HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

WEB MINING

LECTURE 01: INFORMATION RETRIEVAL

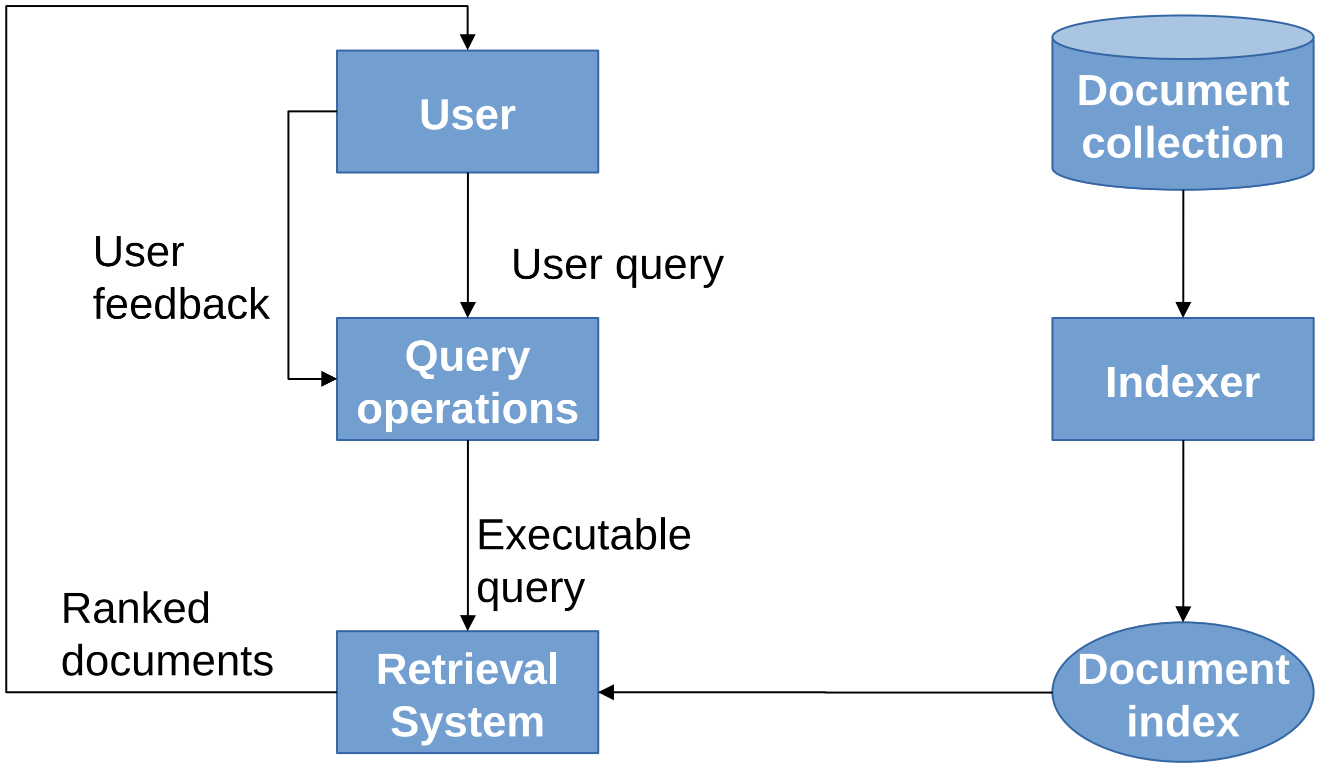
ONE LOVE. ONE FUTURE.

Agenda

1. Basic Concepts of Information Retrieval
2. Information Retrieval Models
3. Relevance Feedback
4. Evaluation Measures
5. Text and Web Page Pre-Processing
6. Inverted Index and Its Compression
7. Latent Semantic Indexing
8. Web Search
9. Meta-Search: Combining Multiple Rankings
10. Web Spamming

1. Basic Concepts of Information Retrieval

- Information retrieval (IR) helping users to find information that matches their information needs
- IR studies the acquisition, organization, storage, retrieval, and distribution of information.
- IR is about document retrieval, emphasizing document as the basic unit.
- User issues a query (**user query**) to the **retrieval system** through the **query operations** module. The retrieval module uses the **document index** to retrieve documents relevant to the query (contain some query terms), compute relevance scores for them, and then rank the retrieved documents according to the scores. The ranked documents are then presented to the user. The **document collection** is also called the **text database**, which is indexed by the **indexer** for efficient retrieval.


User
User feedback
User query
Query operations
Executable query
Ranked documents
Retrieval System
Document collection
Indexer
Document index

- Type of user query
  1. **Keyword query**: a list of keywords (or terms) to find documents contain some or all query terms. Order of keywords is also significan and affect the results. E.g: *information retrieval*
  2. **Boolean query**: consists of terms and Boolean operators AND, OR, and NOT. E.g: *information* OR *retrieval*
  3. **Phrase query**: consists of a sequence of words that makes up a phrase. Each returned document must contain at least one instance of the phrase. E.g: "*information retrieval systems*"
  4. Proximity query: can consists combination of terms and phrases. Returned documents are ranked by distance between terms and phrases.
  5. Full document query: find documents that are similar to the query document
  6. Natural language question (question answering): The most complex case. User express query as natural language questions.

- Query operations can range from very simple to very complex: remove stopwords, transform natural language queries into executable queries, use user feedback to expand and redefine original query.
- Indexer index original raw documents in some data structures to enable efficient retrieval. The result is document index. Invert index is easy to build and very efficient to search.
- Retrieval system computes a relevance score for each indexed document to the query. The documents are ranked and presented to the user. Only documents that contains at least one query term is first found from the index are scored.

2. Information Retrieval Models

- Information retrieval models represent documents and queryies and score the relevance of a document to a user query.
- Documents and queries are represented as bag of words. Sequence and position are ignored.
- Collection of documents *D*
- $V = \{t_1, t_2, \ldots, t_{|V|}\}$ is the set of distinct terms in D, t_i is a term. V is usually called the vocabulary, and $|V|$ is vocabulary size.
- Each term t_i in document $\boldsymbol{d}_j \in D$ has w_{ij} important weight of t_i in $\boldsymbol{d}_j$; $\boldsymbol{d}_j = (w_{1j}, w_{2j}, \ldots, w_{|V|j})$

2.1 Boolean Model

- Document representation: documents and queries are represented as sets of terms.

$$w_{ij} = \begin{cases} 1 \text{ if } t_i \text{ in } \boldsymbol{d}_j \\ 0 \text{ otherwise} \end{cases}$$

- Boolean query: terms are combined with Boolean operators AND, OR and NOT. E.g:
  - *((x AND y) AND (NOT z))* returns document contain both x and y but not z.
  - *(x OR y) returns d*ocument contain at least on term x or y
- Document retrieval: All document satisfied query are returned without ranking

2.2 Vector Space Model

- Document representation:
  - Tf: number of times t_i appears in document $\boldsymbol{d}_j$ (f_{ij}).
  - Cons: does not consider the situation where a term appears in many documents of the collection.

$$tf_{ij} = \frac{f_{ij}}{max(f_{1j}, f_{2j}, \ldots, f_{|V|j})}$$

  - Tf-idf: terms that appear in many documents have low weights

$$idf_i = log\,\frac{N}{df_i} \qquad\qquad w_{ij} = tf_{ij}\ x\ idf_i$$

N: number of documents; df_i number of documents have t_i

- Query (Salton and Buckley)

$$w_{iq} = \left(0.5 + \frac{0.5f_{ij}}{max(f_{1q}, f_{2q}, \ldots, f_{|V|q})}\right) x\ log\,\frac{N}{df_i}$$

