Web Mining

- Web mining aims to discover useful information or knowledge from the Web hyperlink structure, page content, and usage data.
- Categories
  - Web structure mining
  - Web content mining
  - Web usage mining

Chapter 6: Information Retrieval and Web Search

An introduction

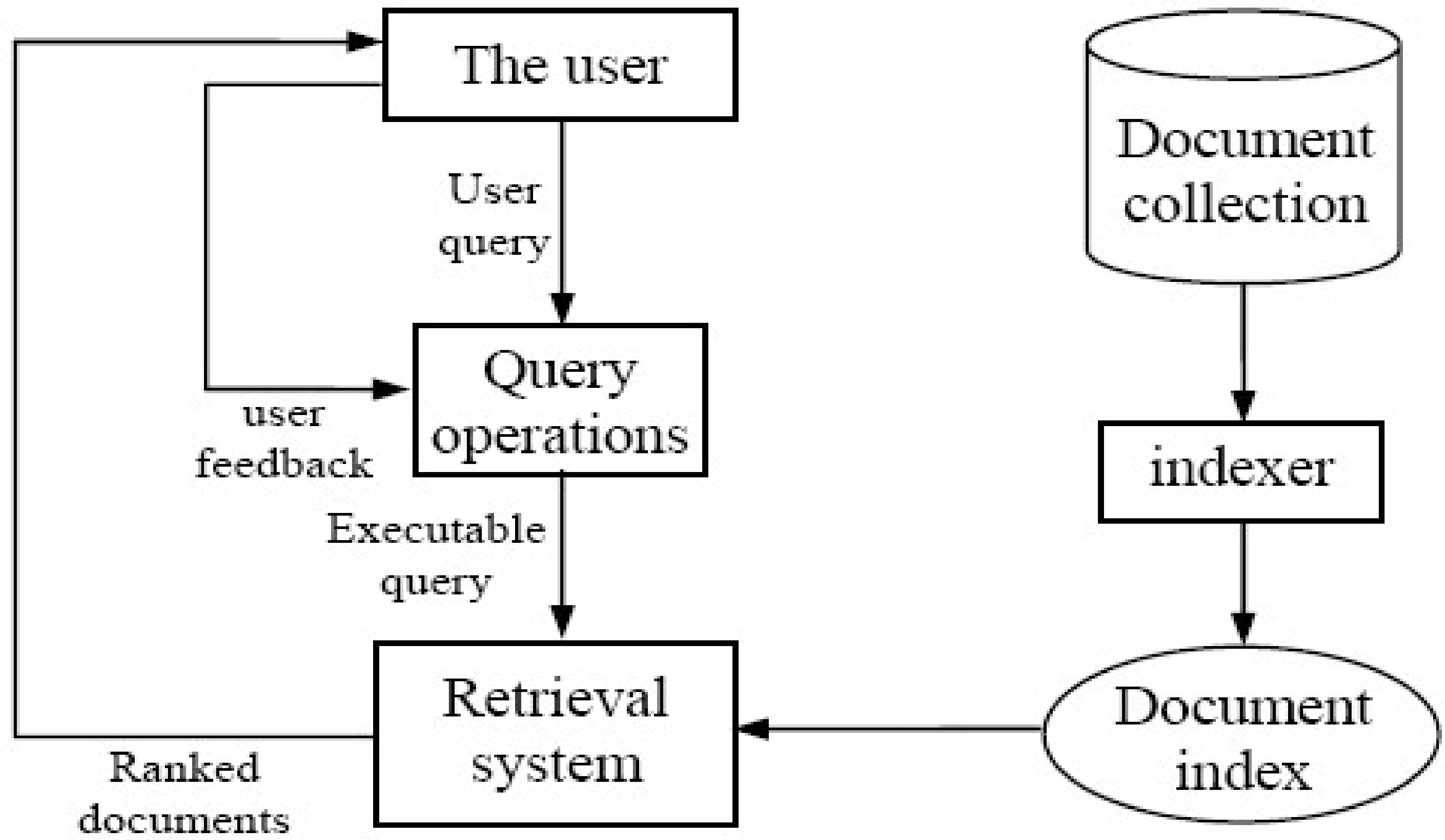
Introduction

- Text mining refers to data mining using text documents as data.
- Most text mining tasks use Information Retrieval (IR) methods to pre-process text documents.
- These methods are quite different from traditional data pre-processing methods used for relational tables.
- Web search also has its root in IR.

