COMP7630 – Web Intelligence and its Applications

# Web Information Retrieval
## (part 2)

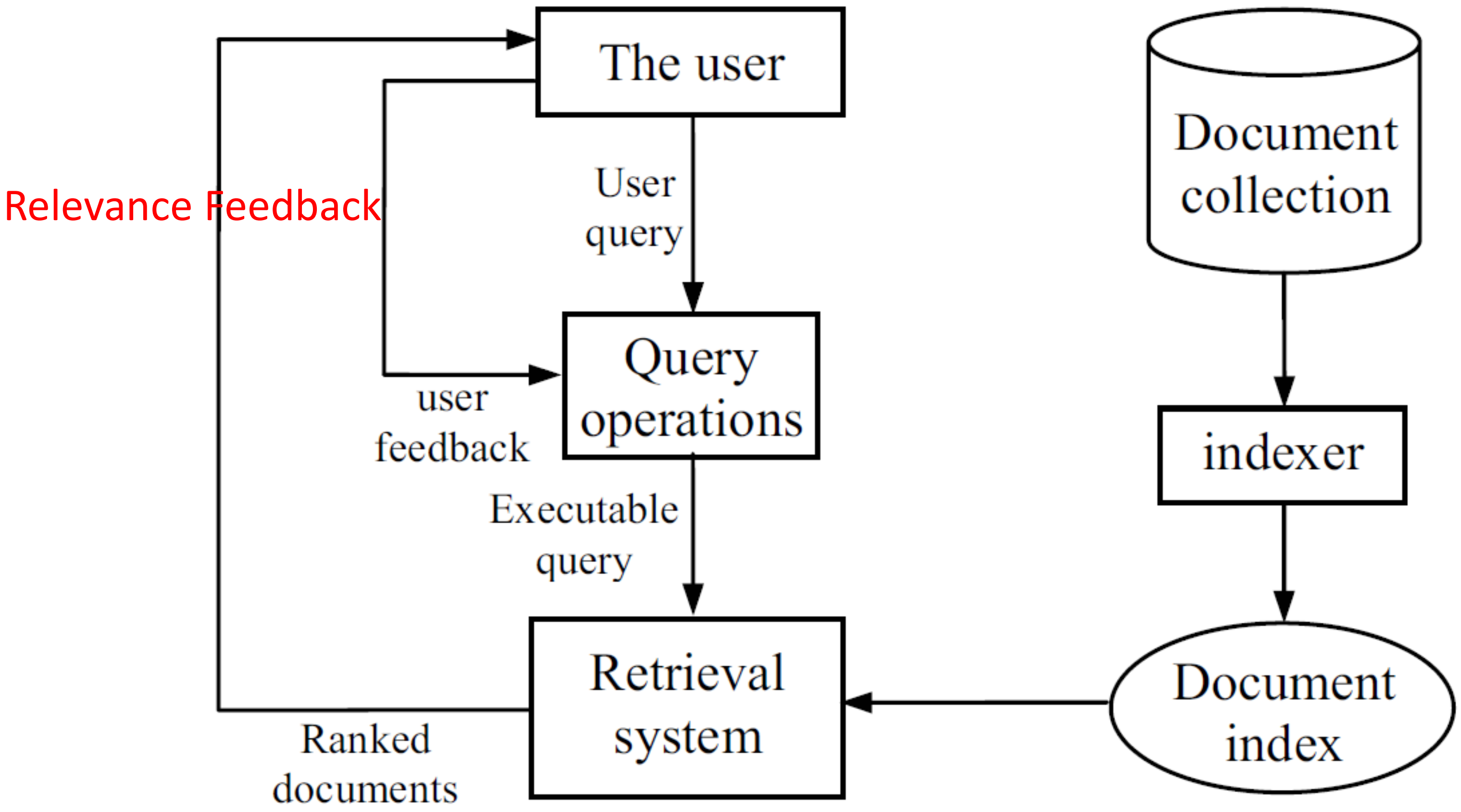
Valentino Santucci

(valentino.santucci@unistrapg.it)

# Outline

- Relevance Feedback
- Evaluation Ranking Quality
- Inverted Index

# Relevance Feedback



# Relevance Feedback via Query

- To provide feedback on relevance to improve retrieval results.
- Starting from the initial list of retrieved documents
  - Step 1: User indicates relevant & irrelevant documents.
  - Step 2: System extracts some additional terms from the relevant and irrelevant documents and expands query for a second round of retrieval.
  - Step 3: Repeat Steps 1&2 until the user is satisfied with the retrieval result.

