University Politehnica of Bucharest
Faculty of Automatic Control and Computers
Computer Science and Engineering Department

Information Retrieval Systems

Ciprian-Octavian Truică
ciprian.truica@cs.pub.ro

Overview

- Information Retrieval
- Text Preprocessing
- Inverted Index
- Latent Semantic Indexing

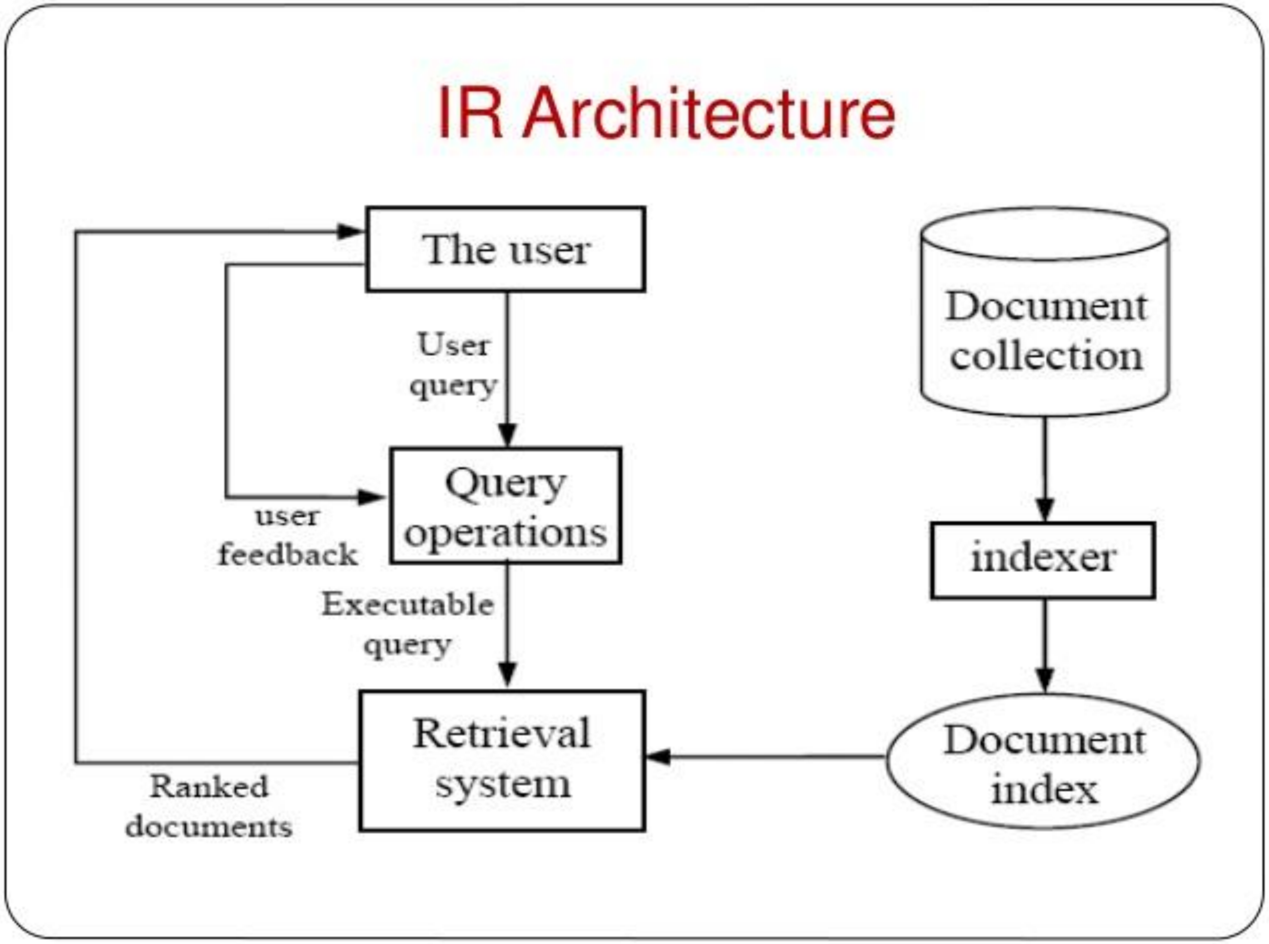
Overview

- Information Retrieval
- Text Preprocessing
- Inverted Index
- Latent Semantic Indexing

Information Retrieval

- Information Retrieval (IR) is the field that deals with retrieving relevant information from a large corpus of records given a user search query
- In traditional IR:
  - The records are documents:
  - The search query is a list of words (terms)
  - The output contains the relevant documents for the search query

