  - Performed by companies (**Search Engine Optimizers**) for their clients. 회사에 돈 주고 맡길 수 있음
  - Some perfectly legitimate, some very shady.

잔머리도 있고 어둠의 루트도 있다고 한다

키워드 채우기

Search Engine Optimization : **Keyword Stuffing**

- First generation Search Engines relied heavily on *tf·idf*. Web pages with high *tf* would rank highly.
  - This led to the first generation of spam.
  - Sophisticated spammers rendered the repeated terms in the same color as the background.
    - Repeated terms got indexed by indexers.
    - But not visible to humans on browsers.
  - Thus, pure word density cannot be trusted as an IR signal.

문서의 tf idf 값을 높인다. idf는 못건드리고 tf는 바꿀 수 있다.
배경과 똑같은 색으로 텍스트 잔뜩 써넣기

Search Engine Optimization : **Keyword Stuffing**

- They also use meta-tags. 메타 태그 채우기

```
<meta name="description" content="… London hotels, hotel, holiday inn, hilton, discount, booking, reservation, mp3, britney spears, …" />
```

  - **Meta tags** are snippets of text that describe a page's content.
  - Meta tags don't appear on the page itself, but only in the page's source code.
  - Meta tags are essentially little content descriptors that tell Search Engines what a web page is about.

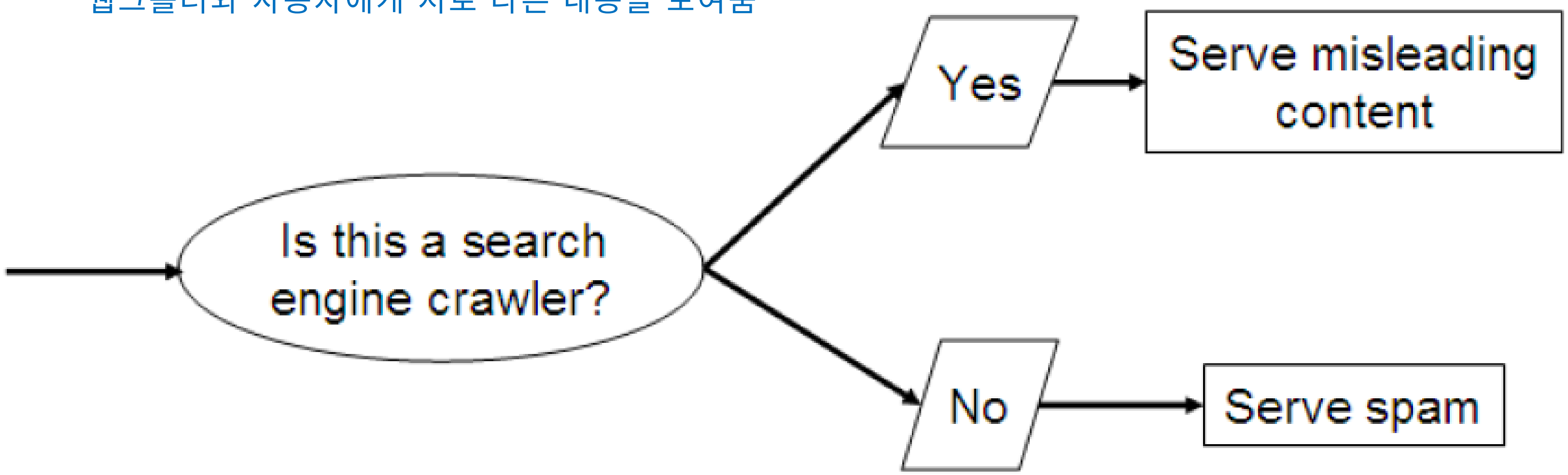
As a result, ...

- Search Engine became sophisticated enough to screen out a large number of repetitions of particular keywords.
- Spammers responded with a richer set of spam techniques.
  - Cloaking
  - Doorway Page
  - Click Spam

Search Engine Optimization : **Cloaking**

- Serve fake content to search engine spider.
  - Indexed by the search engine under misleading keywords.
- When users search for these keywords, they receive different web pages.