# Rocchio Method

- For query expansion
  - Original query vector = $q$
  - Set of relevant documents selected by the user be $D_r$
  - Set of irrelevant documents be $D_{ir}$
  - Expanded query $q_e$ becomes:

$$q_e = \alpha q + \frac{\beta}{|D_r|} \sum_{d_r \in D_r} d_r - \frac{\gamma}{|D_{ir}|} \sum_{d_{ir} \in D_{ir}} d_{ir}$$

  - where $\alpha$, $\beta$ and $\gamma$ are parameters
  - Negative entries in $q_e$, if any, are usually truncated to 0

# Rocchio Classification Method

- We have a set of relevant and irrelevant documents, so we can construct a classification model from them.
- Rocchio classifier:

Construct a prototype vector $c_i$ for each class *i*, which is either *relevant* or *irrelevant*

$$\mathbf{c}_i = \frac{\alpha}{|D_i|}\sum_{\mathbf{d}\in D_i}\frac{\mathbf{d}}{\|\mathbf{d}\|} - \frac{\beta}{|D - D_i|}\sum_{\mathbf{d}\in D-D_i}\frac{\mathbf{d}}{\|\mathbf{d}\|}, \tag{15}$$

**Algorithm**

```
for each class i do
    construct its prototype vector c_i using Equation (15)
endfor
for each test document d_t do
    the class of d_t is argmax_i cosine(d_t, c_i)
endfor
```

# Outline

- Relevance Feedback
- Evaluation Ranking Quality
- Inverted Index

# Evaluation Ranking Quality

- Given
  - $D$ - Collection of documents, $N$ - Total number of documents, $q$ - User query,
- Compute relevance scores for all documents in $D$ and produce their ranking $R_q$ based on the scores:

$$R_q: \langle d_1^q, d_2^q, \dots, d_N^q \rangle$$

  - $d_1^q \in D$ is the most relevant document to query $q$
  - $d_N^q \in D$ is the most irrelevant document to query $q$.

# Recall and Precision

- Let $D_q$ ($\subseteq D$) be the set of actual relevant documents of query $q$ in $D$. We can compute the precision and recall values at each $d_i^q$ in the ranking.
- Recall at rank position $i$ or document $d_i^q$ (denoted by $r(i)$) is the fraction of relevant documents from $d_1^q$ to $d_i^q$ in $R_q$.

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

where $s_i$ ($\leq |D_q|$) is the number of relevant docs from $d_1^q$ to $d_i^q$ in $R_q$

- Precision at rank position $i$ or document $d_i^q$ (denoted by $p(i)$) is the fraction of documents from $d_1^q$ to $d_i^q$ in $R_q$ that are relevant:

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

# Example 1

- A document collection $D$ with 20 documents.
- Given a query $q$, we know that 8 documents are relevant to $q$.
- A retrieval algorithm produces the ranking on the right.