Information Retrieval Architecture



Information Retrieval Architecture

- User Queries types:
  - Keywords queries (search terms – most used): The user uses a list of terms (at list one) with the aim to retrieve documents that contain at least one or all of the search terms
    - Soft IR systems – the documents returned contain at least one keyword (logical OR between the terms – added automatically)
    - Hard IR systems – the documents returned contain all the keywords (logical AND between the terms – added automatically)
    - In some IR systems the order of the keywords is also important – returns documents where the list of words appear together
  - Note:
    - Bag of Words (BOW) approach is used when the order is not important
    - N-gram approach is used when the order is important

Information Retrieval Architecture

- User Queries types:
  - Boolean queries (BOW approach): the user can use Boolean operators in their search queries (AND, OR, and NOT)
    - Example 1: ‘data or web’ or ‘data and web’ – different results
    - Example 2: ‘data or web and not datamining’

Information Retrieval Architecture

- User Queries types:
  - Phrase queries (n-gram approach):
    - Such a query consists of a sequence of words that make up a phrase
    - The documents returned must contain at least one instance of this phrase
  - Proximity queries
    - Is a relaxed version of the phase query and can be a combination of terms and phrases
    - These queries seek the documents that contain the search term in close proximity of each other

Information Retrieval Architecture

- User Queries types:
  - Full document queries
    - Users search for documents that are similar to the query document
  - Natural language queries
    - This is the most complex case
    - The user asks a question and the IR system returns an answer to the that question
    - Used in question-answering systems

Information Retrieval Architecture

- Query operation module
  - Can range between very
    - Simple: just passes the query to the retrieval system
    - Complex: does preprocessing
- Indexer module
  - Used to index the original raw documents in some data structures
  - The data structures enables efficient retrieval
  - Most common is the inverted index

Information Retrieval Architecture

- Retrieval system module:
  - Computes a relevance score (for the user query) for each retrieved documents
  - According the relevance the documents are ranked and presented to the user
- Document collection module:
  - A File System (FS), e.g. OSFS, HDFS, etc.
  - A database (Relational or NoSQL).

Information Retrieval Models

- The way in which terms and documents are represented governs the IR model
- There are three main document representations:
  - Boolean model
  - Vector Space model
  - Statistical Language model

- A document is represented using the bag of words (terms) model:
- Given a set (collection) of n documents:

$$D = \{d_1, d_2, \dots, d_n\}, n = |D|$$

- All the distinct terms in the collection of documents can be modeled as a vocabulary:

$$V = \{t_1, t_2, \dots, t_m\}, m = |V|$$

Document Representations

- The entire corpus of documents can be represented as a matrix (document-term matrix) where the lines represent the document in D and the columns represent the terms in V
- Each cell in the matrix is a weight (w_{ij}) associated for the number of occurrences of a term in the document.
- A document $d_i \in D$ is modeled as a vector $d_i = \{w_{i1}, w_{i2}, \ldots, w_{jm}\}$ where each w_{ij} is the weight associated to the term $t_j \in V$

- The document-term matrix is:

$$
\begin{array}{cc}
 & \begin{array}{cccc} t_1 & t_2 & \cdots & t_m \end{array} \\
\begin{array}{c} d_1 \\ d_2 \\ \vdots \\ d_n \end{array} &
\begin{pmatrix}
w_{11} & w_{12} & \cdots & w_{1m} \\
w_{21} & w_{22} & \cdots & w_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
w_{n1} & w_{n2} & \cdots & w_{nm}
\end{pmatrix}
\end{array}
$$

- Boolean model
  - This is one of the simples models used to represent the weights in the document-term matrix
  - This model only considers if a term is present or not in a document, $w_{ij} \in \{0, 1\}$:

$$w_{ij} = \begin{cases} 1 & if\ t_j\ appears\ in\ d_i \\ 0 & otherwise \end{cases}$$

- Boolean model
  - This model is used for Boolean queries
  - For example, given 3 terms $x, y\ and\ z$:
    - $(x\ AND\ y)\ AND\ (NOT\ z)$ says that a document must contain both x and y but not he term z
    - $x\ OR\ z$ says that a document must contain at least one of the terms x or z

Information Retrieval
Weighting Schemas

- Vector space model
  - One of the best known and widely used IR models
  - Each weight is computed as a variation of the number of occurrences of the term in the document or collection of documents
  - Most widely used representation are:
    - $f_{t,d}$ the frequency of the term t in the document d is computed by counting
    - $TF(t,d)$ term frequency, the normalization of $f_{t,d}$
    - $TFIDF(t,d,D)$ term frequency – inverted document frequency
    - $Okapi\ BM25$
      - Okapi is the name of the information system proposed by Robertson, Spärck Jones, et al.
      - BM Best match