Information Retrieval (IR)

- Conceptually, IR is the study of finding needed information. I.e., IR helps users find information that matches their information needs.
  - Expressed as queries
- Historically, IR is about document retrieval, emphasizing document as the basic unit.
  - Finding documents relevant to user queries
- Technically, IR studies the acquisition, organization, storage, retrieval, and distribution of information.

IR architecture



IR queries

- Keyword queries
- Boolean queries (using AND, OR, NOT)
- Phrase queries
- Proximity queries
- Full document queries
- Natural language questions

Information retrieval models

- An IR model governs how a document and a query are represented and how the relevance of a document to a user query is defined.
- Main models:
  - Boolean model
  - Vector space model
  - Statistical language model
  - etc

Boolean model

- Each document or query is treated as a **"bag" of words** or **terms**. Word sequence is not considered.
- Given a collection of documents D, let $V = \{t_1, t_2, ..., t_{|V|}\}$ be the set of distinctive words/terms in the collection. V is called the vocabulary.
- A weight $w_{ij} \epsilon \{0, 1\}$ is associated with each term t_i of a document $\mathbf{d}_j \in D$. For a term that does not appear in document $\mathbf{d}_j$, $w_{ij} = 0$; otherwise $w_{ij} = 1$.

$$\mathbf{d}_j = (w_{1j}, w_{2j}, ..., w_{|V|j}),$$

Boolean model (contd)

- Query terms are combined logically using the Boolean operators **AND**, **OR**, and **NOT.**
  - E.g., ((*data* AND *mining*) AND (NOT *text*))
- Retrieval
  - Given a Boolean query, the system retrieves every document that makes the query logically true.
  - Called **exact match**.
- The retrieval results are usually quite poor because term frequency is not considered.

Vector space model

- Documents are also treated as a “bag” of words or terms.
- Each document is represented as a vector.
- However, the term weights are no longer 0 or 1. Each term weight is computed based on some variations of TF or TF-IDF scheme.

- Term Frequency (TF) Scheme: The weight of a term t_i in document $\mathbf{d}_j$ is the number of times that t_i appears in $\mathbf{d}_j$, denoted by f_{ij}. Normalization may also be applied.

TF-IDF term weighting scheme

- The most well known weighting scheme
  - TF: still **term frequency**
  - IDF: **inverse document frequency**.

  N: total number of docs

  df_i: the number of docs that t_i appears.
- The final TF-IDF term weight is:

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

$$idf_i = \log \frac{N}{df_i}$$

$$w_{ij} = tf_{ij} \times idf_i.$$

Retrieval in vector space model

- Query **q** is represented in the same way or slightly differently.
- Relevance of $\mathbf{d}_i$ to **q**: Compare the similarity of query **q** and document $\mathbf{d}_i$.
- Cosine similarity (the cosine of the angle between the two vectors)

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

- Cosine is also commonly used in text clustering

An Example

- A document space is defined by three terms:
  - hardware, software, users
  - the vocabulary
- A set of documents are defined as:
  - A1=(1, 0, 0), A2=(0, 1, 0), A3=(0, 0, 1)
  - A4=(1, 1, 0), A5=(1, 0, 1), A6=(0, 1, 1)
  - A7=(1, 1, 1) A8=(1, 0, 1). A9=(0, 1, 1)
- If the Query is “hardware and software”
- what documents should be retrieved?

An Example (cont.)