| Rank $i$ | +/– | $p(i)$ | $r(i)$ |
|---|---|---|---|
| 1 | + | 1/1 = 100% | 1/8 = 13% |
| 2 | + | 2/2 = 100% | 2/8 = 25% |
| 3 | + | 3/3 = 100% | 3/8 = 38% |
| 4 | – | 3/4 = 75% | 3/8 = 38% |
| 5 | + | 4/5 = 80% | 4/8 = 50% |
| 6 | – | 4/6 = 67% | 4/8 = 50% |
| 7 | + | 5/7 = 71% | 5/8 = 63% |
| 8 | – | 5/8 = 63% | 5/8 = 63% |
| 9 | + | 6/9 = 67% | 6/8 = 75% |
| 10 | + | 7/10 = 70% | 7/8 = 88% |
| 11 | – | 7/11 = 63% | 7/8 = 88% |
| 12 | – | 7/12 = 58% | 7/8 = 88% |
| 13 | + | 8/13 = 62% | 8/8 = 100% |
| 14 | – | 8/14 = 57% | 8/8 = 100% |
| 15 | – | 8/15 = 53% | 8/8 = 100% |
| 16 | – | 8/16 = 50% | 8/8 = 100% |
| 17 | – | 8/17 = 53% | 8/8 = 100% |
| 18 | – | 8/18 = 44% | 8/8 = 100% |
| 19 | – | 8/19 = 42% | 8/8 = 100% |
| 20 | – | 8/20 = 40% | 8/8 = 100% |

# Average Precision

- An average precision ($p_{avg}$) can be computed based on the precision at each relevant document in the ranking.

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

- $D_q$ is the set of relevant documents.

- Refer to Example 1
- $p_{avg} = \frac{100\%+100\%+100\%+80\%+71\%+67\%+70\%+62\%}{8} = 81\%$

# Precision-Recall Curve

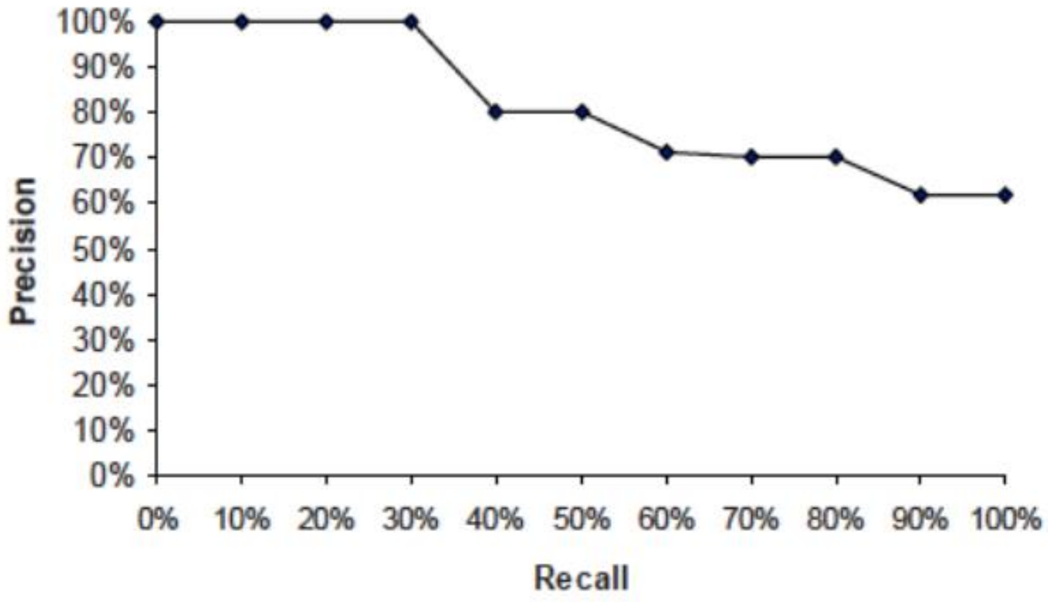
- A plot with $x$-axis being recall and $y$-axis being precision.
- It is commonly plotted using 11 standard recall levels, 0%, 10%, 20%, ..., 100%.
- It is hard to have recall levels exactly taking those standard values. Interpolation can be used.
- Let $r_i$ be a recall level (0%, 10%, 20%, ... or 100%) and $p(r_i)$ be the precision at the recall level $r_i$.