- Weighting schemas
  - $TF(t,d)$ term frequency is a normalization of the $f_{t,d}$
  - Logarithmic normalized $TF(t,d)$:

$$TF(t,d) = 1 + \log f_{t,d}$$

  - This has a bias towards longer documents because in longer documents terms that are irrelevant can appear multiple times and thus these terms have a higher $TF(t,d)$

- Weighting schemas
  - To removes the bias towards longer documents, the augmented $TF(t,d)$

$$TF(t,d) = K + (1-K)\cdot\frac{f_{t,d}}{\max_{t'\in d} f_{t',d}}$$

  - For $K = 0.5$ we obtained the double normalization form of the $TF(t,d)$:

$$TF(t,d) = 0.5 + 0.5\cdot\frac{f_{t,d}}{\max_{t'\in d} f_{t',d}}$$

- Weighting schemas
  - Another way to removes the bias towards longer documents is normalize the frequency of the term with the length of the document

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

- Weighting schemas
  - The $IDF(t, D)$ is a measure of how much information a term provides:
    - Weather a term is common or rare across all documents
    - n is the number of documents in D
    - n_t is the number of documents where term t appears:

$$IDF = 1 + \log \frac{n}{n_t}$$

  - $TFIDF(t, d, D)$ is used to determine the importance of a term for a document in a corpus of documents:

$$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D)$$

- Weighting Schemas
  - Okapi BM25 is another weighting schema used to compute the weights for the document-term matrix

$$OkapiBM25(t,d,D) = \frac{TFIDF(t,d,D) \cdot (k_1 + 1)}{TF(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{avg_{d' \in D}(|d'|)}\right)}$$

  - The free parameters values are: $k_1 \in [1.2, 2.0]$ and $b = 0.75$
  - $|d|$ is the
  - $avg_{d' \in D}(|d'|)$ is the average document length

- Ranking functions
  - Given a search query $Q = (q_1, q_2, \dots, q_k)$ where q_i are the query terms ($i = \overline{1, k}$, $k = |Q|$),
  - The objective is to retrieve the documents that contain the terms in the search query Q
  - We need to give a score that measures the importance of each document to the query
  - Then order the list of documents using this score and create a documents ranking

- Ranking functions
  - The score for a document $d \in D$ is the sums of the weights $(TFIDF, OkapiBM25)$ computed for each search term in query Q

$$S\_TFIDF(Q, d, D) = \sum_{i=1}^{k} TFIDF(q_i, d, D)$$

$$S\_OkapiBM25(Q, d, D) = \sum_{i=1}^{k} OkapiBM25(q_i, d, D)$$

  - Using one of these scores, the documents' relevance to the query is determined

- Statistical Language Model
  - Are based on probability and have foundations in statistical theory
  - Given a set of n documents $D = \{d_1, d_2, \dots, d_n\}$ and a query $Q = (q_1, q_2, \dots, q_k)$
  - In the statistical language model, we consider a query Q as being "generated" by a probabilistic model based on a document d_i
  - Using the Bayes rule, the rank of documents are estimated by the posterior probability $p(d_i|q)$:

$$p(d_i|Q) = \frac{p(Q|d_i) \cdot p(d_i)}{p(Q)}$$

- ## Statistical Language Model
  - For ranking, $p(Q)$ is not needed, as is the same for all the documents
  - The model assumes that each term is independently generated, which is essentially a multinomial distribution over words, and so:

$$p(Q = (q_1, q_2, \dots, q_k)|d_i) = \prod_{j=1}^{k} p\left(q_j|d_i\right) = \prod_{j=1}^{m} p(t_i|d_i)^{f_{jq}}$$

  - Where f_{jq} is the number of times the term t_j appears in Q
  - So, the retrieve problem is reduced to estimating $p\left(t_j|d_i\right)$:

$$p\left(t_j|d_i\right) = \frac{f_{ij}}{|d_i|}$$

  - Where f_{ij} is actually the frequency of word t_j in document d_i, f_{t_j,d_i}

Overview

- Information Retrieval
- Text Preprocessing
- Inverted Index
- Latent Semantic Indexing

Text Preprocessing

- Before the documents in a collection are used for retrieval some preprocessing steps are usually performed
- These steps may include:
  - Expanding contractions
  - Sentence tokenization
  - Extracting terms
  - Removing stop words and punctuation