웹크롤러와 사용자에게 서로 다른 내용을 보여줌



Search Engine Optimization : **Doorway Page**

- A doorway page contains text and meta data carefully chosen to rank highly on selected keywords.
- When a browser request the doorway page, it is redirected to a more commercial page.

```
<HTML>     링크가 변경되었을 때 자동으로 새 링크로 이동하게 함
           + 크롤러와 사용자에게 서로 다른 내용을 전달함
<HEAD>
<meta http-equiv="refresh" content="1" ; url="fake.com" />
</HEAD>
<BODY>
... text and metadata carefully chosen to rank highly on
   selected keywords ...
</BODY>
</HTML>
```

Search Engine Optimization : **Click Spam**

- No universally accepted definition of click spam.

  딱 정의되지는 않았음

- Example
  - Repeatedly clicking on *sponsored search results* to exhaust the advertising budget of a competitor.

    예 : 오토마우스
    이용 : 봇으로 경쟁사의 광고링크 클릭 횟수를 높여서 광고비를 소진하고 광고를 내리게 만든 다음
    자사의 광고를 올림

Index Size Estimation

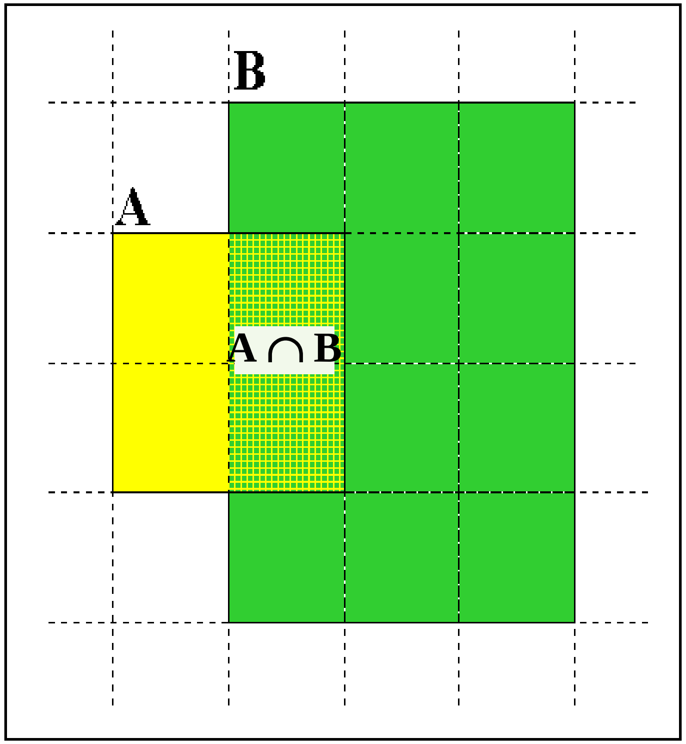
Index Size Estimation

- What is the size of web?
  - The web is really infinite.
  - Static web
    contains duplication, mostly due to mirroring (~40%).
  - Dynamic web
    e.g. calendar, HTTP 404

HTTP 404 - 파일을 찾을 수 없음
Internet Explorer

- What can we measure? 서로 다른 검색 엔진의 인덱스 사이즈 비교하는 방법 배운다
  - The static web pages are whatever search engines have to index.
  - To know the coverage of a search engine relative to another, we measure the **relative index sizes**.

Relative Index Sizes of Two Engines



참고 : 몬테카를로 방법

개신기하고 재밌어보임

Sample choose URLs randomly from A.
Check if contained in B.
→ A ∩ B = (1/2)×Size A

Sample choose URLs randomly from B.
Check if contained in A.
→ A ∩ B = (1/6)×Size B

(1/2)×Size A = (1/6)×Size B

∴ Size A / Size B = (1/6) / (1/2) = 1/3

Size A = 1/3 X Size B
Size B = 3 X size A

Each test involves: (i) Sampling URLs (ii) Checking

Relative Index Sizes of Two Engines

- Ideal Strategy
  - 랜덤하고 균일하게 분포하는 링크를 생성해야 한다
  - Generate a random, *uniformly-distributed* URL.
  - Check for containment in the index of each engine.
  - 각 엔진의 인덱스가 그것을 포함하는지 확인한다
- Problem
  - 랜덤하고 균일하게 분포하는 링크를 어디 가서 찾냐
  - Random, uniformly-distributed URLs are hard to find!
  - It is enough to generate a *random URL contained in a given Engine*. 타협 : 양쪽 엔진에서 랜덤추출
    - Approach 1: Random Searches
    - Approach 2: Random IP Addresses
    - Approach 3: Random Queries