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

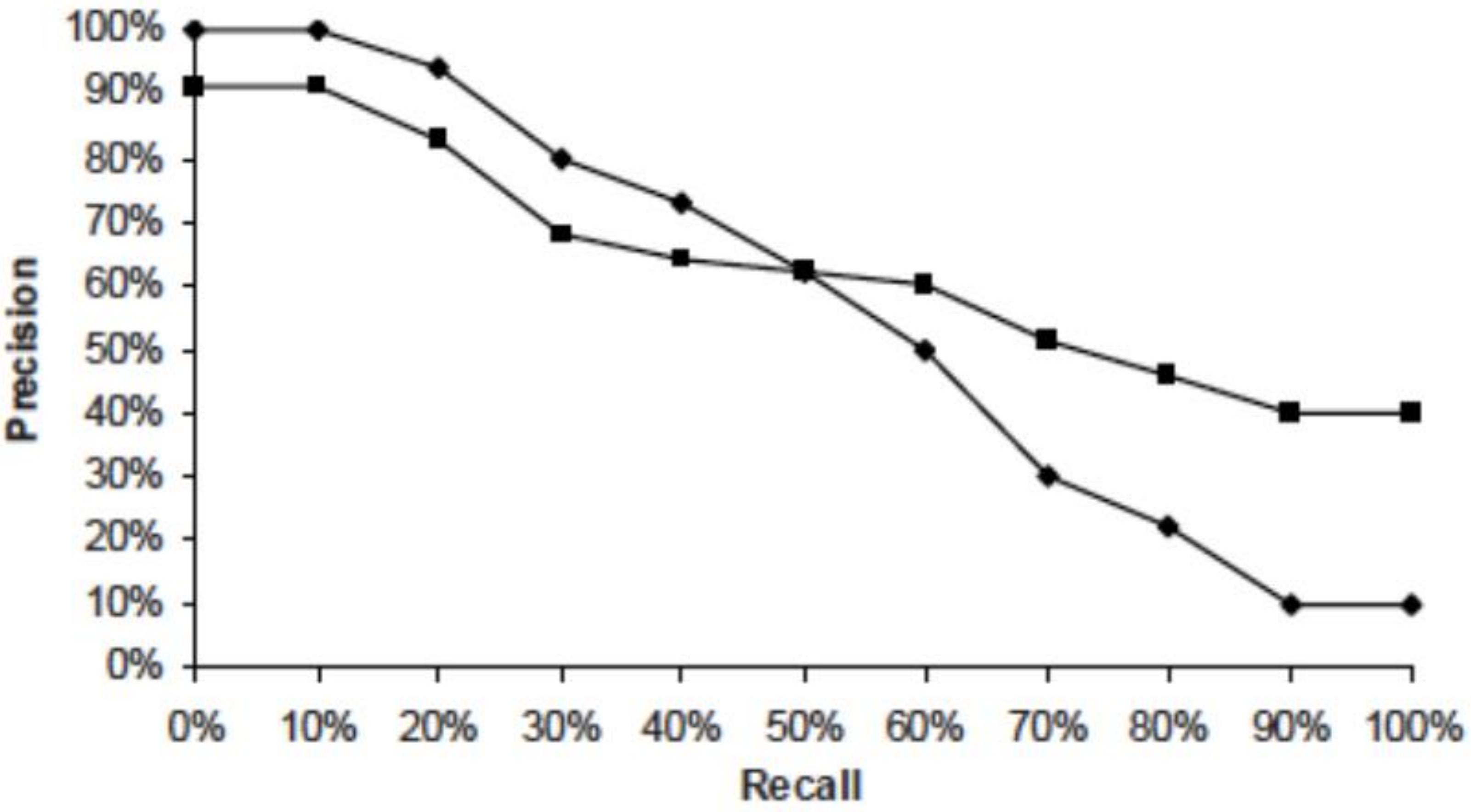
# Precision-Recall Curve

| $p(i)$ | $r(i)$ |
|---|---|
| 1/1 = 100% | 1/8 = 13% |
| 2/2 = 100% | 2/8 = 25% |
| 3/3 = 100% | 3/8 = 38% |
| 3/4 = 75% | 3/8 = 38% |
| 4/5 = 80% | 4/8 = 50% |
| 4/6 = 67% | 4/8 = 50% |
| 5/7 = 71% | 5/8 = 63% |
| 5/8 = 63% | 5/8 = 63% |
| 6/9 = 67% | 6/8 = 75% |
| 7/10 = 70% | 7/8 = 88% |
| 7/11 = 63% | 7/8 = 88% |
| 7/12 = 58% | 7/8 = 88% |
| 8/13 = 62% | 8/8 = 100% |
| 8/14 = 57% | 8/8 = 100% |
| 8/15 = 53% | 8/8 = 100% |
| 8/16 = 50% | 8/8 = 100% |
| 8/17 = 53% | 8/8 = 100% |
| 8/18 = 44% | 8/8 = 100% |
| 8/19 = 42% | 8/8 = 100% |
| 8/20 = 40% | 8/8 = 100% |

| $i$ | $p(r_i)$ | $r_i$ |
|---|---|---|
| 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% |



# Comparison of Retrieval Algorithms



# Better a high precision or a high recall?

- An application-agnostic way to choose a retrieval algorithm: calculate the Area Under the Curve (AUC) and select the algorithm with the largest AUC.

- However, often the decision depends on the specific requirements of the application at hand. For example:
  - E-commerce search: both precision and recall are important, but precision is usually more important. In fact, customers expect to see relevant results when they search for a product, and irrelevant results can lead to a negative user experience.
  - Medical information retrieval: recall is usually more important than precision. In fact, missing a relevant medical document could lead to a misdiagnosis or an incorrect treatment plan.