- In Boolean query matching:
  - document A4, A7 will be retrieved (“AND”)
  - retrieved: A1, A2, A4, A5, A6, A7, A8, A9 (“OR”)
- In similarity matching (cosine):
  - q=(1, 1, 0)
  - S(q, A1)=0.71, S(q, A2)=0.71, S(q, A3)=0
  - S(q, A4)=1, S(q, A5)=0.5, S(q, A6)=0.5
  - S(q, A7)=0.82, S(q, A8)=0.5, S(q, A9)=0.5
  - Document retrieved set (with ranking)=
    - {A4, A7, A1, A2, A5, A6, A8, A9}

Relevance feedback

- Relevance feedback is one of the techniques for improving retrieval effectiveness. The steps:
  - the user first identifies some relevant (D_r) and irrelevant documents (D_{ir}) in the initial list of retrieved documents
  - the system expands the query **q** by extracting some additional terms from the sample relevant and irrelevant documents to produce $\mathbf{q}_e$
  - Perform a second round of retrieval.
- Rocchio method (α, β and γ are parameters)

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

Rocchio text classifier

- In fact, a variation of the Rocchio method above, called the **Rocchio classification** method, can be used to improve retrieval effectiveness too
  - so are other machine learning methods. Why?
- Rocchio classifier is constructed by producing a prototype vector $\mathbf{c}_i$ for each class i (*relevant* or *irrelevant* in this case):

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

- In classification, cosine is used.

Text pre-processing

- Word (term) extraction: easy
- Stopwords removal
- Stemming
- Frequency counts and computing TF-IDF term weights.

Stopwords removal

- Many of the most frequently used words in English are useless in IR and text mining – these words are called *stop words*.
  - the, of, and, to, ….
  - Typically about 400 to 500 such words
  - For an application, an additional domain specific stopwords list may be constructed
- Why do we need to remove stopwords?
  - Reduce indexing (or data) file size
    - stopwords accounts 20-30% of total word counts.
  - Improve efficiency and effectiveness
    - stopwords are not useful for searching or text mining
    - they may also confuse the retrieval system.

Stemming

- Techniques used to find out the root/stem of a word. E.g.,
  - user
  - users
  - used
  - using

  engineering
  engineered
  engineer
- stem: use engineer

Usefulness:

- improving effectiveness of IR and text mining
  - matching similar words
  - Mainly improve recall
- reducing indexing size
  - combing words with same roots may reduce indexing size as much as 40-50%.

Basic stemming methods

Using a set of rules. E.g.,

- remove ending
  - if a word ends with a consonant other than s,

    followed by an s, then delete s.
  - if a word ends in es, drop the s.
  - if a word ends in ing, delete the ing unless the remaining word consists only of one letter or of th.
  - If a word ends with ed, preceded by a consonant, delete the ed unless this leaves only a single letter.
  - ……
- transform words
  - if a word ends with “ies” but not “eies” or “aies” then “ies --> y.”

Frequency counts + TF-IDF

- Counts the number of times a word occurred in a document.
  - Using occurrence frequencies to indicate relative importance of a word in a document.
    - if a word appears often in a document, the document likely “deals with” subjects related to the word.
- Counts the number of documents in the collection that contains each word
- TF-IDF can be computed.

Evaluation: Precision and Recall

- Given a query:
  - Are all retrieved documents relevant?
  - Have all the relevant documents been retrieved?
- Measures for system performance:
  - The first question is about the precision of the search
  - The second is about the completeness (recall) of the search.

Precision-recall curve

Example 2: Following Example 1, we obtain the interpolated precisions at all 11 recall levels in the table of Fig. 6.4. The precision-recall curve is shown on the right.

i	$p(r_i)$	r_i
0	100%	0%
1	100%	10%
2	100%	20%
3	100%	30%
4	80%	40%
5	80%	50%
6	71%	60%
7	70%	70%
8	70%	80%
9	62%	90%
10	62%	100%