링크는 IP 주소로 변환된다 → IP 주소는 4개의 숫자로 이루어진다
→ 4개의 랜덤한 숫자를 생성하면 랜덤한 링크로 이동할 수 있다
문제 : 랜덤은 되지만 균일함은 보장할 수 없다

Sampling URLs

- Approach 1: Random Searches
  - Send a random search from a search log. == 쿼리 로그
    - Search log is an accumulation of all search queries of a work group.
    - This may include the ***bias*** from the types of searches made by the work group. 기존 사용자의 편향에 영향을 받는다
  - Pick a random page from the search results.
  - Check if the page is contained in the other engine.

쿼리 로그를 이용해 아무 쿼리나 검색하고
검색 결과를 랜덤하게 고름 → 검사에 사용

Sampling URLs

- Approach 2: Random IP Addresses
  - Generate a random IP address. IP 주소 랜덤 생성
  - 해당 웹서버에 문서 요구
  - Send a request to a *web server of that address*.
  - Collect all pages at that server. 서버의 모든 문서 수집
  - Check if those pages are contained in each engine.
  - 각 엔진이 수집한 문서를 갖고있는지 확인

Sampling URLs

- Approach 3: Random Queries 랜덤 쿼리
  - 사전에서 랜덤 단어 뽑기
  - Pick a set of random terms from dictionary.
    - **Lexicon:** 400,000+ words from a web crawl
  - Form a random query with two or more terms connected conjunctively. 뽑은 단어로 질의 생성(단순, 복합 질의)
  - 만든 쿼리로 검색
  - Do search with the random conjunctive query.
  - 검색 결과 랜덤 뽑기
  - Pick a page *p* at random from the top 100 returned results.
  - Check if the page *p* is contained in the other engine.

반대쪽 엔진도 해당 결과를 갖고있는지 확인
: 문서의 내용을 그대로 검색해서 같은 문서가 나오는지 확인하면 됨

Duplicate Documents

Duplicate Documents

- The web is full of duplicated content.
  - Mirroring for reliability 똑같은 문서를 여러 개 만들어둔다 : 여분 세이브
  - 40% of the web pages are duplicates of other pages.
  - No need to index multiple copies of the same pages.
    - Save storage space 용량 절약
    - Reduce processing overheads 검색 시간 감소

남의 글 복붙해서 올리거나 복사해와서 약간 수정해서 올리는 글이 많아 중복되는 문서가 많다
→ 중복 인덱싱 발생, 링크는 다른데 내용은 똑같은 문서들이 있음
→ 사용자의 불편을 줄이기 위해 중복 문서는 인덱스에서 빼라

Duplicate Documents

- Duplication 완전 중복된 문서
  중복된 두 문서가 완전히 동일한지(카피인지) 확인할 때 사용하는 핑거프린트
  - Can be detected with a ***fingerprint***, a succinct (say 64-bit) digest of the characters on a page.
    핑거프린트 : 문서 전체의 아스키코드 중 랜덤한 64개의 비트
  - If two fingerprints are equal, we test whether the pages are **really equal.**
    핑거프린트가 다르다 : 확실하게 다른 문서임
    같다 : 동일한 문서일 가능성이 있으니까 이것만 전수조사 하면 됨
- Near-Duplication 거의 비슷한 문서 : 핑거프린트 못씀
  - There are many cases of near duplication on the web.
  - Compute similarity of two documents.
    - **Jaccard Coefficient** can be used. 자카드 계수 사용
  - Use similarity threshold to detect near duplicates.
    - e.g. Similarity > 80% => Documents are “near duplicates”

Jaccard Coefficient

- Measurement of the overlap of two sets *A* and *B.*
  - Jaccard(A, B) = $|A \cap B| / |A \cup B|$
- Always assigns a number between *0* and *1*.
  - Jaccard(A, A) = 1
  - Jaccard(A, B) = 0 if $A \cap B = \emptyset$
- A and B don't have to be the same size.
- Used to measure the similarity of A and B.