- Ranking: Base on similarity between document ***d***$_j$ and query *q*
- Cosine similarity is the most well known.
- Okapi is more efficient on for short query retrieval

  dl$_j$ is document length of ***d***$_j$ (bytes)

  avdl is average document length of the collection

$$cosine(\mathbf{d}_j, \mathbf{q}) = \frac{\langle \mathbf{d}_j \bullet \mathbf{q} \rangle}{\| \mathbf{d}_j \| \times \| \mathbf{q} \|} = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}}.$$

$$sim(\mathbf{d}_j, \mathbf{q}){}^{\backprime} = \langle \mathbf{d}_j \bullet \mathbf{q} \rangle.$$

$$okapi(d_j, q) = \sum_{t_i \in q, d_j} \ln \frac{N - df_i + 0.5}{df_i + 0.5} \times \frac{(k_1 + 1) f_{ij}}{k_1 (1 - b + b \frac{dl_j}{avdl}) + f_{ij}} \times \frac{(k_2 + 1) f_{iq}}{k_2 + f_{iq}},$$

$$pnw(d_j, q) = \sum_{t_i \in q, d_j} \frac{1 + \ln(1 + \ln(f_{ij}))}{(1 - s) + s \frac{dl_j}{avdl}} \times f_{iq} \times \ln \frac{N + 1}{df_i},$$

2.3 Statistical Language Model

- Estimate language model for each document, then ranks document by the likelihood of the query given the language model.
- Query *q* is a sequence of terms $q=q_1q_2...q_m$, document collection $D = \{d_1,d_2,...,d_N\}$. Estimate the posterior probability,which is the probability of query *q* is being generated from a probabilistic model based on a document d_j: $Pr(q|d_j)$

$$Pr(d_j|q) = \frac{Pr(q|d_j)Pr(d_j)}{Pr(q)}$$

- Unigram language models assume each individual term is generated independently by multinomial distribution over words.

$$\Pr(q = q_1 q_2 \ldots q_m \mid d_j) = \prod_{i=1}^{m} \Pr(q_i \mid d_j) = \prod_{i=1}^{|V|} \Pr(t_i \mid d_j)^{f_{iq}}$$

where f_{iq} number of time t_i appears in q and

- $Pr(t_i|d_j)$ is estimated by:

$$\sum_{i=1}^{|V|} \Pr(t_i \mid d_j) = 1 \qquad \Pr(t_i \mid d_j) = \frac{f_{ij}}{|d_j|}$$

where $|d_j|$ number of words d_j

- Smoothing: prevent zero probabilities (for unseen term in document) (λ=1 Laplace smoothing)

$$\Pr_{add}(t_i \mid d_j) = \frac{\lambda + f_{ij}}{\lambda |V| + |d_j|}$$

3. Relevance feedback

- User identifies some relevant and irrelevant documents in the initial list of retrieved documents, and the system then creates an expanded query by extracting some additional terms from the sample relevant and irrelevant documents for a second round of retrieval.
- The system may also produce a classification model using the user-identified relevant and irrelevant documents to classify the documents in the document collection into relevant and irrelevant documents.
- The relevance feedback process may be repeated until the user is satisfied with the retrieved result.

3.1 Rocchio Method

- Query ***q***, set of relevant documents selected by user D_r, set of irrelevant documents D_{ir}, expanded query $\boldsymbol{q}_e$:

$$\mathbf{q}_e = \alpha\mathbf{q} + \frac{\beta}{|D_r|}\sum_{\mathbf{d}_r \in D_r}\mathbf{d}_r - \frac{\gamma}{|D_{ir}|}\sum_{\mathbf{d}_{ir} \in D_{ir}}\mathbf{d}_{ir}$$

- Original query ***q*** is augmented with additional terms from relevant documents D_r
- Irrelevant documents D_{ir} are used to reduve influence of discriminative terms (appear in both or only in D_{ir})

3.2 Machine Learning Method

- Vector $\boldsymbol{c}_i$ is built for each class *i*, which is relevant and irrelevant (negative elements or components of $\boldsymbol{c}_i$ is usually set to 0)

$$\mathbf{c}_i = \frac{\alpha}{|D_i|}\sum_{\mathbf{d}\in D_i}\frac{\mathbf{d}}{\|\mathbf{d}\|} - \frac{\beta}{|D - D_i|}\sum_{\mathbf{d}\in D-D_i}\frac{\mathbf{d}}{\|\mathbf{d}\|}$$

- Each test document $\boldsymbol{d}_t$ is compared with everyvector $\boldsymbol{c}_i$ based on cosine similarity. $\boldsymbol{d}_t$ is assigned to the class with the highest similarity value

```
for each class $i$ do
    build vector $\boldsymbol{c}_i$
for each class $\boldsymbol{d}_t$ do
    $class(\boldsymbol{d}_t) = argmax_i\ cosine(\boldsymbol{d}_t,\boldsymbol{c}_i)$
```

- Learning from Labeled and Unlabeled examples (LU Learning): the number of user-selected relevant and irrelevant documents is small. Unlabeled examples, those are not selected by user, can be utilized to produce a more accurate classifier.
- Learning from Positive and Unlabeled examples (PU Learning): In some cases (like web search) user only select relevant documents (based on title or summary). These documents are called positive examples. The unselected are unlabeled examples (implicit feedback)
- Ranking SVM: use selected document to rank unselected documents
- Language models

Any other?

?

3.3 Pseudo-Relevance Feedback

- Extract some terms (usually most frequent terms) from top-ranked document to original query to form a new query for second round of retrieval
- The process can be repeated until user satisfied with the final results
- Assume that top-ranked documents are relevant and user is not involved in the process

4. Evaluation Measures

- R^q: $<d_1^q, d_2^q, \ldots, d_N^q>$ is rank of document based on relevance score
- D_q is set of relevant document
- Recall at rank position *i*

$$r(i) = \frac{s_i}{|D_q|}$$

where s_i is number of relevant document from d_1^q to d_i^q

- Precision at rank position *i*

$$p(i) = \frac{s_i}{i}$$

- Average precision

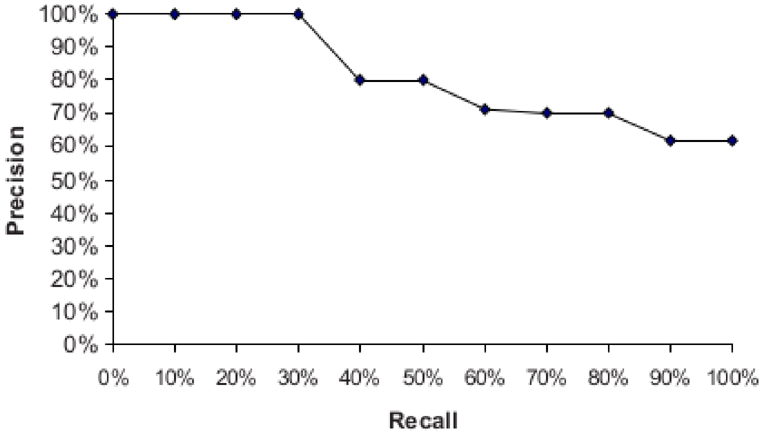
$$p_{avg} = \frac{\sum_{d_i^q \in D_q} p(i)}{|D_q|}.$$

position	relevant	p(i)	r(i)	position	relevant	p(i)	r(i)
1	v	1/1	1/8	11		7/11	7/8
2	v	2/2	2/8	12		7/12	7/8
3	v	3/3	3/8	13	v	8/13	8/8
4		3/4	3/8	14		8/14	8/8
5	v	4/5	4/8	15		8/15	8/8
6		4/6	4/8	16		8/16	8/8
7	v	5/7	5/8	17		8/17	8/8
8		5/8	5/8	18		8/18	8/8
9	v	6/9	6/8	19		8/19	8/8
10	v	7/10	7/8	20		8/20	8/8