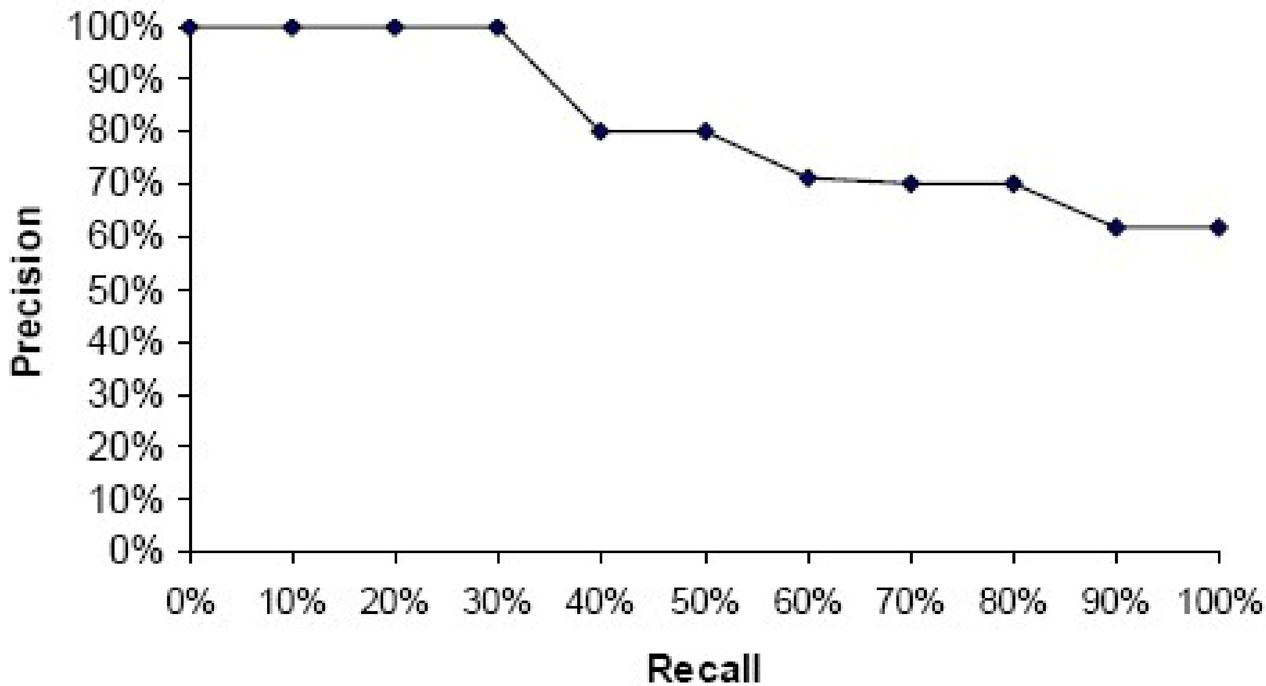


Fig. 6.4. The precision-recall curve ■

Compare different retrieval algorithms

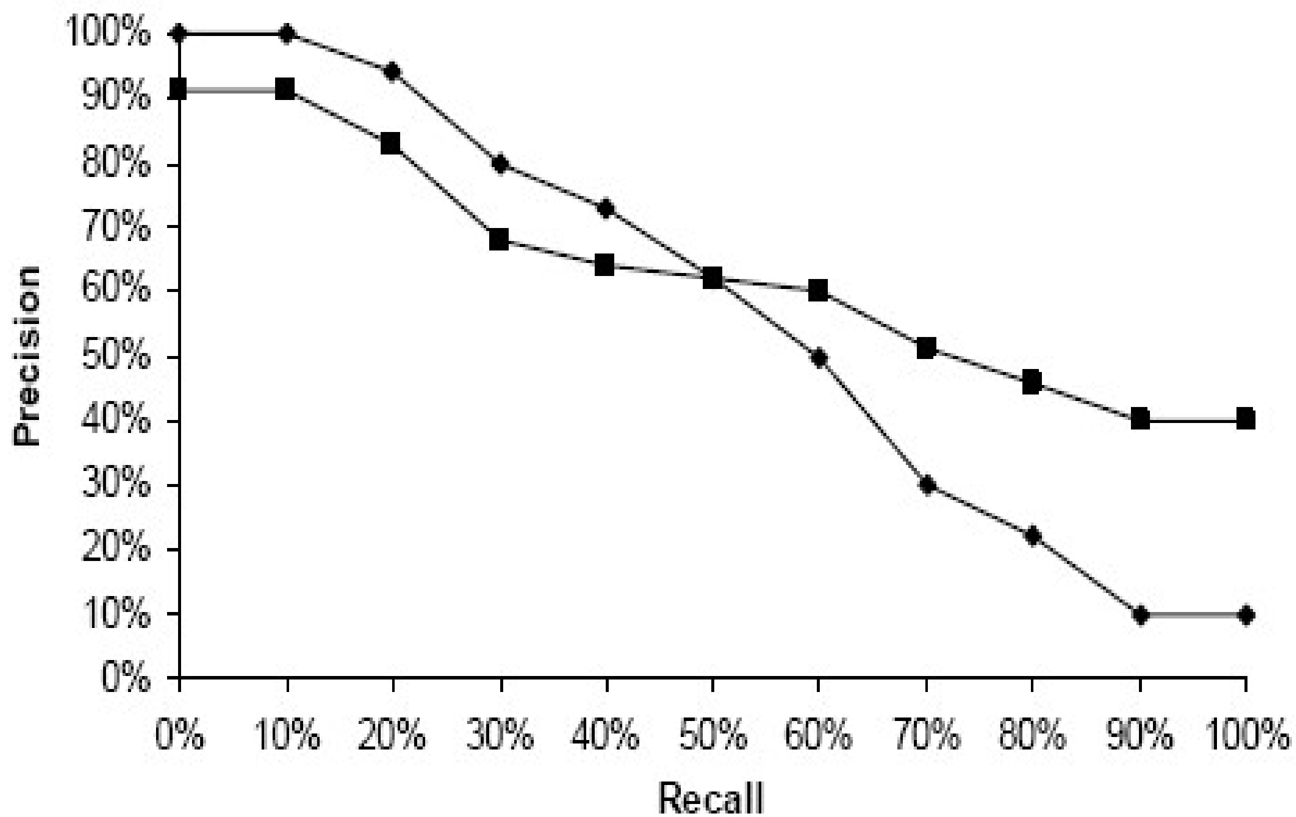


Fig. 6.5. Comparison of two retrieval algorithms based on their precision-recall curves

Compare with multiple queries

- Compute the average precision at each recall level.

$$\overline{p}(r_i) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} p_j(r_i), \qquad (22)$$

where Q is the set of all queries and $p_j(r_i)$ is the precision of query j at the recall level r_i. Using the average precision at each recall level, we can also draw a precision-recall curve.

- Draw precision recall curves
- Do not forget the F-score evaluation measure.

Rank precision

- Compute the precision values at some selected rank positions.
- Mainly used in Web search evaluation.
- For a Web search engine, we can compute precisions for the top 5, 10, 15, 20, 25 and 30 returned pages
  - as the user seldom looks at more than 30 pages.
- Recall is not very meaningful in Web search.
  - Why?

Web Search as a huge IR system