Jaccard Coefficient: Scoring Example

- For each of the two documents below, calculate Jaccard Coefficient of them.
  - Document 1: caesar died in march
  - Document 2: the long march in Korea

Jaccard($d1$, $d2$) = 2/(4 + 5 – 2) = 1/7 = 0.143

Computing Similarity

- Are the documents duplicates of each other?

d1

can a can can a can? a rose is a rose. she sells sea shells.

d2

she sells a rose. a rose is a can. can a can can sea shells.

- Jaccard(*d1*, *d2*) = 15/(15 + 15 – 15) = 1.0
  Therefore, d1 is a duplicate of d2, or vice versa.
- In reality, they are not!!

한계 : 단어만 같고 배열이 다른 경우는 걸러낼 수 없다

Computing Similarity

- Use *k*-gram of terms. 단어 단위 k그램 예제는 4그램

d1

can a can can a can? a rose is a rose. she sells sea shells.

can a can can
a can can a
can can a can
can a can a
a can a rose
can a rose is
…

d2

she sells a rose. a rose is a can. can a can can sea shells.

she sells a rose
sells a rose a
a rose a rose
rose a rose is
a rose is a
rose is a can
…

원문과 그램을 모두 포함해 자카드 계수 계산한다

Computing Similarity

- Shingling of a document
  - *k*-shingles of a document *d* are defined to be a set of all *k*-grams in *d*. k싱글은 한 문서에 존재하는 모든 k그램의 집합이다
  - ***a rose is a rose is a rose*** (assume *k* = 4)

    a_rose_is_a

    rose_is_a_rose

    is_a_rose_is

    a_rose_is_a
- Two documents are near duplicates if the sets of shingles are nearly the same.

Computing Similarity

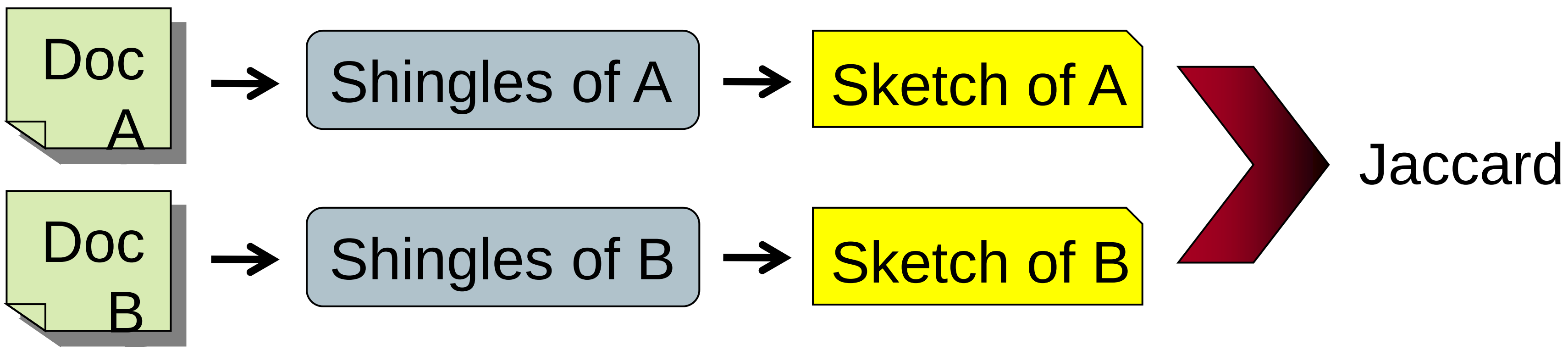
- S(d_j) : the set of shingles of document d_j
- Similarity is measured by

$$\mathrm{Sim}(d_i, d_j) = \mathrm{Jaccard}(S(d_i), S(d_j)) = \frac{|S(d_i) \cap S(d_j)|}{|S(d_i) \cup S(d_j)|}$$

- We have to compute Jaccard coefficients **for all pairs of web documents**.
- Computing exact set intersection of shingles between all pairs of web documents is **expensive** and **intractable**. 웹문서가 너무 많아서 이 방법을 진짜 쓰기는 힘들다

Computing Similarity

- **Approximate** using a cleverly chosen subset of shingles for each document. 두 문서의 싱글들의 부분집합을 만들어서 자카드 계수 계산
- This subset is called a **sketch** of the document.
- Calculate similarity based on the short sketch.