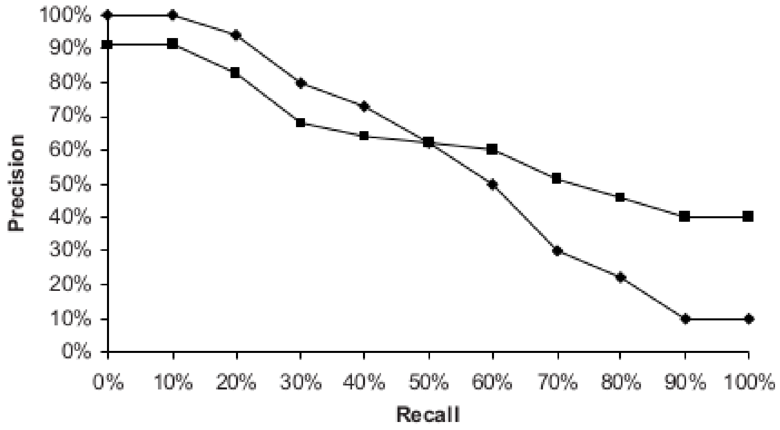
$$p_{avg} = \frac{1/1+2/2+3/3+4/5+5/7+6/9+7/10+8/13}{8} = 0.81$$

i	***p*(*r*$_i$)**	***r*$_i$**
0	1	0
1	1	0.10
2	1	0.20
3	1	0.30
4	0.80	0.40
5	0.80	0.50
6	0.71	0.60
7	0.70	0.70
8	0.70	0.80
9	0.62	0.90
10	0.62	1

$$p(r_i) = \max_{r_i \le r \le r_{10}} p(r) .$$



Precision – Recall Curve



- Evaluation using multiple queries

$$\bar{p}(r_i) = \frac{1}{|Q|}\sum_{j=1}^{|Q|} p_j(r_i),$$

where *Q* is set of queries và $p_j(r_i)$ is precision of query *j* at recall level r_i

- Although in theory precision and recall do not depend on each other, in practice a high recall is almost always achieved at the expense of precision, and a high precision is achieved at the expense of recall.
- In many applications, it can be very hard to determine the set of relevant documents D_q for each query q. (on the Web, D_q is almost impossible)
- Rank precision: *P(5), P(10), P(15), P(20), P(25), P(30)*
- F-score

$$F(i) = \frac{2p(i)r(i)}{p(i)+r(i)}$$

5. Text and Web Page preprocessing

- Text preprocessing:
  - Remove stopwords
  - Stemming
  - Digits, hyphens, punctuations, and cases of letter
- Web Page preprocessing
  - HTML tag
  - Identify main content

5.1 Remove stopwords

- Stopwords frequently occurring and insignificant words that help construct sentences but do not represent any content of the documents
- Articles, prepositions and conjunctions and some pronouns are stopwords

a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or, that, the, these, this, to, was, what, when, where, who, will, with,...

- Such words should be removed before documents are indexed and stored.
- Stopwords in the query are also removed before retrieval is performed

5.2 Stemming

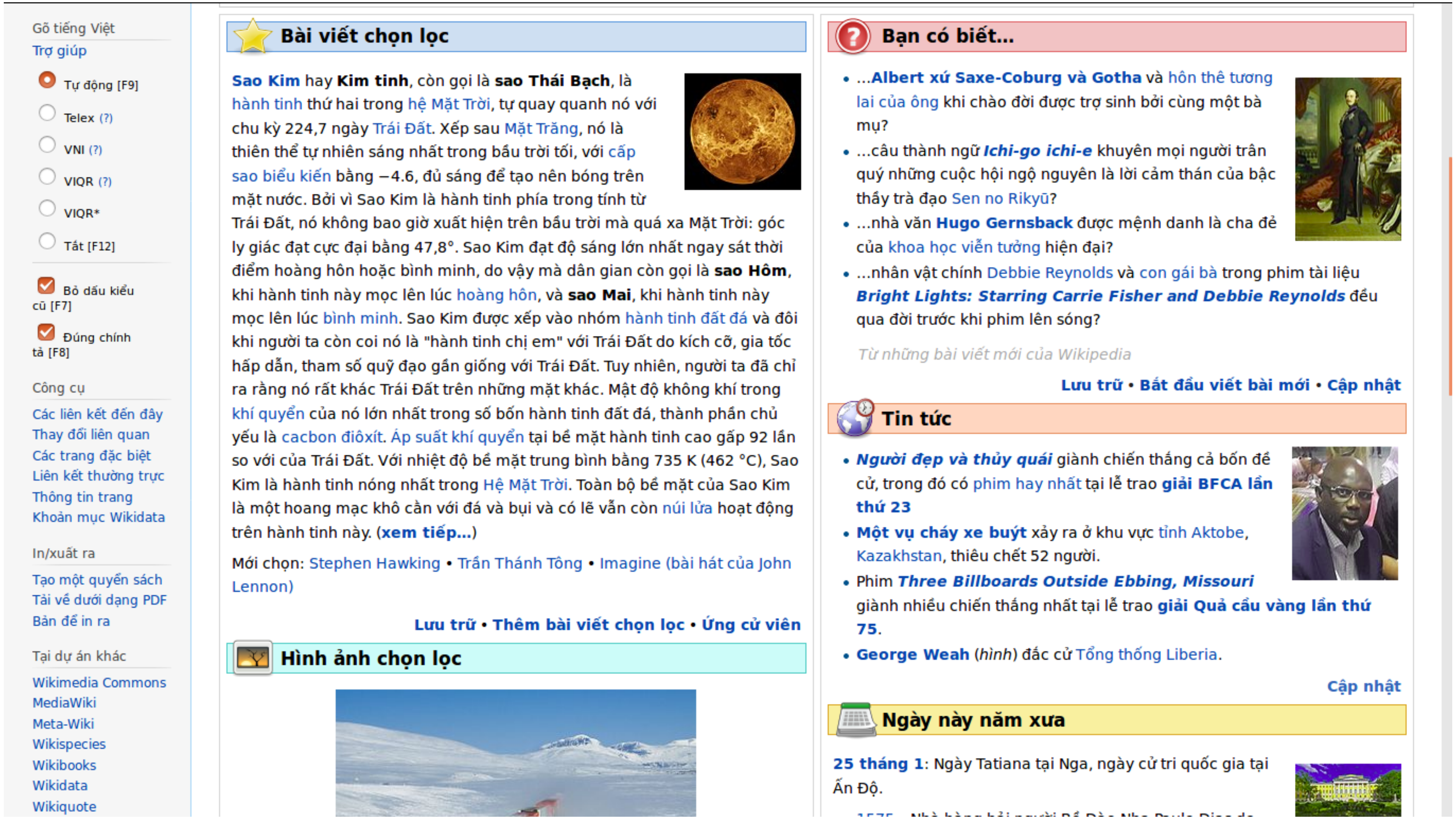
- In many languages, a word has various syntactical forms depending on contexts. For example, in English, nouns have plural forms, verbs have gerund forms (by adding "ing"), and verbs used in the past tense are different from the present tense. These are considered as syntactic variations of the same root form.
- Cause low recall for a retrieval system because a relevant document may contain a variation of a query word but not the exact word itself.
- Stemming: reducing words to their stems or roots. In English, most variants of a word are generated by the introduction of suffixes (rather than prefixes). Thus, stemming in English usually means suffix removal, or stripping. For example, "computer", "computing", and "compute" are reduced to "comput". "walks", "walking" and "walker" are reduced to "walk"
- Martin Porter's stemming algorithm use a set of rules
- Cons of stemming: could return irrelevant document, for example, both "cop" and "cope" are reduced to the stem "cop".

5.3 Other Preprocessing Tasks for Text

- Digits: In traditional IR systems, digits are removed except some specific type, e.g., date, time. However, in search engines, they are usually indexed.
- Hyphens: system usually follow a general rule with exceptions. There are 2 type of removal:
  - Replace hyphens by space, e.g., ‘pre-processing’ → ‘pre processing’
  - Delete hyphens, e.g., ‘state-of-the-art’→ ‘stateoftheart’
- Punctuation marks: similar to hyphens
- Case of letter: Convert into one case

5.4 Web Page Preprocessing

- Identify different text fields: HTML has ,many text fields, e.g., title, metadata, and body. In search engines terms that appear in the title field of a page are regarded as more important than terms that appear in other fields and are assigned higher weights because the title is usually a concise description of the page. In the body text, those emphasized terms (e.g., under header tags <h1>, <h2>, ..., bold tag <b>, etc.) are also given higher weights.
- Identifying anchor text: Anchor text associated with a hyperlink is treated specially in search engines because the anchor text often represents a more accurate description of the information contained in the page pointed to by its link
- Removing HTML tags: The removal of HTML tags can be dealt with similarly to punctuation. One issue needs careful consideration, which affects proximity queries and phrase queries
- Identifying main content blocks: A typical Web page, especially a commercial page, contains a large amount of information that is not part of the main content of the page (banner ads, navigation bars, copyright notices)
  - Partitioning based on visual cues: X and Y coordinates of each element
  - Tree matching: pages are generated by using some fixed templates. The method thus aims to find such hidden templates. Since HTML has a nested structure, it is thus easy to build a tag tree for each page.