# Evaluation Using Multiple Queries

- The overall precision at each recall level $r_i$ is computed as the average of individual precisions at that recall level, i.e.,

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

- $Q$ is the set of all queries
- $p_j(r_i)$ 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.

# Example 2

- $D = \{d_1, d_2, d_3, d_4, d_5\};\ N = 5$
- Relevant documents $D_q = \{d_2, d_5\}$
- Irrelevant documents $D \setminus D_q = \{d_1, d_3, d_4\}$
- Given a query q, the documents in $D$ are ranked as:

| Rank Position | *i=1* | *i=2* | *i=3* | *i=4* | *i=5* |
|---|---|---|---|---|---|
| | $d_1^q$ | $d_2^q$ | $d_3^q$ | $d_4^q$ | $d_5^q$ |
| | $d_2$ | $d_1$ | $d_5$ | $d_3$ | $d_4$ |

# Example 2

| Rank Position | *i=1* | *i=2* | *i=3* | *i=4* | *i=5* |
|---|---|---|---|---|---|
| | $d_1^q$ | $d_2^q$ | $d_3^q$ | $d_4^q$ | $d_5^q$ |
| | $d_2$ | $d_1$ | $d_5$ | $d_3$ | $d_4$ |
| r(i) | 1/2 | 1/2 | 2/2 | 2/2 | 2/2 |
| | 0.5 | 0.5 | 1.0 | 1.0 | 1.0 |
| p(i) | 1/1 | 1/2 | 2/3 | 2/4 | 2/5 |
| | 1.0 | 0.5 | 0.67 | 0.5 | 0.4 |

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

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

# Example 2

| Rank | *i=1* | *i=2* | *i=3* | *i=4* | *i=5* |
|---|---|---|---|---|---|
| | $d_1^q$ | $d_2^q$ | $d_3^q$ | $d_4^q$ | $d_5^q$ |
| | $d_2$ | $d_1$ | $d_5$ | $d_3$ | $d_4$ |
| r(i) | 1/2 | 1/2 | 2/2 | 2/2 | 2/2 |
| | 0.5 | 0.5 | 1.0 | 1.0 | 1.0 |
| p(i) | 1/1 | 1/2 | 2/3 | 2/4 | 2/5 |
| | 1.0 | 0.5 | 0.67 | 0.5 | 0.4 |

| Rank i | +/- | p(i) | r(i) |
|---|---|---|---|
| 1 | + | 1.0 | 0.5 |
| 2 | - | 0.5 | 0.5 |
| 3 | + | 0.67 | 1.0 |
| 4 | - | 0.5 | 1.0 |
| 5 | - | 0.4 | 1.0 |

Average Precision = (1.0 + 0.67)/2 = 0.835

# Precision@5/10/15

- For Web Search, it can be very hard to determine the set of relevant documents $D_q$ for each query $q$.
- Without $D_q$, the recall value cannot be computed. In fact, recall does not make much sense for Web search because the user seldom looks at pages ranked below 30.
- But precision is critical (quality of top ranked docs).
- Refer to Example 1, we have p@5 = 80%, p@10 = 70%, p@15 = 53%, and p@20 = 40%.

# F-Score

It is the harmonic mean of precision and recall

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

# Outline

- Relevance Feedback
- Evaluation Ranking Quality
- <u>Inverted Index</u>

# Text pre-processing

- Before indexing terms, we have to extract terms and (possibly) simplify them through text pre-processing
  - Lowercase reduction (easy)
  - Word (term) extraction: requires tokenization (easy)
  - Punctuation removal (easy)
  - Stopwords removal
  - Normalization through stemming or lemmatization

# 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.

# Normalization of terms

- Lemmatization exploits linguistic rules and provide the "vocabulary form" of a word

- Stemming is a technique used to find out the root/stem of a word.
  - User, users, used, using -> use
  - Engineering, engineered, engineer -> engineer
  - Various -> variou (not necessarily a real word!)

- Combination of lemmatization and stemming can provide a much higher grouping
  - Replace a *word* with *stem*(*lemma*(*word*))

# Basic stemming methods

- Stemming are usually crude heuristic processes that chop off the end of a word using a set of predefined rules

- Examples of stemming rules:
  - 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.”

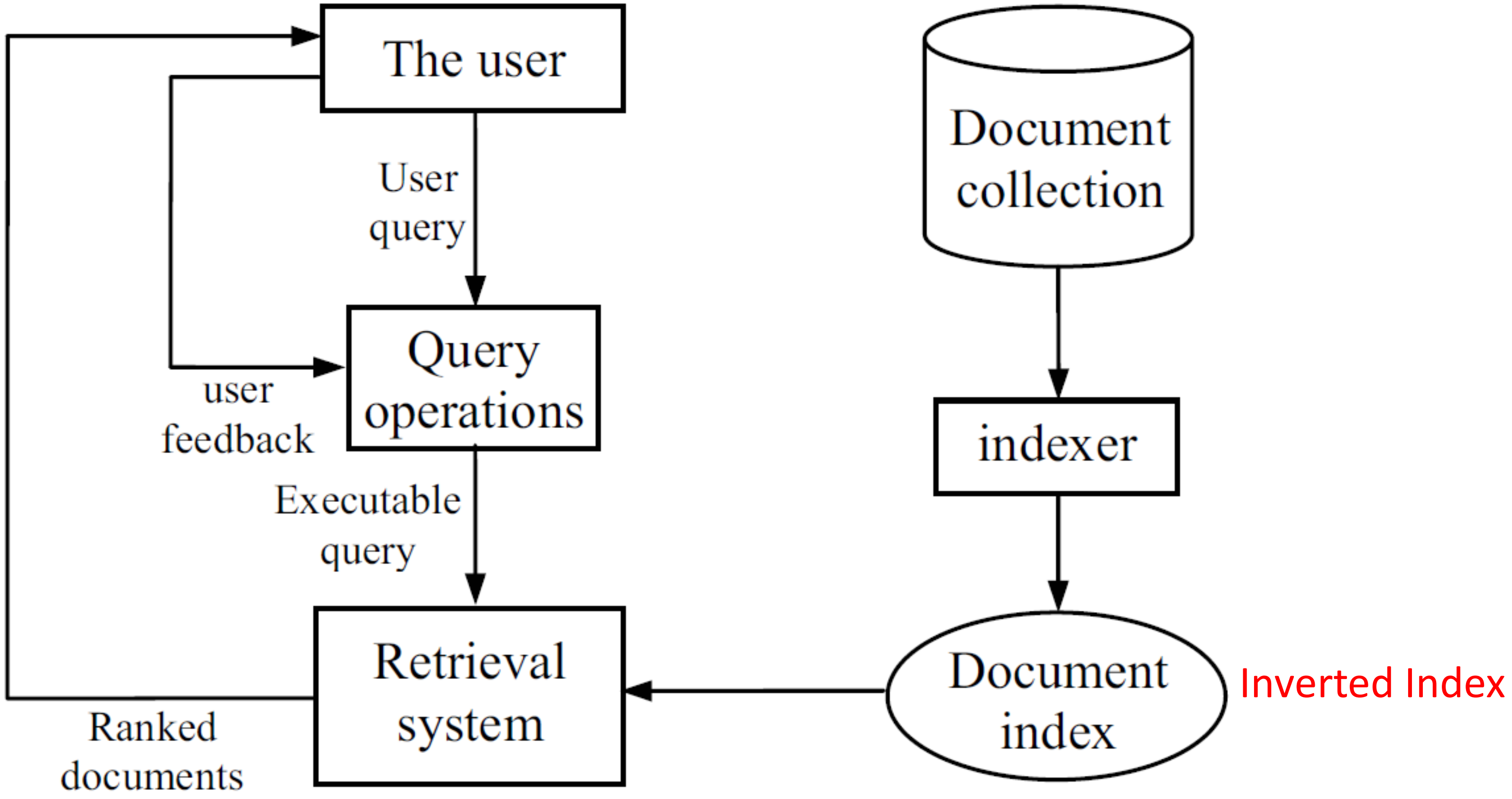
- We already saw how to apply the well-known Porter stemmer using Python (when we talked about NLP pipelines)

# Usefulness of term's normalization

- 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%.

# Inverted Index



# Indexer

- Indexer:
  - indexes the original raw documents in some data structures to enable efficient retrieval.
  - Inverted index is commonly used in search engines and most IR systems. Easy to build and efficient to search.

# Inverted Index

- A data structure (index) allows efficient retrieval
- Support phrase and proximity search based on query terms
- Create a posting for each term $t_i$ and each document $d_j$

$$\left\langle id_j, f_{ij}, \left[o_1, o_2, \dots, o_{|f_{ij}|}\right]\right\rangle$$

where

$id_j$ is the ID of document $d_j$ that contains the term $t_i$,

$f_{ij}$ is the frequency count of $t_i$ in $d_j$

$o_k$ are the offsets (or positions) of term $t_i$ in $d_j$.

*$id_1$*: 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

Terms "is" and "the" are ignored. Why?

Inverted list (contains postings)

vocabulary

| | |
|---|---|
| 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: | $<id_1, 1, [1]>$, $<id_3, 2, [1, 6]>$ |

E.g. Query = "web mining"

# Search using an Inverted Index

- Given the query terms:
  - Step 1 (Vocabulary Search): Search for each query term in vocabulary to obtain the inverted list of each term.
  - Step 2 (Results Merging): Merge the inverted lists of the terms to locate the documents with all the terms.
  - Step 3 (Results Ranking): Compute a relevance score for each document based on a function (as in the IR models previously described)
    - The phrase and term proximity information can also be considered.

Query = “web mining”

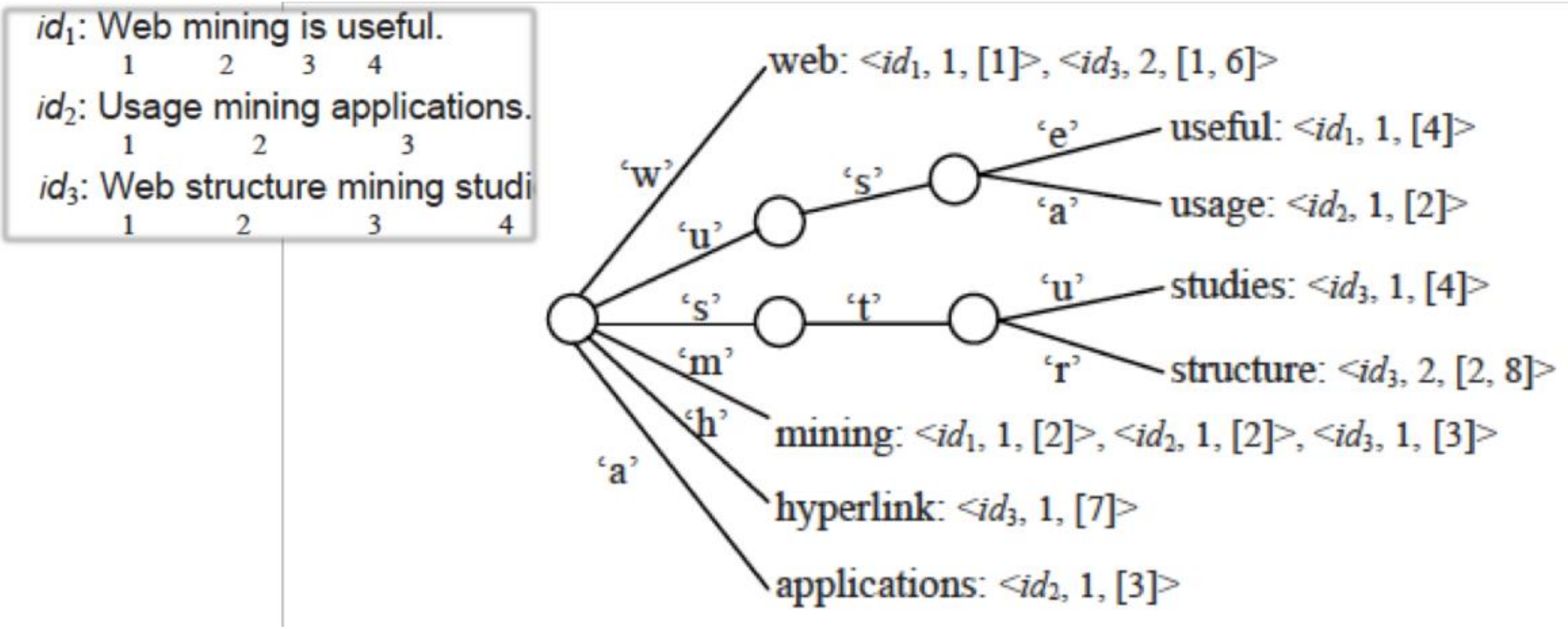
- Step 1

| | |
|---|---|
| Mining: | $<id_1, 1, [2]>$, $<id_2, 1, [2]>$, $<id_3, 1, [3]>$ |
| Web: | $<id_1, 1, [1]>$, $<id_3, 2, [1, 6]>$ |

- Step 2
  - Traverse the two lists and finds documents containing the terms “web” and “mining (i.e., $d_1$ and $d_3$)
- Step 3
  - Compute the relevance score
  - In  terms of proximity and word ordering: $d_1$ will be ranked higher than $d_3$. Why?
    - “web” and “mining” are next to each other in $d_1$
    - They are in the same order as that in the query.

# Index Data Structure

- An inverted index adopting a trie data structure



# 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
- At query (search) time, search engines conduct different types of vector query matching.

# 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 are tightly guarded secrets …
  - … but, often, they include additional factors in the relevance scores, like centrality measures calculated for the web pages calculated on the basis of their "social network structure" (we will see them when we will talk about Social Network Analysis)

# References


- *Liu, Bing. Web data mining: exploring hyperlinks, contents, and usage data. Berlin: springer, 2011. Chapter 6.*