Sketch of a Document

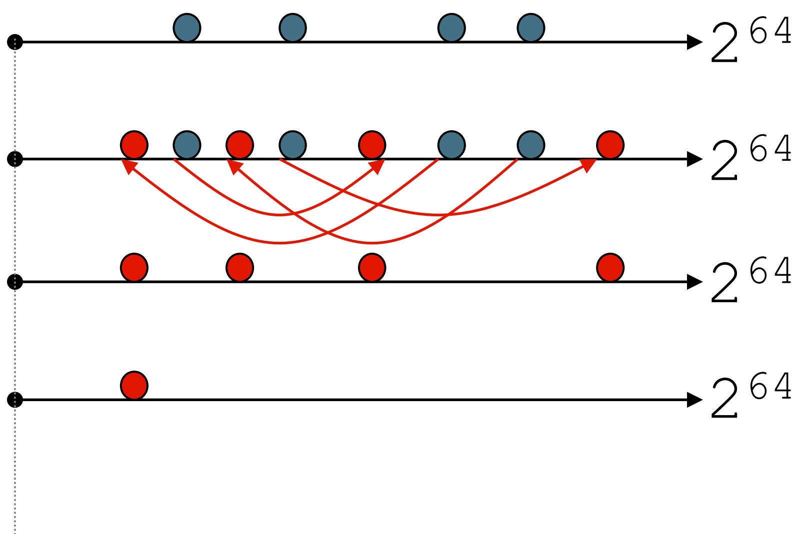
h : 문자열을 숫자로 바꿔주는 해시함수

- For a document *d*, a *sketch$_d$* is given as follows:
  - Let ***h*** be a hash function that maps all shingles in the universe to [0..2^m].

    h(a_rose_is_a) = 921037246

  - 랜덤하게 섞는다

    Let π_i be a *random permutation* on [0..2^m].
  - Pick MIN {π_i(h(s))} over all shingles *s* for *d*. (=> *sketch$_d$*)
    It is much like a *card shuffling* and choosing the first one.



문서의 싱글을 해시함수로 돌리고 표시

Start with 64-bit ***h*(shingles)**

πi 규칙에 따라 점을 뒤섞음

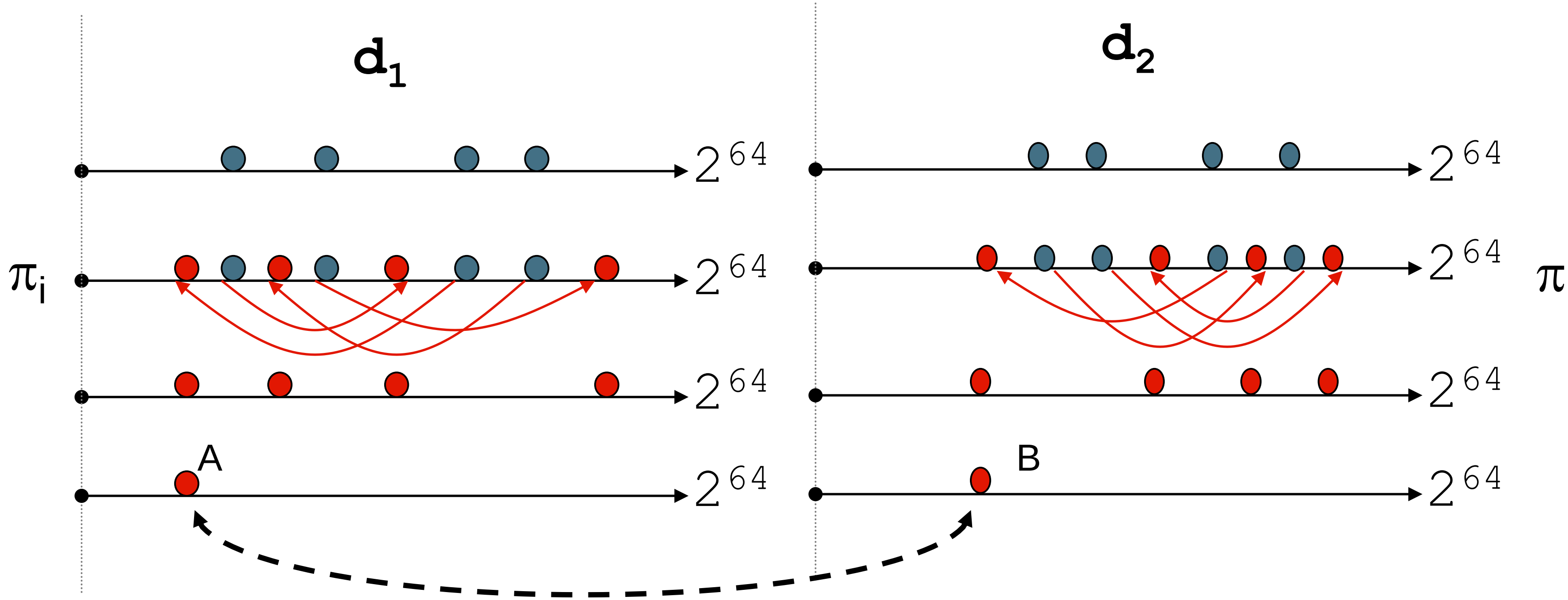
Permute on the number line with π_i.