Gõ tiếng Việt
Trợ giúp

- ● Tự động [F9]
- ○ Telex (?)
- ○ VNI (?)
- ○ VIQR (?)
- ○ VIQR*
- ○ Tắt [F12]

- ☑ Bỏ dấu kiểu cũ [F7]
- ☑ Đúng chính tả [F8]

Công cụ
- Các liên kết đến đây
- Thay đổi liên quan
- Các trang đặc biệt
- Liên kết thường trực
- Thông tin trang
- Khoản mục Wikidata

In/xuất ra
- Tạo một quyển sách
- Tải về dưới dạng PDF
- Bản để in ra

Tại dự án khác
- Wikimedia Commons
- MediaWiki
- Meta-Wiki
- Wikispecies
- Wikibooks
- Wikidata
- Wikiquote

Bài viết chọn lọc

Sao Kim hay **Kim tinh**, còn gọi là **sao Thái Bạch**, là hành tinh thứ hai trong hệ Mặt Trời, tự quay quanh nó với chu kỳ 224,7 ngày Trái Đất. Xếp sau Mặt Trăng, nó là thiên thể tự nhiên sáng nhất trong bầu trời tối, với cấp sao biểu kiến bằng −4.6, đủ sáng để tạo nên bóng trên mặt nước. Bởi vì Sao Kim là hành tinh phía trong tính từ Trái Đất, nó không bao giờ xuất hiện trên bầu trời mà quá xa Mặt Trời: góc ly giác đạt cực đại bằng 47,8°. Sao Kim đạt độ sáng lớn nhất ngay sát thời điểm hoàng hôn hoặc bình minh, do vậy mà dân gian còn gọi là **sao Hôm**, khi hành tinh này mọc lên lúc hoàng hôn, và **sao Mai**, khi hành tinh này mọc lên lúc bình minh. Sao Kim được xếp vào nhóm hành tinh đất đá và đôi khi người ta còn coi nó là "hành tinh chị em" với Trái Đất do kích cỡ, gia tốc hấp dẫn, tham số quỹ đạo gần giống với Trái Đất. Tuy nhiên, người ta đã chỉ ra rằng nó rất khác Trái Đất trên những mặt khác. Mật độ không khí trong khí quyển của nó lớn nhất trong số bốn hành tinh đất đá, thành phần chủ yếu là cacbon điôxít. Áp suất khí quyển tại bề mặt hành tinh cao gấp 92 lần so với của Trái Đất. Với nhiệt độ bề mặt trung bình bằng 735 K (462 °C), Sao Kim là hành tinh nóng nhất trong Hệ Mặt Trời. Toàn bộ bề mặt của Sao Kim là một hoang mạc khô cằn với đá và bụi và có lẽ vẫn còn núi lửa hoạt động trên hành tinh này. (**xem tiếp...**)

Mới chọn: Stephen Hawking • Trần Thánh Tông • Imagine (bài hát của John Lennon)

Lưu trữ • Thêm bài viết chọn lọc • Ứng cử viên

Hình ảnh chọn lọc

Bạn có biết...

- ...**Albert xứ Saxe-Coburg và Gotha** và hôn thê tương lai của ông khi chào đời được trợ sinh bởi cùng một bà mụ?
- ...câu thành ngữ ***Ichi-go ichi-e*** khuyên mọi người trân quý những cuộc hội ngộ nguyên là lời cảm thán của bậc thầy trà đạo Sen no Rikyū?
- ...nhà văn **Hugo Gernsback** được mệnh danh là cha đẻ của khoa học viễn tưởng hiện đại?
- ...nhân vật chính Debbie Reynolds và con gái bà trong phim tài liệu ***Bright Lights: Starring Carrie Fisher and Debbie Reynolds*** đều qua đời trước khi phim lên sóng?

Từ những bài viết mới của Wikipedia

Lưu trữ • Bắt đầu viết bài mới • Cập nhật

Tin tức

- ***Người đẹp và thủy quái*** giành chiến thắng cả bốn đề cử, trong đó có phim hay nhất tại lễ trao **giải BFCA lần thứ 23**
- **Một vụ cháy xe buýt** xảy ra ở khu vực tỉnh Aktobe, Kazakhstan, thiêu chết 52 người.
- Phim ***Three Billboards Outside Ebbing, Missouri*** giành nhiều chiến thắng nhất tại lễ trao **giải Quả cầu vàng lần thứ 75**.
- **George Weah** (*hình*) đắc cử Tổng thống Liberia.

Cập nhật

Ngày này năm xưa

25 tháng 1: Ngày Tatiana tại Nga, ngày cử tri quốc gia tại Ấn Độ.

- 1575 – Nhà hàng hải người Bồ Đào Nha Paulo Dias de

5.5 Duplicate detection

- Duplicate are often used to improve efficiency of browsing and file downloading worldwide to tackle network problems. Some duplicate pages are the results of plagiarism. Detecting such pages and sites can reduce the index size and improve search results
- Copying a page is usually called duplication or replication, and copying an entire site is called mirroring
- Hashing or checksum can detect exact duplicate.
- To detect partial duplicate, use technique based on n-gram (or shingles) and Jaccard similarity (e.g., the sentence, “John went to school with his brother,” can be represented with five 3-gram phrases “John went to”, “went to school”, “to school with”, “school with his”, and “with his brother”)

$$sim(d_1,d_2) = \frac{|S_n(d_1) \cap S_n(d_2)|}{|S_n(d_1) \cup S_n(d_2)|}$$

6. Inverted Index

- Baseline: for each query, scan every document in database to find the terms in the query → not efficient
- Inverted index boost speed of searching and database building

6.1 Inverted Index

- Inverted index of a set of documents is a data structure that attacks each term with a list of documents contain that term. Searching time of term in document is constant to number of documents
- Set of documents $D=\{d_1,d_2,\ldots,d_N\}$ each document has a unique *id*; inverted index of *D* contains:
  - Vocabulary *V* contain all distinct term in all documents
  - Each term t_i is attack with a list of postings, each posting stores id of a document contain t_i and some information:

  $<id_j, f_{ij}, [o_1,o_2,\ldots,o_{|fij|}]>$
    - id_j is unique id of d_j
    - f_{ij} *frequency of* t_i in d_j
    - o_k is offset of t_i in d_j
  - Postings list is sort by id and and offset

id_1: Web mining is useful.
1 2 3 4
id_2: Usage mining applications.
1 2 3
id_3: Web structure mining studies the Web hyperlink structure.
1 2 3 4 5 6 7 8

V={web,mining,useful,applications,usage,structure,studies,hyperlink}

applications:	*id_2*	*applications:*	*<id_2,1,[3]>*
hyperlink:	*id_3*	*hyperlink:*	*<id_3,1,[7]>*
mining:	*id_1, id_2, id_3*	*mining:*	*<id_1,1,[2]>, < id_2,1,[2]>, <id_3,1[2]>*
structure:	*id_3*	*structure:*	*<id_3,2,[2,8]>*
studies:	*id_3*	*studies:*	*<id_3,1,[4]>*
usage:	*id_2*	*usage:*	*<id_2,1,[1]>*
useful:	*id_1*	*useful:*	*<id_1,1,[4]>*
web:	*id_1, id_3*	*web:*	*<id_1,1,[1]>, <id_3,2,[1,6]>*

a) b)

6.2 Search using Inverted Index

- Given the query terms, searching for relevant documents in the inverted index consists of three main steps:

1. Vocabulary search: find each query term in vocabulary, which gives the inverted list of each item. To speed up, vocabulary is usually strored using specicial data structure like hash, tries, B-tree. Lexicographical order can be used due to its space efficiency. Binary search can be applied. The complexity is *O(log|V|)* where *|V|* size of vocabulary
2. Results merging: Traverse all lists in synchronization to check whether each document contains all query terms. Start with the shortest list, for each document in it, binary search on other lists to find the same document. The whole inverted index can not fit in memory, so part of it is cache in memory for efficient.
3. Rank score computation: compute rank score for each document base on a relevance function (e.g. cosine or okapi), my consider term and phrase proximity information.