- A Web crawler (robot) crawls the Web to collect all the pages.
- Servers establish a huge inverted indexing database and other indexing databases
- At query (search) time, search engines conduct different types of vector query matching.

Inverted index

- The inverted index of a document collection is basically a data structure that
  - attaches each distinctive term with a list of all documents that contains the term.
- Thus, in retrieval, it takes constant time to
  - find the documents that contains a query term.
  - multiple query terms are also easy handle as we will see soon.

An example

Example 3: We have three documents of id_1, id_2, and id_3:

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

(A)

Applications:	id_2
Hyperlink:	id_3
Mining:	id_1, id_2, id_3
Structure:	id_3
Studies:	id_3
Usage:	id_2
Useful:	id_1
Web:	id_1, id_3

(B)

Applications:	$<id_2, 1, [3]>$
Hyperlink:	$<id_3, 1, [7]>$
Mining:	$<id_1, 1, [2]>$, $<id_2, 1, [2]>$, $<id_3, 1, [3]>$
Structure:	$<id_3, 2, [2, 8]>$
Studies:	$<id_3, 1, [4]>$
Usage:	$<id_2, 1, [1]>$
Useful:	$<id_1, 1, [4]>$
Web:	$<id_1, 1, [1]>$, $<id_3, 2, [1, 6]>$

Fig. 6.7. Two inverted indices: a simple version and a more complex version

Index construction

- Easy! See the example,

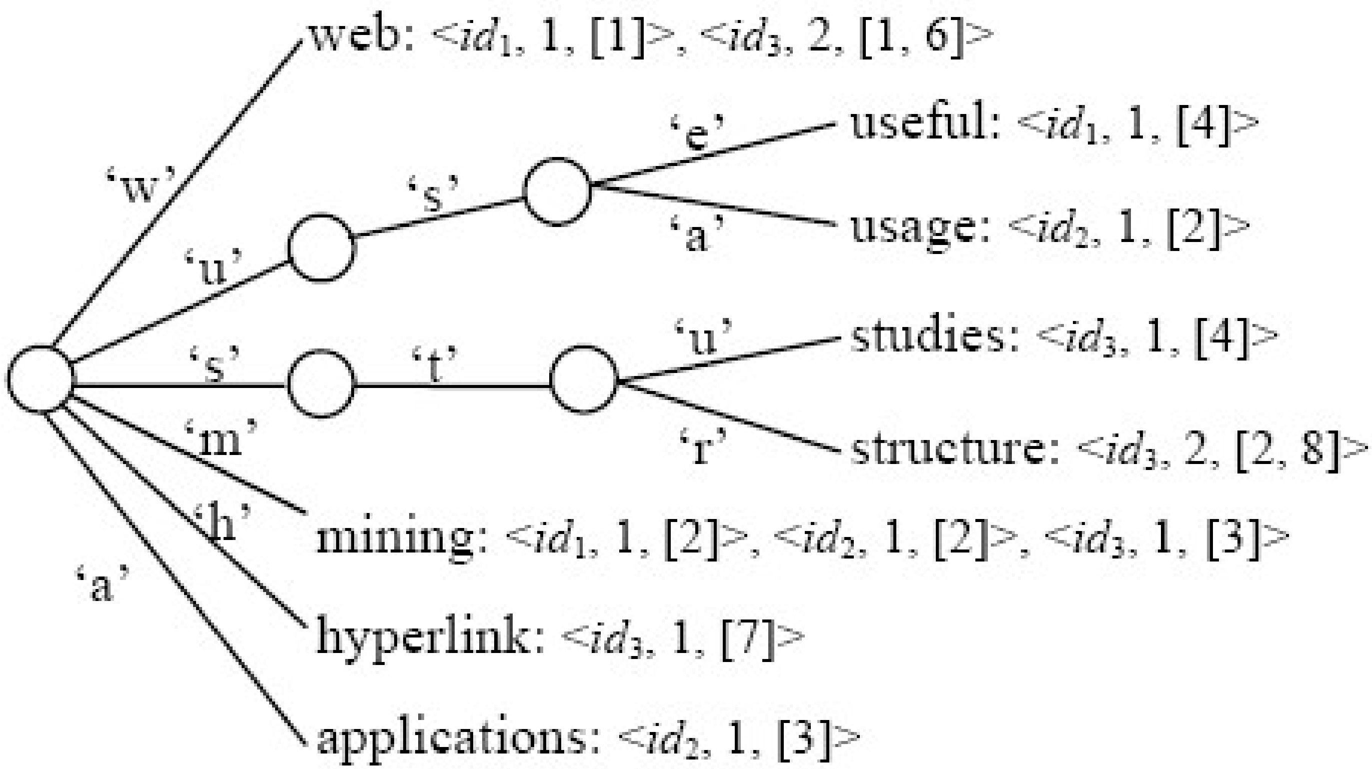


Fig. 6.8. The vocabulary trie and the inverted lists

Search using inverted index

Given a query **q**, search has the following steps:

- Step 1 (**vocabulary search**): find each term/word in **q** in the inverted index.
- Step 2 (**results merging)**: Merge results to find documents that contain all or some of the words/terms in **q**.
- Step 3 (**Rank score computation**): To rank the resulting documents/pages, using,
  - content-based ranking
  - link-based ranking

Different search engines

- The real differences among different search engines are
  - their index weighting schemes
    - Including location of terms, e.g., title, body, emphasized words, etc.
  - their query processing methods (e.g., query classification, expansion, etc)
  - their ranking algorithms
  - Few of these are published by any of the search engine companies. They aretightly guarded secrets.