After permutation.

Pick one with the min value.

가장 작게 나온거 하나만 고르기 → 문서의 스케치

Test if ($Sketch_{d_1} == Sketch_{d_2}$)



Are these equal?

It is much like a *card shuffling for both decks* and choosing the first one *from each deck, and seeing if they are the same one.*

Sketch of a Document

- Let S(d1) = {"apple", "banana", "berry"}

  S(d2) = {"cherry", "apple", "lemon", "berry"}

- After hashing

편의상 단어로 써있지만 싱글이라 보자

S(d1)	S(d2)
	cherry
berry	berry
	lemon
apple	apple
banana	

Sketch of a Document

- After Permutation

S(d1)	S(d2)
	cherry
berry	berry
	lemon
apple	apple
banana	

Sketch

S(d1)	S(d2)
	lemon
	cherry
apple	apple
berry	berry
banana	

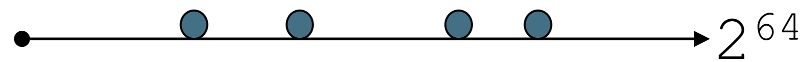
S(d1)	S(d2)
berry	berry
banana	
	cherry
apple	apple
	lemon

Sketch of a Document

- Let S(d1) = {"red", "blue", "purple", "green", "yellow"}
  S(d2) = {"blue", "orange", "purple", "violet"}

편의상 단어로 써있지만 싱글이라 보자

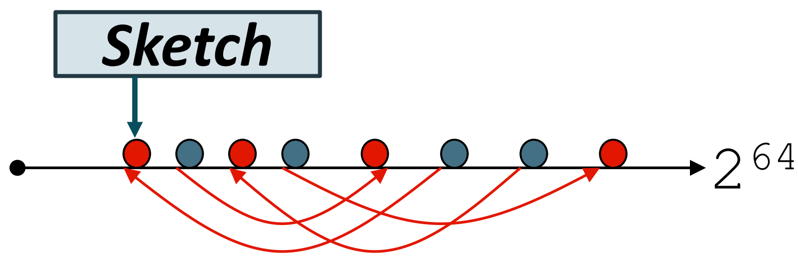
- After hashing
  S(d1) = {1, 2, 4, 5, 7}
  S(d2) = {2, 3, 4, 6}



- Assume the following permutations

$\pi_1 = \{(1 \rightarrow 7), (2 \rightarrow 2), (3 \rightarrow 4), (4 \rightarrow 1), (5 \rightarrow 3), (6 \rightarrow 6), (7 \rightarrow 5)\}$
$\pi_2 = \{(1 \rightarrow 4), (2 \rightarrow 6), (3 \rightarrow 5), (4 \rightarrow 2), (5 \rightarrow 1), (6 \rightarrow 7), (7 \rightarrow 3)\}$
$\pi_3 = \{(1 \rightarrow 3), (2 \rightarrow 4), (3 \rightarrow 1), (4 \rightarrow 5), (5 \rightarrow 2), (6 \rightarrow 7), (7 \rightarrow 6)\}$

- Apply π_1 to S(d1) and S(d2)
  π_1(S(d1)) = {7, 2, 1, 3, 5}
  π_1(S(d2)) = {2, 4, 1, 6}



Sketch of a Document

- After hashing   아까의 그 문서들에 다른 π 규칙을 적용해보자

  S(d1) = {1, 2, 4, 5, 7}, S(d2) = {2, 3, 4, 6}

- Assume the following permutations

  $\pi_1 = \{(1\rightarrow 7), (2 \rightarrow 2), (3\rightarrow 4), (4\rightarrow 1), (5\rightarrow 3), (6\rightarrow 6), (7\rightarrow 5)\}$
  $\pi_2 = \{(1\rightarrow 4), (2 \rightarrow 6), (3\rightarrow 5), (4\rightarrow 2), (5\rightarrow 1), (6\rightarrow 7), (7\rightarrow 3)\}$
  $\pi_3 = \{(1\rightarrow 3), (2 \rightarrow 4), (3\rightarrow 1), (4\rightarrow 5), (5\rightarrow 2), (6\rightarrow 7), (7\rightarrow 6)\}$

- Apply π_1, π_2, π_3 to S(d1) and S(d2)

  π_1(S(d1)) = {1, 2, 3, 5, 7}, π_1(S(d2)) = {1, 2, 4, 6}
  π_2(S(d1)) = {1, 2, 3, 4, 6}, π_2(S(d2)) = {2, 5, 6, 7}
  π_3(S(d1)) = {2, 3, 4, 5, 6}, π_3(S(d2)) = {1, 4, 5, 7}

  permutation을 3번 했는데 그 중 1번만 같았다 → 자카드 계수의 근사값을 1/3로 본다
  형관펜 친 부분 : 이 예시는 싱글 수가 적어서 중복 스케치가 나왔지만
  실제로는 싱글이 아주 많아서 중복 스케치가 나올 일이 거의 없다

  문서끼리 스케치를 비교할 때는 순서대로 1대1로 비교한다

Computation of $Jaccard(S(d_1), S(d_2))$

- Theorem
  $Sim(d_1, d_2) = Jaccard(S(d_1), S(d_2)) = P(\text{sketch}_{d_1}, \text{sketch}_{d_2})$
- Approximate $Jaccard(S(d_1), S(d_2))$
  - Create a **sketch vector** $sketch_d$[] (of size < 200) for each document d_1 *and* d_2 200번 섞어서 스케치 고르기
    → Do 200 random permutations: $\pi_1, \pi_2, \ldots, \pi_{200}$
  - If ($sketch_{d_1}[i] == sketch_{d_2}[i]$) more than t times, then they are **near duplicates.** 두 문서 사이에 똑같은 스케치가 많을수록 문서가 비슷함
  - t is determined empirically.

Computation of $Jaccard(S(d_1), S(d_2))$

- Is sketch efficient?
  - Assume the $S(d_1)$ and $S(d_2)$ are shingles of two documents
  - To compute Jaccard of $S(d_1)$ and $S(d_2)$, we have to know $S(d_1) \cap S(d_2)$ which requires $|S(d_1)| \times |S(d_2)|$ operations
  - Now using the sketch, the number of operation reduces $|S(d_1)| + |S(d_2)|$ for each permutation.

    이거 잘못 쓴 거임 스케치 개수 더하는 거임

    As we repeat the permutation P times, total operation required is:

    $(|S(d_1)| + |S(d_2)|) \times P \quad \ll \quad |S(d_1)| \times |S(d_2)|$

Computation of *Jaccard*($S(d_1)$, $S(d_2)$)

- Is sketch efficient? 전수조사에 비하면 훨씬 연산이 적음
  - Assume that there are N documents in the collection
  - To compute Jaccard of $S(d_i)$ and $S(d_j)$, we have to know $S(d_j) \cap S(d_j)$ which requires $|S(d_i)|$ X $|S(d_j)|$ operations
  - This should be repeat for all the pairs in the collection
  - Once a **sketch vector** $sketch_d$[] is created for each document d in the collection
  - Now using the sketch vectors, It is just 200 iterations of comparison to compute $P(\text{sketch}_{d_i}, \text{sketch}_{d_j})$ for a pair of d_i and d_j in the collection

Near Duplicate Documents

- Now we have an extremely efficient method for estimating a Jaccard coefficient for a pair of documents.
- But we still have to estimate N^2 coefficients where N is the number of web pages.
  - Still slow!!

스케치 비교가 좋은 방법이긴 한데 워낙 문서가 많아서 느림