- Example, query “web mining”:
  - Step 1: found 2 inverted index

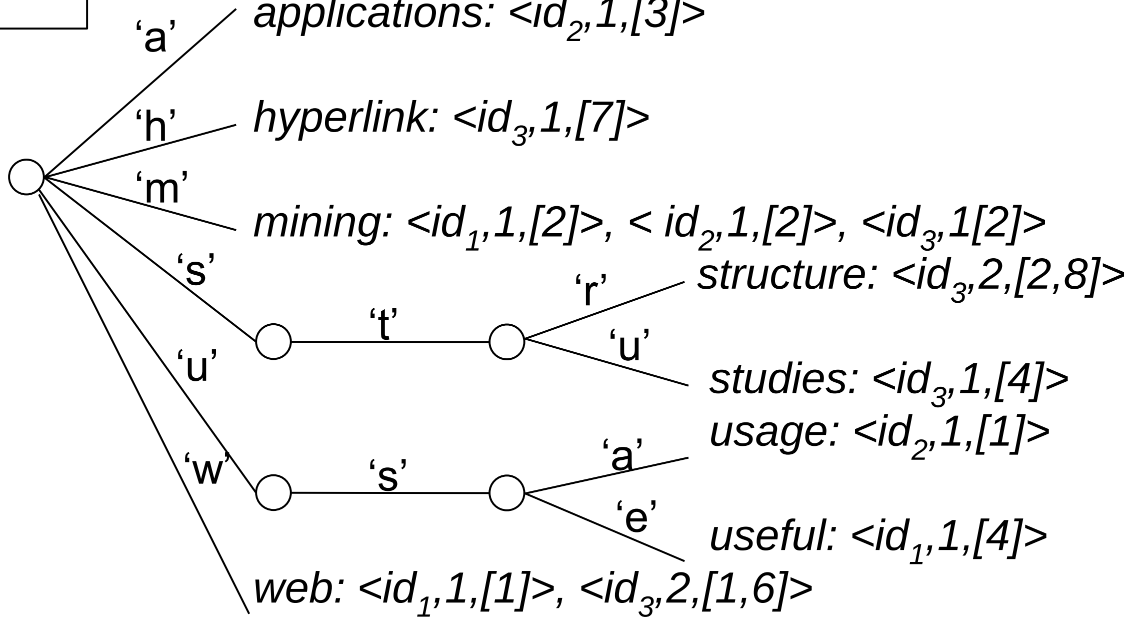
    mining: <id_1,1,[2]>, < id_2,1,[2]>, <id_3,1[2]>
    web: <id_1,1,[1]>, <id_3,2,[1,6]>

  - Step 2: found id_1 and id_3 contains both 2 term
  - Step 3: id_1 has higher relevance score than id_3 due to “web” and “mining” come together in id_1 and has the same order as in q

6.3 Index construction

- Trie, a data structure, has complexity of O(T), where T is size of vocabulary

```
for each doc do
    for each term in doc do
        if term in trie update inverted list
        else add term into trie
```



applications: <id$_2$,1,[3]>

hyperlink: <id$_3$,1,[7]>

mining: <id$_1$,1,[2]>, < id$_2$,1,[2]>, <id$_3$,1[2]>

structure: <id$_3$,2,[2,8]>

studies: <id$_3$,1,[4]>

usage: <id$_2$,1,[1]>

useful: <id$_1$,1,[4]>

web: <id$_1$,1,[1]>, <id$_3$,2,[1,6]>

- Problem: not enough memory to save all index → solution:
  - Partial build index in memory and save to disks, I_1, I_2,…,I_k
  - Pair-wise merge until all index are merge into one, I_1, I_2 into $I_{1\text{-}2}$, I_3, I_4 thành $I_{3\text{-}4}$
- Problem: pages are constantly added, modified or deleted → Solution:
  - Build index D_+ for added pages and D_- for deleted pages
  - Search on all of the indexes, final results: $D \cup D_+ \setminus D_-$

6.4 Index compression

- Index compression is for memory size reduction without information lost
- Information is represented with positive integers → integer compression technique
- In variable-bit (also called bitwise) scheme, an integer is represented with an integral number of bits. (unary coding, Elias gamma coding, delta coding, Golumb coding)
- Variable-byte use 7 bits to represent value and the very right bit as delimiter (0 if it is the last byte, otherwise 1)
- Since ID are sorted in increasing order, we can store smallest ID and the difference between any two adjacent document IDs. The gap is smaller than ID value and thus requires fewer bits

<4,10,300,305> → <4,6,290,5>

- **Unary coding:** represent x by x-1 bit 0 and a bti 1, e.g. 5 = 00001
- **Elias gamma coding:** Represent 1+[$\log_2 x$] in unary follow by binary representation of x without most significant bit. Efficient for small integers.

E.g.: 9: 0001001 since 1+[$\log_2 9$]=4 and 9=$1001_{(2)}$

- Decode:
  - **1.** Read K 0 until counter 1;
  - **2.** Consider the 1 reached is first digit of the integer (2^K), read next K bits.

To decode 0001001, we have K=3 bit 0, thus binary representation is $1001_{(2)}$=9

- **Elias delta coding:** Represent 1+[$\log_2 x$] in gamma follow by binary representation of x without most significant bit. Efficient for large integers

E.g.: 9=00100001 since 1+[$\log_2 9$]=4 has gamma coded of 00100

- Decode:
  - **1.** Read L 0 until reach first 1
  - **2.** Consider reached *1* is first bit of the integer (2^L), read next *L* bit to get integer M.
  - **3.** Put 1 to first place of final output (2^M) and read next M-1 bit.

e.g.: decode 00100001, **1.** L=2; **2.** M=4; **3.** $1001_{(2)}$=9

- **Golomb coding**:

1. First part is unary representation of q+1 where $q=\lfloor(x/b)\rfloor$

2. Next part is binary representation of the remain $r=x-qb$. consider $i = \lfloor\log_2 b\rfloor$, $d=2^{i+1}-b$, first d remainders are represented using *i* bit; the others represented using $\lceil\log_2 b\rceil$ (i+1) bit (fixed prefix coding)

e.g.: $x=9$, $b=3$; $q=\lfloor(9/3)\rfloor=3$, $i=\lfloor\log_2 3\rfloor=1$,

$d=2^{1+1}-3=1$, $r=9-3x3=0$ → 9: 00010

- select b:

$$b \approx 0.69\left[\frac{N}{n_t}\right]$$

where *N* is number of documents, n_t number of documents contain *t*

- **Decode**:

1. decode q

2. i = $\lfloor \log_2 b \rfloor$, d = 2^{i+1}-b

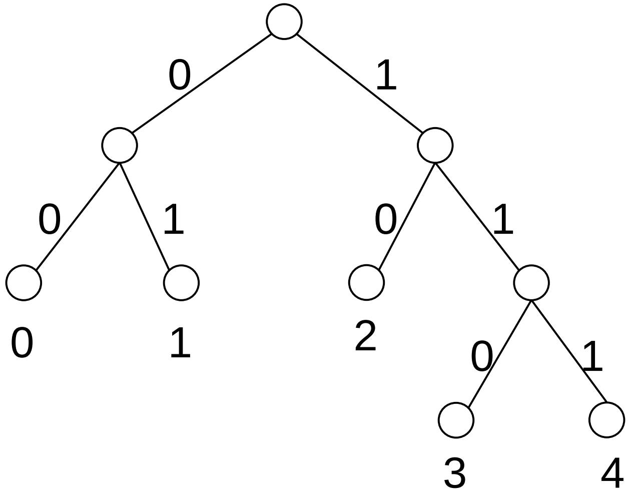
3. Retrieve the next i bits

and assign it to r.