Text Preprocessing

- Expanding contractions, i.e., shortened versions of the written and spoken forms of a word, syllable, or word group, created by omission of internal letters and sounds
- Sentence tokenization is the process by which the text of a document is split in sentences.
  - This step can be skipped if the terms are not processed further

Text Preprocessing

- Extracting terms is the process by which each term is determined
  - Sometimes it is useful to extract the stem or the lemma of a word (not useful for opinion mining and sentiment analysis)
  - To extract the lemma sentence tokenization is used and also a part of speech tagger

Text Preprocessing

- Extracting term – stemming
  - In many languages, a word has various forms depending on the context, e.g. verbs have the gerund form (-ing termination) or nouns can be singular or plural.
  - This variations cause low retrieval because a relevant document can contain variation of a word but not the exact word
  - By removing the suffixes and prefixes of a word we can obtain the stem of the word
  - Stemming is the process of extracting the stems
  - The stems sometimes are not accurate, e.g. the words “cops” and “cope” are both reduce to “cop”

- Extracting term – lemmatization
  - A more accurate way of extracting the root word is by extracting the lemma, this process is called lemmatization
  - Lemmatization uses the part of speech of a word, the process that extracts the part of speech is called Part of Speech Tagging (PoS)
  - This is a very costly process but it increases the accuracy of the retrieval

Text Preprocessing

- Removing stop words and punctuation
  - Stop words are frequently occurring and insignificant words in a language, e.g.: the, a, an, etc.
  - These can be removes together with punctuation as they add no significate information to the process of information retrieval
  - For some tasks are important, e.g. opinion mining and sentiment analysis.

Overview

- Information Retrieval
- Text Preprocessing
- Inverted Index
- Latent Semantic Indexing

Inverted Index

- The main purpose of IR is to retrieve documents given a search query
- One approach is to scan the collection of documents and to return the documents that match the query terms.
  - This method is actually impractical for a large collection of documents
- A second approach is to build a data structure (inverted index) that maps each word to the documents where it appears.

Inverted Index

- How to construct an inverted index
  1. Given a collection of documents $D = \{d_1, d_2, \dots, d_n\}, n = |D|$
  2. Attach to each documents an unique identifier $\{id_1, id_2, \dots, id_n\}$ where id_i is the unique identifier for document d_i
  3. Construct the vocabulary
  4. Attach to each word in the vocabulary the list of documents that contain the word

Inverted Index

- In practice, an inverted index is a dictionary with the key the word and the value a list of documents.

$$t_i = \{id_j |\, t_i \in d_i\}$$

Inverted Index

- Sometimes, additional information can be stored in the inverted index, e.g.:
  - The term frequency of the word in the document
  - A list with positions, etc.

$$t_i = \left\{ < id_j, f_{ij}, \left[o_1, o_2, \ldots, o_{|d_j|} \right] > | t_i \in d_i \right\}$$

- Where:
  - f_{ij} is the frequency of term t_i in document d_i
  - $o_k, k = \overline{1, |d_i|}$ is the position of term t_i in document d_i

Inverted Index

- E.g.

id_1: I am learning about inverted indexes.

id_2: Inverted indexes are used in information retrieval.

id_3: Applications that retrieve documents use inverted indexes.

- The vocabulary is:
  $\{learning, inverted, indexes, information, retrieval, applications, retrieve, documents\}$
- Notes:
  - the stop words were removed: I, am, about, are, used, in, that, use
  - Stemming or lemmatization can be applied, in this case it wasn't.

Inverted Index

- Simple Inverted Index

$$learning: [id_1],$$
$$inverted: [id_1, id_2, id_3],$$
$$indexes: [id_1, id_2, id_3],$$
$$information: [id_1],$$
$$retrieval: [id_2],$$
$$applications: [id_3],$$
$$retrieve: [id_3],$$
$$documents: [id_3]$$

Inverted Index

- Complex Inverted Index:

$$learning\colon\ [<id_1, 1, [3]>],$$
$$inverted\colon\ [<id_1, 1, [5]>, <id_2, 1, [1]>, <id_3, 1, [6]>],$$
$$indexes\colon [<id_1, 1, [5]>, <id_2, 1, [2]>, <id_3, 1, [6]>],$$
$$information\colon\ [<id_2, 1, [5]>],$$
$$retrieval\colon\ [<id_2, 1, [7]>],$$
$$applications\colon\ [<id_3, 1, [1]>],$$
$$retrieve\colon\ [<id_3, 1, [3]],$$
$$documents\colon\ [<id_3, 1, [4]>]$$

Inverted Index

- Searching with inverted indexes:
  - Given a search terms query each term is searched in the index and a concatenated list of all the document unique identifiers without duplicates is returned
  - Example:
    - For the search query ‘applications information documents’ the following documents are going to be returned: $\{id_3, id_2\}$
    - If a ranking function is used then document d_3 will have a bigger weight that document d_2

Overview

- Information Retrieval
- Text Preprocessing
- Inverted Index
- Latent Semantic Indexing

Latent Semantic Indexing

- The retrieval models used so far are based on keyword or term matching, i.e. terms in the search query are matched with terms in the documents
- However, many concepts or objects can be described in multiple ways (synonyms), e.g. image, picture, photo
- The retrieval process can have a low recall if the search query contains a synonym that is not frequent in the corpus of documents

Latent Semantic Indexing

- Latent Semantic Indexing (LSI – also called Latent Semantic Analysis LSA) tries to solve the problem of synonyms by identifying terms that statistically appear together.
- It assumes that there are some underlying latent semantic structure in the data that is partially obscured by the randomness of word choice.
- It uses a statistical technique, called Singular Value Decomposition (SVD), to estimate this latent structure.
- It identifies syntactical different but semantically similar terms using a structure called hidden “concept” space

Latent Semantic Indexing

- Given the term-document matrix (A) with the size $m \times n$ (n is number of documents, m is the number of terms in the vocabulary)
- LSI uses SVD to factorize A into a product of three matrices:

$$A = U\Sigma V^T$$

Latent Semantic Indexing

$$A = U\Sigma V^T$$

- Where
  - U
    - Is a $m \times r$ matrix and its columns, called left singular values, are eigenvectors associated with r non-zero eigenvalues of AA^T.
    - The columns of U are unit orthogonal vectors, i.e. $U^T U = I$
  - V
    - Is an $n \times r$ matrix and its columns, called right singular vectors, are eigenvectors associated with the r non-zero eigenvalues of $A^T A$.
    - The columns of V are also unit orthogonal vectors, i.e., $V^T V = I$.
  - Σ is a $r \times r$ diagonal matrix, $\Sigma = diag(\sigma_1, \sigma_2, \dots, \sigma_r), \sigma_i > 0$. The diagonal values σ_i
    - Are called singular values
    - Are non-negative square roots of r non-zero eigenvalues of AA^T.
    - They are arranged in decreasing order, i.e. $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$

Latent Semantic Indexing

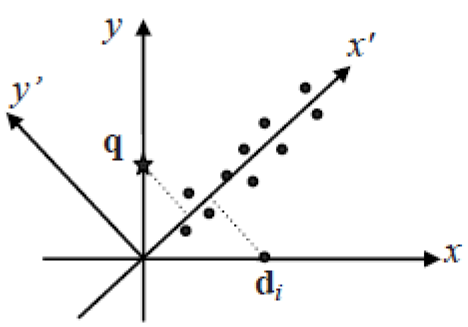
- Notes:

1. The initial U is an $m \times m$ matrix, V is an $n \times n$ matrix and Σ is an $m \times n$ matrix.
2. Σ's diagonal consists of non-negative eigenvalues of AA^T or $A^T A$.
   - However, due to zero eigenvalues, Σ has zero-valued rows and columns.
   - Matrix multiplication tells us that those zero-valued rows and columns from Σ can be dropped.
   - Then, the last $m - r$ columns in U and the last $n - r$ columns in V can also be dropped.
3. r is the rank of A, $r \leq min(m, n)$

Latent Semantic Indexing

- An eigenvectors is a non-negative vector whose direction does not change when a linear transformation is applied to it:
$$T(v) = \lambda v$$
- Where
  - $T(\cdot)$ is a linear transformation
  - λ is the eigenvalue
- For a matrix A, the eigenvectors and eigenvalue gives us the following property:
$$Av = \lambda v \text{ or } (A - \lambda I)v = 0$$
- To compute the eigenvalues, we must solve the linear system $\det(A - \lambda I) = 0$

- **Intuitive Idea of LSI:**
  - The intuition of LSI is that SVD rotates the axes of n-dimensional space of $\boldsymbol{A}$ such that
    - The first axis runs along the largest variation (variance) of terms among the documents
    - The second axis runs along the second largest variation (variance) of term
    - And so on



Summary

- This course presented:
  - IR architecture
  - IR query types
  - Document representation
  - Weighting schemas
  - Text preprocessing
  - Inverted Indexes
  - LSI

References

- [Liu 2011] Bing Liu: *Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data*, 2011