4. If r ≥ d

Retrieve one more bit and append
to r at the end
r = r - d

5. x = qb + r



Cây mã hóa với b=5

- E.g: decode 11111 where b = 10

1. q = 0 **2.** i = $\lfloor \log_2 10 \rfloor$ = 3, d = $2^{3+1 -}$ 10 = 6 **3.** r = $111_{(2)}$ = 7 **4.** 7 > 6→ r = $1111_{(2)}$ = 15; r = 15 – 6 = 9 **5.** x = 0x10+9=9

- **Variavle-byte coding:** 7 bits to represent, the very right bit = 0 if last byte, otherwise 1. efficient for small integers. E.g. 135: 00000011 00001110
- **Decode:**

1. Read all bytes until a byte with the zero last bit is seen

2. Remove the least significant bit from each byte read so far and concatenate the remaining bits.

e.g.: 00000011 00001110 decode to $00000010000111_{(2)}$ = 135

- Comment:
  - Golumb coding has better compression ratio and faster retrieval than non-parameterized Elias coding
  - Variable-byte are often faster tham variable-bit integers despite having higher storage costs
  - A suitable compression technique can allow retrieval to be up to twice as fast than without compression, while the space requirement averages 20% – 25% of the cost of storing uncompressed integers

7. Latent Semantic Indexing

- If a user query uses different words from the words used in a document, the document will not be retrieved although it may be relevant because the document uses some synonyms of the words in the user query. For example, “picture”, “image” and “photo” are synonyms in the context of digital cameras.
- Lantent Semantic Indexing (LSI) aims to deal with this problem through the identification of statistical associations of terms. It is assumed that there is some underlying latent semantic structure in the data that is partially obscured by the randomness of word choice.
- LSI use Singular Value Decomposition (SVD) to estimate this latent structure, and to remove the “noise”
- Transformed terms and documents in the “concept” space are then used in retrieval, not the original terms or documents.
- The query is also transformed into the “concept” space before retrieval

7.1 SVD

- SVD fector a matrix m x n A into product of three matrix

$$\mathbf{A=U\Sigma V^T}$$

where

U is a m x r matrix and its columns called left singular vectors, are eigenvectors associated with the r non-zero eigenvalues of $\mathbf{A}^T$;

$\mathbf{U^TU=I}$

V is an n x r matrix and its columns, called right singular vectors, are eigenvectors associated with the r non-zero eigenvalues of $\mathbf{A^TA}$;

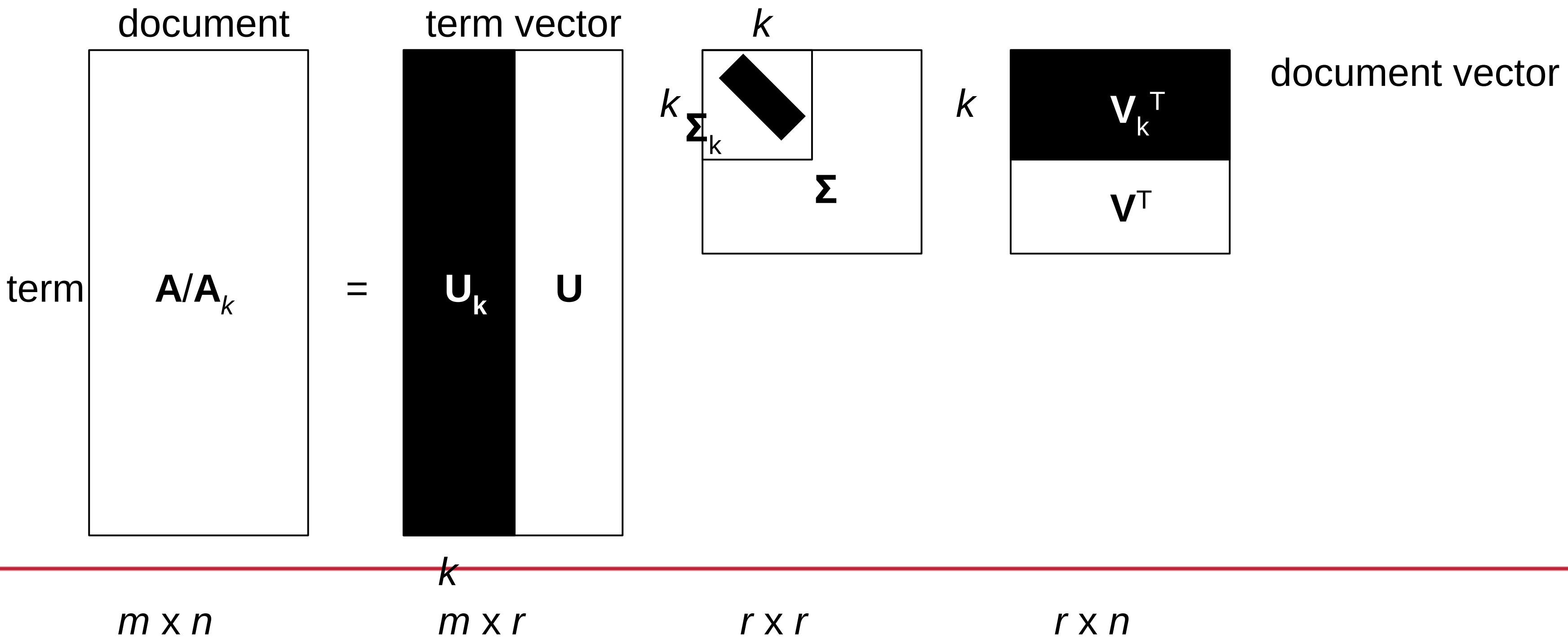
$\mathbf{V^TV=I}$

Σ is a r x r diagonal matrix, $\mathbf{\Sigma}=\text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, $\sigma_i > 0$. $\sigma_1, \sigma_2, \ldots, \sigma_r$ (called singular values) are the non-negative square roots of the r (non-zero) eigenvalues of $\mathbf{AA}^T$. $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$

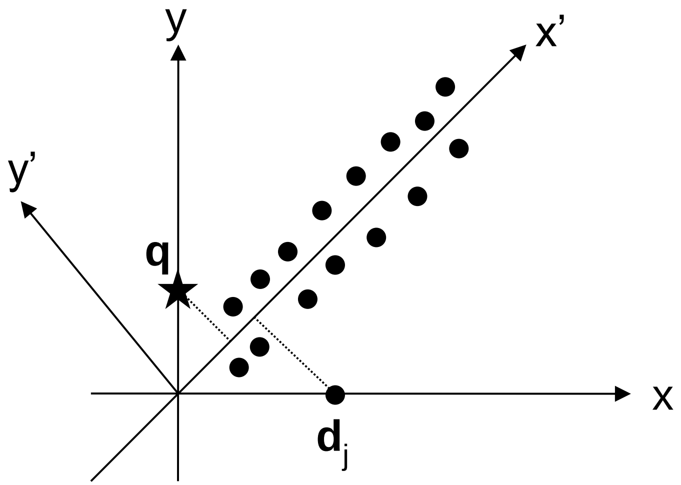
- *m*: number of terms; *n*: number of documents; *r* rank of A **A**, $r \leq \min(m, n)$

- Use the k-largest singular triplets to approximate the original term-document matrix **A** (**A**$_k$), The new space is called the k-concept space, remove “noise”

$$\mathbf{A}_k=\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$$



- SVD rotates the axes of m-dimensional space of A such that the first axis runs along the largest variation (variance) among the documents, the second axis runs along the second largest variation (variance) and so on
- The original x-y space is mapped to the x'-y' space generated by SVD. x and y are clearly correlated. y' direction is insignificant, and may represent some "noise", so we can remove it



Giả thiết của LSI

7.2 Query and Retrieval

- Query ***q*** in original space:

$$\mathbf{A_k = U_k \Sigma_k V_k^T}$$

- Since ***U****$_k$* are unit orthogonal vectors, ***U****$_k$*^{T}***U****$_k$* = ***I***

$$\mathbf{q} = \mathbf{U_k \Sigma_k q_k}^T$$

- Finally,

$$\mathbf{U_k}^T\mathbf{q} = \mathbf{\Sigma_k q_k}^T$$

- Compare ***q****$_k$* to documents in k-dimension space using similarity function

$$\mathbf{\Sigma_k}^{-1}\mathbf{U_k}^T\mathbf{q} = \mathbf{q_k}^T$$

$$\mathbf{q_k = \Sigma_k}^{-1}\mathbf{U_k q}^T$$

7.3 Example

human-computer interaction	c_1:	Human machine interface for Lab ABC computer applications
	c_2:	A survey of user opinion of computer system response time
	c_3:	The EPS user interface management system
	c_4:	System and human system engineering testing of EPS
	c_5:	Relation of user-perceived response time to error measurement
graphs	m_1:	The generation of random, binary, unordered trees
	m_2:	The intersection graph of paths in trees
	m_3:	Graph minors IV: Widths of trees and well-quasi-ordering
	m_4:	Graph minors: A survey

$$
\mathbf{A}= \begin{array}{ccccccccc|l}
c_1 & c_2 & c_3 & c_4 & c_5 & m_1 & m_2 & m_3 & m_4 & \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & human \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & interface \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & computer \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & user \\
0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & system \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & response \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & time \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & EPS \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & survey \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & trees \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & graph \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & minors
\end{array}
$$

$$\mathbf{U} = \begin{pmatrix} 0.22 & -0.11 & 0.29 & -0.41 & -0.11 & -0.34 & 0.52 & -0.06 & -0.41 \\ 0.20 & -0.07 & 0.14 & -0.55 & 0.28 & 0.50 & -0.07 & -0.01 & -0.11 \\ 0.24 & 0.04 & -0.16 & -0.59 & -0.11 & -0.25 & -0.30 & 0.06 & 0.49 \\ 0.40 & 0.06 & -0.34 & 0.10 & 0.33 & 0.38 & 0.00 & 0.00 & 0.01 \\ 0.64 & -0.17 & 0.36 & 0.33 & -0.16 & -0.21 & -0.17 & 0.03 & 0.27 \\ 0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\ 0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\ 0.30 & -0.14 & 0.33 & 0.19 & 0.11 & 0.27 & 0.03 & -0.02 & -0.17 \\ 0.21 & 0.27 & -0.18 & -0.03 & -0.54 & 0.08 & -0.47 & -0.04 & -0.58 \\ 0.01 & 0.49 & 0.23 & 0.03 & 0.59 & -0.39 & -0.29 & 0.25 & -0.23 \\ 0.04 & 0.62 & 0.22 & 0.00 & -0.07 & 0.11 & 0.16 & -0.68 & 0.23 \\ 0.03 & 0.45 & 0.14 & -0.01 & -0.30 & 0.28 & 0.34 & 0.68 & 0.18 \end{pmatrix}$$

$$\mathbf{\Sigma} = \begin{pmatrix} 3.34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.54 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.35 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.64 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.31 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.56 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.36 \end{pmatrix}$$

$$\mathbf{V} = \begin{pmatrix} 0.20 & -0.06 & 0.11 & -0.95 & 0.05 & -0.08 & 0.18 & -0.01 & -0.06 \\ 0.61 & 0.17 & -0.50 & -0.03 & -0.21 & -0.26 & -0.43 & 0.05 & 0.24 \\ 0.46 & -0.13 & 0.21 & 0.04 & 0.38 & 0.72 & -0.24 & 0.01 & 0.02 \\ 0.54 & -0.23 & 0.57 & 0.27 & -0.21 & -0.37 & 0.26 & -0.02 & -0.08 \\ 0.28 & 0.11 & -0.51 & 0.15 & 0.33 & 0.03 & 0.67 & -0.06 & -0.26 \\ 0.00 & 0.19 & 0.10 & 0.02 & 0.39 & -0.30 & -0.34 & 0.45 & -0.62 \\ 0.01 & 0.44 & 0.19 & 0.02 & 0.35 & -0.21 & -0.15 & -0.76 & 0.02 \\ 0.02 & 0.62 & 0.25 & 0.01 & 0.15 & 0.00 & 0.25 & 0.45 & 0.52 \\ 0.08 & 0.53 & 0.08 & -0.03 & -0.60 & 0.36 & 0.04 & -0.07 & -0.45 \end{pmatrix}$$

$$\mathbf{A}_k = \begin{array}{ccc} \mathbf{U}_k & \mathbf{\Sigma}_k & \mathbf{V}_k \\ \begin{pmatrix} 0.22 & -0.11 \\ 0.20 & -0.07 \\ 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & -0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{pmatrix} & \begin{pmatrix} 3.34 & 0 \\ 0 & 2.54 \end{pmatrix} & \begin{pmatrix} 0.20 & 0.61 & 0.46 & 0.54 & 0.28 & 0.00 & 0.02 & 0.02 & 0.08 \\ -0.06 & 0.17 & -0.13 & -0.23 & 0.11 & 0.19 & 0.44 & 0.62 & 0.53 \end{pmatrix} \end{array}$$

Query "*user interface*"

$$\mathbf{q}_k = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T \begin{pmatrix} 0.22 & -0.11 \\ 0.20 & -0.07 \\ 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & -0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{pmatrix} \begin{bmatrix} 3.34 & 0 \\ 0 & 2.54 \end{bmatrix}^{-1} = (0.179 \; -0.004)$$

<u>cosine($\mathbf{q}$,$\mathbf{d}_i$):</u>

c_1:	0.964
c_2:	0.957
c_3:	0.968
c_4:	0.928
c_5:	0.922
m_1:	0.022
m_2:	0.023
m_3:	0.010
m_4:	0.127

⬇

Ranking: $(c_3, c_1, c_2, c_4, c_5, m_4, m_3, m_2, m_1)$

7.3 Discussion

- Pros: LSI give better results than traditional IR
- Cons:
  - Complexity of O(m^2n), not suitable to Web search
  - Concept space is not interpretable as its description consists of all numbers with little semantic meaning.
  - the value of k needs to be determined based on the specific document collection via trial and error, which is a very time consuming
- Association rules may be able to approximate the results of LSI

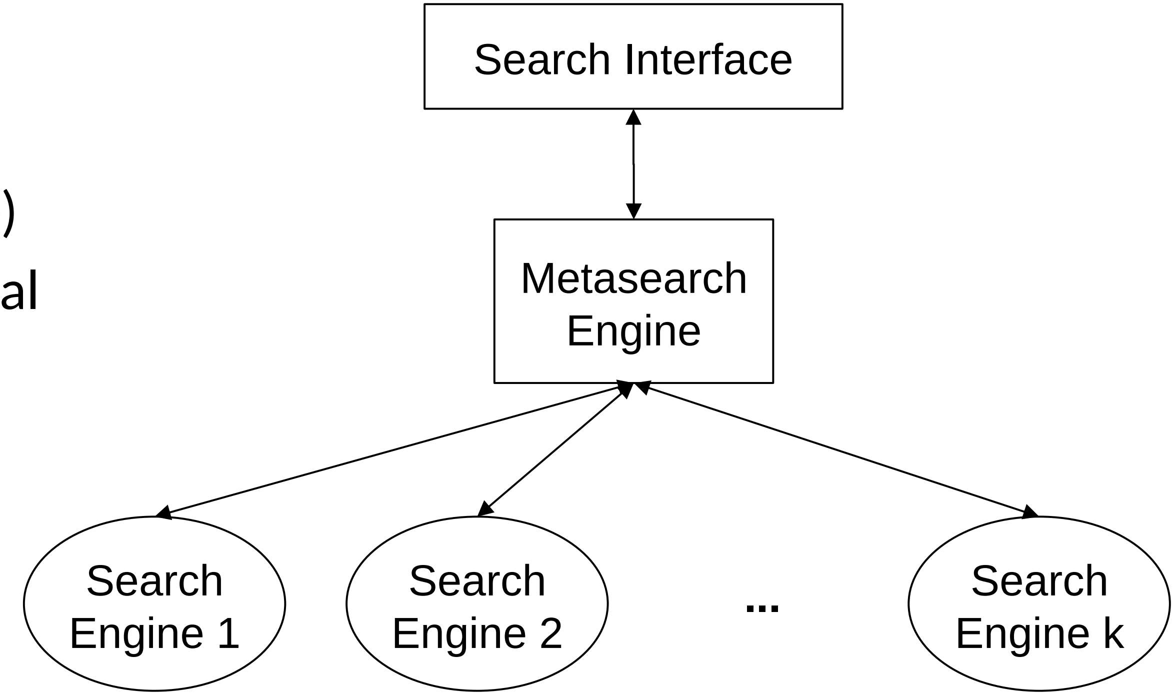
8. Web Search

- Search Engine (Google, Bing, Baidu, Yandex)
- Process: crawl pages, parse, index, store, query, retrieve
- Parsing: Read HTML using lexical analyzer generator such as YACC and Flex
- Indexing: small inverted index may be constructed based on the terms appeared in title and summary. Full index built on entire page.
- Searching and Ranking:
  - Using only relevance score of document may return low quality results.
  - Number of pages are very large, best results need to be on top

- Hyperlink can be used to assess the quality of each page to some extent. A link from page x to page y is an implicit conveyance of authority of page x to page y. That is, the author of page x believes that page y contains quality or authoritative information. It makes use of the link structure of Web pages to compute a quality or reputation score for each page
- Content factors:
  - Occurrence type: title, hyperlink, URL, body, tag
  - Count: number of occurrence of a term of each type
  - Position: Query terms that are near to each other are better than those that are far apart. Furthermore, query terms appearing in the page in the same sequence as they are in the query are also better.

9. Meta-Search and Combining Multiple Rankings

- Meta-search engine combine results of multiple search engine
- increases the search coverage of the Web
- improve the search effectiveness
- Remove duplicate results (compare URL, title)
- Combine the ranked results from individual search engines to produce a single ranking



9.1 Combine using Similarity Score

- Set of returned document $D = \{d_1, d_2, \ldots, d_N\}$, search engine i return s_{ij} is similarity score of query to document d_j

CombMIN(d_j) = $\min(s_{1j}, s_{2j}, \ldots, s_{kj})$

CombMAX(d_j) = $\underset{k}{\max}(s_{1j}, s_{2j}, \ldots, s_{kj})$

CombSUM(d_j) = $\Sigma_{i=1} s_{ij}$

CombANZ(d_j) = $$\frac{\text{CombSUM}(d_j)}{r_j}$$

where r_j number of non-zero similarity score

CombMNZ(d_j) = CombSUM(d_j) x r_j

9.2 Combination using Rank Positions

- *Borda Ranking*: For each search engine, the first document has n point (n is number of documents), second document has n-1 point,…, the rest point is divided evenly to all the document that are not returned. Fina point of a document sum of points from all search engine
- *Condorcet Ranking*: document A is ranked higher than document B if A is return by morn search engine than B. In the same result from a search engine, ranked documents win unranked documents. All ranked document tie with one another. Final result is document(s) that wins the pair-wise comparison.
- *Reciprocal Ranking*: For each search engine, top document got 1 point, the second got ½ point, the third got 1/3 point, …, unranked ones got zero. Final point is sum of all points for all search engines

Borda:
Score(a) = 4 + 3 + 2 + 1 + 1.5 = 11.5
Score(b) = 3 + 4 + 3 + 3 + 3 = 16
Score(c) = 2 + 1 + 4 + 4 + 4 = 15
Score(d) = 1 + 2 + 1 + 2 + 1.5 = 7.5
Rank: *b, c, a, d*

Reciprocal Ranking:
Score(a) = 1 + 1/2 + 1/3 = 1.83
Score(b) = 1/2 + 1 + 1/2 + 1/2 + 1/2 = 3
Score(c) = 1/3 + 1/4 + 1 + 1 + 1 = 3.55
Score(d) = 1/4 + 1/3 + 1/4 + 1/3= 1.17
Rank: c, b, a, d.

Results:

Search engine 1:	a , b , c , d
Search engine 2:	b , a , d , c
Search engine 3:	c , b , a , d
Search engine 4:	c , b , d
Search engine 5:	c , b

Condorcet:

	a	*b*	*c*	*d*
a	-	1:4:0	2:3:0	3:1:1
b	4:1:0	-	2:3:0	5:0:0
c	3:2:0	3:2:0	-	4:1:0
d	1:3:1	0:5:0	1:4:0	-

	Win	Lose	Tie
a	1	2	0
b	2	1	0
c	3	0	0
d	0	3	0

Thứ hạng: *c, b, a, d*

10. Web spamming

- Increasing content and influence on the web will bring popularity and financial benefits to organizations and individuals
- When users search for information through a search engine query, high-ranking websites benefit businesses, organizations, and individuals that publish that website.
- For a query, let's say web pages with more informational value rank higher. However, search engines do not "understand" the information and make ratings based on syntactic and other surface features to evaluate the information. Spammers take advantage of understanding the search engine's ranking mechanism to build website content so that although it does not carry high informational value, it still ranks high (SEO - Search Engine Optimization).
- Web spam affects users making it difficult for them to find real information and reduces the user experience; waste search engine crawling and storage resources and degrade search engine rankings

10.1 Content spamming

- Content spamming builds website content (titles, meta tags, body, hyperlink phrases, URLs) related to certain queries
- Two spamming technique:
  - Repeating some important terms: This method increases the TF scores of the repeated terms in a document and thus increases the relevance of the document to these terms. Add them randomly in an unrelated (or related) sentence
  - Dumping of many unrelated terms: This method is used to make the page relevant to a large number of queries. Simply copy sentences from related pages on the Web and glue them together. Add some frequently searched terms on the Web and put them in the target pages so that when users search for the frequently search terms, the target pages become relevant.

    For example, to advertise cruise liners or cruise holiday packages, spammers put “Tom Cruise” in their advertising pages

10.2 Link spamming

- Out-link spamming: add out-links in one's pages pointing to some authoritative pages to boost the hub cores of one's pages. A page is a hub page if it points to many authoritative pages. (e.g. Yahoo!)
- In-link spamming:

  1. Create honey pot: create a set of pages that contains useful information. The honey pots attract people pointing to them because of useful information, and then have high reputation scores. Honey pots contain (hidden) links to target spam pages that the spammers want to promote.
  2. Adding links to Web directories
  3. Posting links to the user-generated content (reviews, forum discussions, blogs, etc)
  4. Participating in link exchange: In this case, many spammers form a group and set up a link exchange scheme so that their sites point to each other in order to promote the pages of all the sites.
  5. Creating own spam farm: In this case, the spammer needs to control a large number of sites. Then, any link structure can be created to boost the ranking of target spam pages.

10.3 Hiding technique

- Hide the spamming sentences, terms and links from user
- Content hiding: Spam items are made invisible. Make the spam terms the same color as the background color.

<body background = white>

<font color = white> spam items</font>

<a href = target.html"><img src="blank.gif"> </a>

- Cloaking: Spam Web servers return a HTML document to the user and a different document to a Web crawler. Identify Web crawlers in one of the two ways
  - Maintains a list of IP addresses of search engines, identify crawlers by matching IP address
  - Use User-agent field in HTTP request mesage

  User–Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

- Redirection: redirecting the browser to another URL as soon as the page is loaded. The spammed page is given to the search engine for indexing.

10.4 Combating Spam

- Identify crawler as a regular Web browser
- Give terms in such anchor texts higher weights
- Link analysis method: PageRank, authority/hub point
- Classification method. some example features used in detecting content spam
  - Number of words in the page: A spam page tends to contain more words than a non-spam page so as to cover a large number of popular words.
  - Average word length: The mean word length for English prose is about 5 letters. Average word length of synthetic content is often different.
  - Number of words in the page title: Since search engines usually give extra weights to terms appearing in page titles, spammers often put many keywords in the titles of the spam pages.
  - Fraction of visible content: Spam pages often hide spam terms by making them invisible to the user.
- Partition each Web page into different blocks. Each block is given an importance level automatically. The spam links are usually placed in unimportant blocks of the page. links in less important blocks are given lower transition probabilities to be used in the PageRank computation

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !