University of Technology
الجامعة التكنولوجية

Computer Science Department
قسم علوم الحاسوب

Information Retrieval Techniques
تقنيات استرجاع المعلومات

Teaching by: Dr. Khalil I. Ghathwan,
أ.م.د. خليل ابراهيم غثوان
Prepared by: Dr. Mustafa Jasim
أ.م.د. مصطفى جاسم هادي

cs.uotechnology.edu.iq

What is information retrieval?

Information retrieval (IR) deals with the representation, storage, organization of, and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he is interested.

Unfortunately, characterization of the user information need is not a simple problem. Information retrieval (IR) has changed considerably in recent years with the expansion of the World Wide Web (WWW) and the advent of modern and inexpensive graphical user interfaces and mass storage devices. As a result, traditional IR textbooks have document quite out of data and this has led to the introduction of new IR books.

What are the applications of IR?

There are many applications of information retrieval, they include:

1-Digital libraries: is an online database of digital objects that can include text, still images, audio, video, or other digital media formats. Objects can consist of digitized content like print or photographs, as well as originally produced digital content like word processor files or social media posts. In addition to storing content, digital libraries provide means for organizing, searching, and retrieving the content contained in the collection.

2-A recommender system: is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an

item. They are primarily used in commercial applications. Recommender systems have been developed to explore research articles and experts, collaborators, financial services, and life insurance.

3-A media search that includes:

a) *An image retrieval system*: is a computer system for browsing, searching and retrieving images from a large database of digital images.

b) *A music information retrieval (MIR)*: it is the interdisciplinary science of retrieving information from music. MIR is a small but growing field of research with many real-world applications.

c) *A video retrieval*: it is concerned with how to return similar video clips (frames) to a user given a video query. There are two major categories of existing work. One is to first extract key frames from the video data, then use image retrieval techniques to obtain the video data indirectly. The other technique incorporates motion information (sometimes object tracking) into the retrieval process.

H.W1: Try to support the above <u>media search</u> types with report gives more details with figures illustrate their architectures.

4- The search engines that include:

a) *Desktop search*: it's a tool designed to find information on the user's PC, including web browser history, e-mail archives, text documents, sound files, images, and video.

b) *Web search:* is a software system that is designed to carry out web search (Internet search), which means to search the World Wide Web in a systematic way for particular information specified in a textual web search query.

c) *Social search*: is a behavior of retrieving and searching on a social searching engine that mainly searches user-generated content such as news, videos and images related search queries on social media like Facebook, Twitter, Instagram, and LinkedIn.

d) *Mobile search*: is an evolving branch of information retrieval services that is centered on the convergence of mobile platforms and mobile phones.

H.W2: Try to support the above search engines types with report gives more details with figures illustrate their architectures.

Information versus Data Retrieval

Data retrieval, in the context of an IR system, consists mainly of determining which documents of a collection contain the keywords in the user query which, most frequently, is not enough to satisfy the user information need. In fact, the user of an IR system is concerned more with retrieving information about a subject than with retrieving data which satisfies a given query.

A data retrieval language aims at retrieving all objects which satisfy clearly defined conditions such as those in a regular expression or in a relational algebra expression. Thus, for a data retrieval system, a single erroneous object among a thousand retrieved objects means total failure.

For an information retrieval system, the retrieved objects might be inaccurate and small errors are likely to go unnoticed. The main reason for this difference is that information retrieval usually deals with natural language text which is not always well structured and could be semantically ambiguous. On the other hand, a data retrieval system (such as a relational database) deals with data that has a well-defined structure and semantics.

Data retrieval, while providing a solution to the user of a database system, does not solve the problem of retrieving information about a subject or topic. To be effective in its attempt to satisfy the user information need, the IR system must somehow 'interpret' the contents of the information items (documents) in a collection and rank them according to a degree of relevance to the user query. This 'interpretation' of a document content involves extracting syntactic and semantic information from the document text and using this information to match the user information need.

The difficulty is not only knowing how to extract this information but also knowing how to use it to decide relevance. Thus, the notion of relevance is at the center of information retrieval. In fact, the primary goal of an IR

system is to retrieve all the documents which are relevant to a user query while retrieving as few non-relevant documents as possible.

H.W3: Compare between information retrieval and data retrieval. Explain the comparison in a table.

What is the research area in information retrieval?

The area of information retrieval has grown well beyond its primary goals of indexing text and searching for useful documents in a collection. The **research area in IR** includes:

1. Modeling.
2. document classification and categorization.
3. systems architecture.
4. user interfaces.
5. data visualization.
6. filtering.

The Web is becoming a universal repository of human knowledge and culture which has allowed unprecedented sharing of ideas and information in a scale never seen before. It is causing a revolution in the way people use computers and perform their daily tasks. For instance, home shopping and home banking are becoming very popular and have generated several hundred million dollars in revenues. Despite so much success, the Web has introduced new problems of its own.

Finding useful information on the Web is frequently a tedious and difficult task. For instance, to satisfy his information need, the user might

navigate the space of Web links (i.e., the *hyperspace)* searching for information of interest. However, since the hyperspace is vast and almost unknown, such a navigation task is usually inefficient. For naive users, the problem becomes harder, which might entirely frustrate all their efforts. The main obstacle is the absence of a well-defined underlying data model for the Web, which implies that information definition and structure is frequently of low quality. These difficulties have attracted renewed interest in IR and its techniques as promising solutions. As a result, almost overnight, IR has gained a place with other technologies at the center of the stage.

IR in Past and Present

For approximately 4000 years, man has organized information for later retrieval and usage. A typical example is the table of contents of a book. Since the volume of information eventually grew beyond a few books, it became necessary to build specialized data structures to ensure faster access to the stored information.

An old and popular data structure for faster information retrieval is a collection of selected words or concepts with which are associated pointers to the related information (or documents) - the *index.* The indexes are at the core of every modern information retrieval system. They provide faster access to the data and allow the query processing task to be speeded up.

More recently, the advent of modern computers has made possible the construction of large indexes automatically. Automatic indexes provide a view of the retrieval problem which is much more related to the system itself than to the user need. In this respect, it is important to distinguish between two different views of the IR problem: *a computer-centered one* and *a human-centered one*.

What are the two different views of the IR problem?

1- In the **computer-centered view**, the IR problem consists mainly of building up efficient indexes, processing user queries with high performance, and developing ranking algorithms which improve the 'quality' of the answer set.

2- In the **human-centered view**, the IR problem consists mainly of studying the behavior of the user, of understanding his main needs, and of determining how such understanding affects the organization and operation of the retrieval system. According to this view, keyword-based query processing might be seen as a strategy which is unlikely to yield a good solution to the information retrieval problem in the long run.

What are the IR system components?

Information retrieval system is a system used to store items of information that need to be processed, searched and retrieved corresponding to a

user's query. It is basically constituted by three main components, whose composition is introduced as follows:

1- *The documentary database*: This component stores the documents and the representations of their information contents. It is associated with the indexer module, which automatically generates a representation for each document by extracting the document contents. Textual document representation is typically based on index terms (that can be either single terms or sequences), which are the content identifiers of the documents.
2- *The matching mechanism*: It evaluates the degree to the document, which representations satisfy the requirements expressed in the query, the retrieval status value and retrieves those documents that are judged to be relevant to it.
3- *The query subsystem*: It allows the users to formulate their information needs and presents the relevant documents retrieved by the system to them. To do that, it includes a query language that collects the rules to generate legitimate queries and procedures to select the relevant documents.

H.W4: Try to support the above IR system components with report gives more details with figure illustrate the architecture.

Information Retrieval and Web Search

Information Retrieval (IR) is generally broader than modern Web Search; the aim of which is usually to satisfy a user's need as quickly as possibly (where users are typically not interested in all relevant documents).

Current *Web search engines* are a product of continued research in traditional IR and remain a vibrant field today. Given an information need expressed as a short query consisting of a few terms, the system's task is to retrieve relevant Web objects and present them to the user.

There are basically three different forms of searching the Web:

1- The first is to use search engines that index a portion of the Web documents as a full-text database.

2- The second is to use Web directories, which classify selected Web documents by subject.

3- The third is to search the Web exploiting its hyperlink structure**.**

Nearly all retrieval engines for full-text search today rely on a data structure called an *inverted index*, which given a term provides access to the list of documents that contain the term.

In information retrieval parlance, objects to be retrieved are generically called "documents" even though in actuality they may be Web pages, PDFs, or even fragments of code. Given a user query, the retrieval engine uses the inverted index to score documents that contain the query terms with respect to some ranking model, taking into account features such as term matches, term proximity, attributes of the terms in the document (such as the bold or appearing in the title), as well as the hyperlink structure of the documents.

Web search is a complex problem that breaks down into three conceptually distinct components.

1- First, the documents collection must be gathered (by crawling the Web).
2- Next, inverted indexes and other auxiliary data structures must be built from the documents. Both of these can be considered offline problems.
3- Finally, index structures must be accessed and processed in response to user queries to generate search results. This last task is an online problem that demands both low latency and high throughput.

The documents collection resource

Before building inverted indexes, one must first acquire the documents collection over which the indexes are to be built. For real-world Web search, one cannot simply assume that the collection is already available. Acquiring Web content requires crawling, which is the process of traversing the Web by repeatedly following hyperlinks and storing downloaded pages for subsequent processing. Conceptually, the process is quite simple to understand: the start is by populating a queue with a "seed" list of pages.

The crawler downloads pages in the queue, extracts links from those pages to add to the queue, stores the pages for further processing, and repeats. In fact, rudimentary Web crawlers can be written in a few

hundred lines of code. However, effective and efficient Web crawling is far more complex.

Extraction of index terms

Not all words are equally significant for representing the semantics of a document. In written language, some words carry more meaning than others. Usually, noun words or groups of noun words are the ones that are most representative of document content.

Therefore, it is usually considered worthwhile to preprocess the text of the documents in the collection to determine the terms to be used as index terms.

Text preprocessing procedure

Text preprocessing is a procedure which can be divided mainly into four text operations or transformations**:**

1- Lexical analysis of the text with the objective of treating digits, hyphens, punctuation marks, case of letters, and the white spaces. For example,

Input: "*They have the best bookstore online—shopping over 4.5 million books!*"

Output: [*they, have, the, best, bookstore, online, shopping, over, million, books*]

2- Elimination of stopwords with the objective of filtering out words with very low discrimination values for retrieval purposes. For example,

   Input: [*they, have, the, best, bookstore, online, shopping, over, million, books*]

   In the above input, the stopwords are: *['they', 'have', 'the', 'over']*

   Output: [*best, bookstore, online, shopping, million, books]*

3- Stemming of the remaining words with the objective of removing affixes (i.e. prefixes and suffixes) and allowing the retrieval of documents containing syntactic variations of query terms (e.g., connect, connecting, connected, etc). For example,

   Input: [*best, bookstore, online, shopping, million, books]*

   Output: *[best, bookstore, onlin, shop, million, book]*

Selection of index terms to determine which words/stems or groups of words will be used as indexing elements. In fact, noun words frequently carry more semantics than adjectives, adverbs, and verbs.

What are the stopwords?

The stop words refer to every word that's empty of true meaning given a context. They have high frequency on the documents collection.

For example, the stopwords in English language are: *['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is',*

'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']

H.W5: What are the stopwords in <u>Arabic</u> language?

Why we remove the stopwords in IR?

Words such as articles and some verbs are usually considered stop words because they don't help us to find the context or the true meaning of a sentence. These are words that can be removed without any negative consequences to the final model that you are training.

What is the stemming?

The stemming is kind of morphological analysis that concerns with studying the internal structure of words.

Why we need the stemmers in IR?

In IR, the stemmers are stemming programs that treat the indexed words in a way that reduces the storage size and increases the matching chance. The process of reducing inflected words to their stem or root form along with the converting of all words to lower case is believed that they increase the recall rate.

H.W6: Try to do a report gives more details about the available standard English and Arabic stemmers.

What is the inverted index?

One obvious way to design an algorithm for IR is to browse the whole collection of documents and calculate the similarity between the document and the query. This refers to the exhaustive search and it is suitable only when the collection is small or is highly dynamic relative to the query rate. In contrast, the fast query evaluation makes use of an index: a data structure that maps terms to the documents that contain them. For example, the index of a book maps a set of selected terms to page numbers.

Once all needed preprocessing has been performed, the (text) document set can be indexed in an inverted index that supports efficient answering to term queries. i.e. the information found in the inverted index is used by the system to process search queries.

What are the inverted index components?

The inverted index consists of two major components, the dictionary and the inverted lists.

1- The dictionary stores for each distinct term *t*,

a) a count df_t (*document frequency)* of the documents containing *t*. i.e., the number of documents to which a term is assigned.

b) a pointer to the start of the corresponding inverted list.

2- The inverted lists store for the corresponding term *t*,

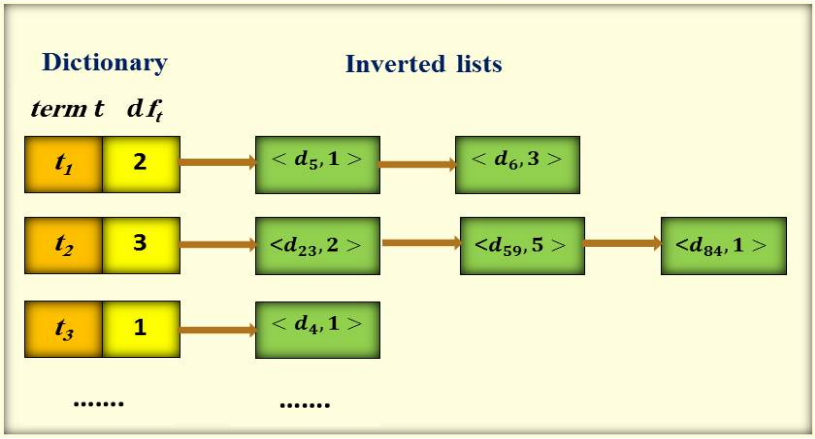
a) the identifiers *d* of documents containing *t*, represented as ordinal document numbers.

b) the associated set of frequencies $tf_{d,t}$ (*term frequency*) of terms *t* in a document *d*.

The lists are represented as sequences of $\langle d, tf_{d,t} \rangle$ pairs.

How the inverted index is created? An Example

Each term t is coupled with the document frequency of term t, and associated with an inverted list for *t*. The inverted list is comprised of individual postings, each of which consists of a document identifier *d* and the term frequency of a term t in the document *d*.



The figure shows the following: Term t_1 occurs in only two documents $\{d_5, d_6\}$ therefore df_{t_1}=2,the occurrence in d_5 just once, tf_{d_5,t_1}=1 while in $d_6$isthreetimes,tf_{d_6,t_1}=3.Termt_2 occurs in three documents $\{d_{23}, d_{59}, d_{84}\}$ therefore df_{t_2}=3.The occurrence in $d_{23}$is twice, tf_{d_{23},t_2}=2 and in d_{59} is five times,tf_{d_{59},t_2}=5, and in d_{84} is once,tf_{d_{84},t_2}=1.

Term$t_3$appears in one document $\{d_4\}$ therefore df_{t_3}=1 and the term appears once, tf_{d_4,t_3}=1.

Ranking process

When the user gives a query, the index is consulted to get the document most relevant to the query. For example, using "*Web Information Retrieval*" as the query, the search engine Google estimated that there were 46,500,000 relevant pages.

One central problem regarding information retrieval systems is the issue of predicting which documents are relevant and which are not. Such a decision is usually dependent on a ranking algorithm which attempts to establish a simple ordering of the documents retrieved. Documents appearing at the top of this ordering are considered to be more likely to be relevant. However, there are many issues that affect ranking the Web such as:

1- Quality of the pages, anyone can publish anything, so there is no quality control.

2- Duplicate content, mainly due to the mirror, meaning that identical documents appear on the Web with different URLs.

3- Spam, Spamming refers to actions that do no increase the information value of a page, but dramatically increase its rank position by misleading search algorithm to rank it high than they deserve.

Vector space model

The vector-based document retrieval system ,that allows a query to be expressed as a natural language text describing the user's information need, is based on the vector space model and transforms the description of information problems as well as the stored documents into the vectors of the form:

$$d_i = (w_{i1}, w_{i2}, \ldots, w_{in})$$

where d_i represents a document (or query) text and w_{ik} represents the weight of term t_k in d_i.

A weight of zero is used for the terms that are absent from a particular document, and positive weights characterize the terms actually assigned for the contents identification.

Once term vectors are available for all information items, all subsequent processing is based on the term vector manipulations. When document *d* is represented by a vector of the form $< w_{d,1}, w_{d,2}, \ldots, w_{d,n} >$ and query q by the vector $< w_{q,1}, w_{q,2}, \ldots, w_{q,n} >$.

H.W7: Try to search the Internet about the other standard document retrieval models, give more details about them.

Query-document similarity

The similarity between document *d* and query q is calculated as the inner product between the corresponding weighted term vectors as follows:

$$Similarity(q,d) = \sum_{t=1}^{n} w_{q,t} . w_{d,t}$$

The similarity between a query and the documents can be computed in order to rank the retrieved documents in descending order of the query-document similarity. The query-document similarity depends on the weights of coinciding terms in the two vectors, and therefore the term weighting scheme is an important factor affecting the effectiveness of the vector space model.

Weighting scheme

In constructing a term weighting scheme, three main components such as the *term frequency*, the *collection frequency*, and the *normalization* have been considered in the information literature .

What is the term frequency component? The term frequency component assigns higher weights to the terms that occur more frequently in the text.

What is the *collection frequency* component? The *collection frequency* component assigns higher weights to the terms that occur in fewer documents of the collection.

What is the *normalization* component? The *normalization* component equalizes the length of the document vectors in the collections of varying document vector length.

Best fully weighting system

There are many term weighting schemes mentioned in IR literature. A typical complex term weighting scheme called "best fully weighting system" uses a cosine normalized $tf \times idf$ weight (term frequency times inverse document frequency) for the document terms, and an enhanced but unnormalized$tf \times idf$ factor for the queries.

For the document terms, the term frequency component uses tf. This is the number of time say term occurs in a document or the query text. The inverse document frequency component idf_tuses $\log(N/df_t)$, where N is the total number of the documents in a collection, and df_t is the document frequency of term t (the number of documents to which a term is assigned).

Normalization component uses$1/\sqrt{\sum_{vector} w_t^2}$. This is the cosine normalization, where each term weight w_t is divided by a factor representing Euclidean vector length. Therefore, the document term weight $w_{d,t}$ of best fully weighted system is as follows:

$$w_{d,t} = \frac{tf_{d,t} \times \log(N/df_t)}{\sqrt{\sum_{vector}\left(tf_{d,t} \times \log(N/df_t)\right)^2}}$$

For the query terms, the term frequency component uses:

$$\left(0.5 + 0.5\ \frac{tf_{q,t}}{\max\ tf_{q,t}}\right)$$

This is the augmented normalized term frequency, where the $tf_{q,t}$ factor isnormalized by maximum $tf_{q,t}$ in the vector. It is further normalized to lie between 0.5 and 1.0.

Normalization component uses 1.0, which is not applied normalization. Therefore, the query term weight $w_{q,t}$of best fully weighted system is as follows:

$$w_{q,t} = \left(0.5 + 0.5\ \frac{tf_{q,t}}{\max\ tf_{q,t}}\right) \times \log\left(N/df_t\right)$$

H.W8: Try to search the Internet about the other standard weighting schemes, give more details about them.

An Example for IR System Process

1) Text Preprocessing: Lexical Analysis, Stopword Removal, and Stemming.

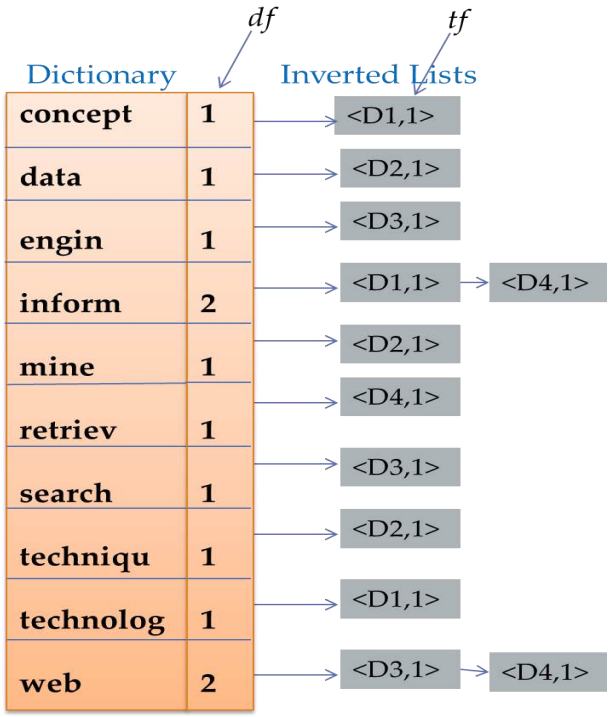
Example

D1: Concepts of information technology.
D2: Data mining techniques.
D3: Web and Search engines.
D4: Information retrieval on the web.

D1: concept inform technolog
D2: data mine techniqu
D3: web search engin
D4: inform retriev web

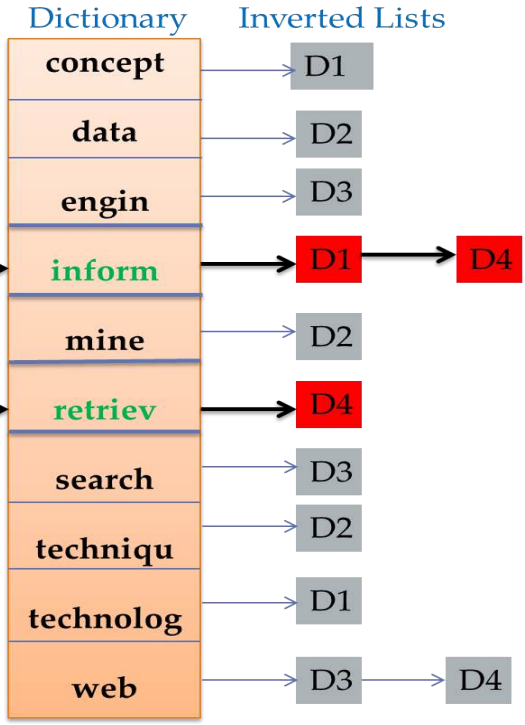
2) Indexing: Build the inverted index

Example:
D1: concept inform technolog
D2: data mine techniqu
D3: web search engin
D4: inform retriev web



3) Searching: Matching Query-Documents.

Q: Information retrieval



inform={D1,D4}
retriev={D4}

Result is two docs →{D1,D4}

What is the best D1 or D4?

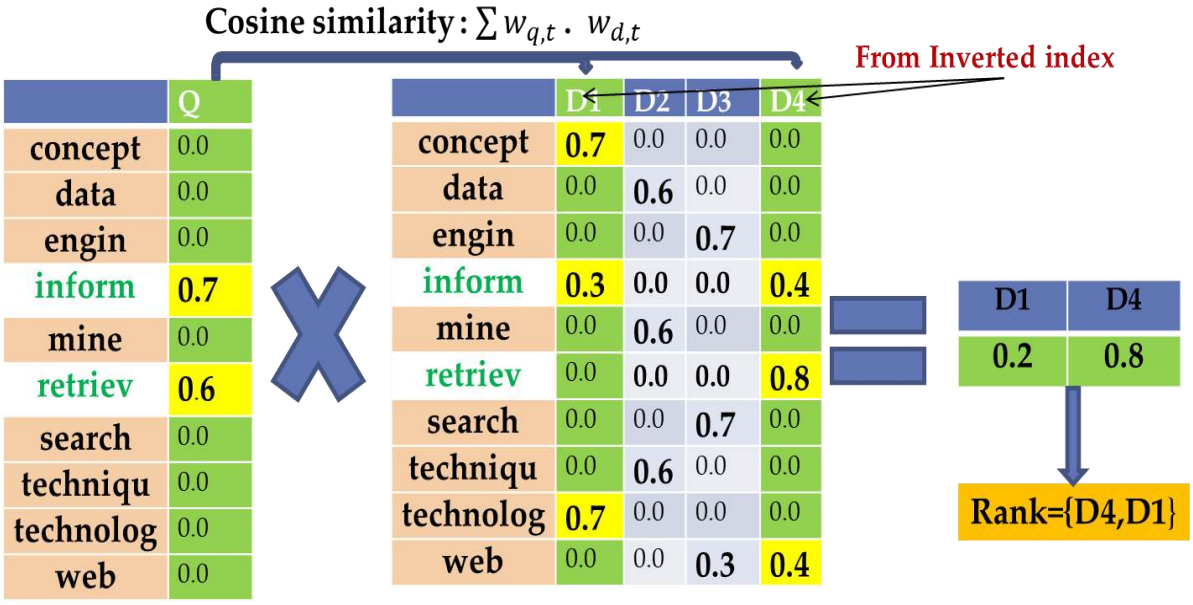
4) Ranking (scores all retrieved documents):

1-Assignment term weighting (tf * idf)

2-Build retrieval model (VSM)

3-Calculate Query-Docs similarity (Cosine)

Example:
Q: Information retrieval
D1: Concepts of information technology
D2: Data mining techniques.
D3: Web and Search engines.
D4: Information retrieval on the web.



Conclusion: *The document D4 is better than the document D1 with corresponding to the given query.*

System performance evaluation

Evaluation is the key to making progress in building better search engines. It is also essential to understanding if a search engine is being used effectively in a specific application. There are two principal aspects to measuring IR system performance: *efficiency* and *effectiveness*.

What are the two principal aspects to measuring IR system?

1- *Effectiveness* measures the ability of the search engine to find the right information and efficiency measures how quickly this is done. For a given query, and a specific definition of relevance, one can more precisely define effectiveness as a measure of how well the ranking produced by the search engine corresponds to a ranking based on user relevance judgments.

2- *Efficiency* is defined in terms of the time and space requirements for the algorithm that produces the ranking. Viewed more generally, however, search is an interactive process involving different types of users with different information problems.

What are the efficiency measures?

When fine-tuning the performance of a search engine, one often finds oneself in a situation in which there are two competing implementations (both producing the same search results), and he/she would like to know which one is better. The two most common performance measures that can help answer this question are *throughput* and *latency*.

1- *Throughput* refers to the number of queries a search engine processes within a given period of time. When one refers to the throughput it usually means theoretical throughput, also known as service rate, which is the fastest possible rate at which the system can process queries. When many simultaneous users must be supported, the query throughput, measured in queries per second, becomes an important factor in system performance. For a general-purpose Web search engine, the required throughput may range well beyond tens of thousands of queries per second.

2- *Latency*, also known as response time, is the most visible aspect of efficiency and is the amount of time that elapses from the search engine's receipt of the query until the results are sent to the user. It is measured in seconds per query. Latency can be measured on the server side or on the client side. The latter is referred to as end-to-end latency; it is the time between the user issuing the query and his/her receiving the search results. It includes network latency and is more indicative of user satisfaction than the search engine's query processing latency alone. However, when conducting comparative performance evaluations, it is usually sufficient to focus on the actual search latency, because the extra latency caused by the network is a constant that is independent of the implementation.

What are the classical IR techniques?

The principle in the classical IR techniques is to search in the corresponding index for documents that are similar with the query. The consulted documents are sorted according to their similarity with the query. The classical search techniques for information retrieval (IR) are:

1- *Forward index-based search.*

2- *Inverted index-based search.*

In the *forward index-based search*, we design an algorithm for browse the whole collection of documents and calculating the similarity between the document and the query. Generally, this algorithm is not considered even for reasonable size of corpuses because there is a smarter way to address this problem.

In the *inverted index-based search*, the idea is to execute the algorithm on the inverted index instead of the whole collection of documents. Only documents that have at least one common term with the query are consulted and this way the complexity of the algorithm is reduced at a phenomenal rate. However, in the Web context, the inverted index remains scalable by containing several millions and more documents and may become untreatable.

What are the modern IR techniques?

The classic search by keywords abstract from the meanings (senses) can rank some of results that don't satisfy the user's need. This makes users spend much time to organize and find the satisfied search results. Query expansion (QE) is a successful idea to overcome the weaknesses in the classic search. The QE requires finding out appropriate word synonyms of the query words in a process that can be made automatically without any user intervention. The candidate synonyms should be associated with an accurate sense of the original word. The query expansion techniques include:

1- *Using the Extended Lesk technique*: The focus on analyzing the implementation of query expansion using thesaurus and pseudo relevance feedback. The Extended Lesk algorithm was used to disambiguate the query using a lexical resource called WordNet. However, the Extended Lesk algorithm needs a long process time and the results are not good enough for topic specialized documents. Relevance feedback using the corpus' information plays a big role in the query expansion and gives a bigger increasing in performance than using the WordNet alone.

2- *Using the relatedness technique*: The use of knowledge-based semantic relatedness technique to overcome the vocabulary mismatch between the query and documents. The query expansion and document expansion are performed using the WordNet lexical resource. The analysis shows that the pseudo-relevance feedback and the knowledge-based semantic relatedness technique are complementary; in that, the first is better for easy queries, and the letter is stronger for difficult queries.

3- *Using the Expectation Maximization techniques*: Expectation Maximization (EM) is used to indicate similarity between two words based on their co-occurrence in a set of documents. Expanding queries in their work consist of three steps: Extracting top 10 documents, extracting top 100 keywords out of the top 10 documents, and eliminating irrelevant keywords using EM distance. The remaining words are then added to the original query to construct the expanded version of the query.

4- *WordNet-Based Semantic technique:* The query expansion can be performed by using semantic measures based on WordNet. These measures have been grouped into four classes: path *length-based measures*, *information content-based measures*, *feature-based measures*, and *hybrid measures*.

The other modern IR techniques can be listed in other concepts such as *the distributed information retrieval* and *the clustering.*

H.W9: Try to search the Internet to check whether the existence of the Arabic WordNet is completed? if it is completely present! give more details about it.

Distributed Information Retrieval (DIR)

Distributed Information Retrieval (DIR) is a model in the modern IR that enables a user to access many searchable documents reside in different locations (servers), where each location contains a collection of documents. There are two basic strategies are defined to distribute the inverted index over a set of servers:

1- The *global* inverted index: in the global inverted index strategy, each server stores inverted lists corresponding to only a subset of the index terms within the inverted index of all collections.

2- The *local* inverted index: in the local inverted index strategy, each server stores inverted lists within an inverted index corresponding to an individual collection.

Why DIR is more complex than the centralized IR?

DIR is more complex than the centralized IR because it requires addressing two significant additional problems that are:

1- The resource selection (RS): it refers to the decision to select the most appropriate servers to search.

2- The results merging (RM): it refers to unify the ranked lists returned by the selected servers into a single ranked list.

What is the RS technique in the modern IR?

To select the resources, the document collection in a server is considered as if it is a single giant document. Ranking the servers is similar to document ranking approaches used in the centralized IR systems.

Let Q is a given query, S_i is a server stores a document collection, and t_j is a term occurs in S_i documents. The local servers are ranked based on the probability $p(Q|S_i)$ that refers to the belief that a query Q is satisfied by a local server S_i. The belief $p(t_j|S_i)$ indicates that the search term t_j contributes in finding the score of server S_i and it is computed as follows:

$$p(t_j|S_i) = defB + (1 - defB) \times R \times I$$

Where:

$$R = \frac{df}{df + k \times \left[(1-b) + b \times \frac{nt}{avg_{nt}}\right]}$$

$$I = \frac{log\ (\frac{|C| + 0.5}{sf})}{log\ (|C| + 1.0)}$$

Where df is the number of documents that contain term t_j within the server S_i, nt is the number of terms in S_i, avg_nt is the average of nt, sf is the number of servers that contain term t_j, $|C|$ is the total number of servers. *defB*, *b* and *k* are constants. Callan suggested to set these constants at the following values: *defB* = 0.4, *k* = 200 and b = 0.75. Finally, the belief $p\ (Q|S_i)$ for the total query terms is computed as follows:

$$p\ (Q|S_i) = \frac{\Sigma_{t_j \in Q}\ p\left(t_j \middle| S_i\right)}{|Q|}$$

What is the RM technique in the modern IR?

RM technique combines both the original document score and the selected document collection score. This method uses a simple heuristic to normalize resultant scores. The normalization of the collection and document scores is shown as follows:

$$S_i' = \frac{S_i^s - S_{min}^s}{S_{max}^s - S_{min}^s}$$

$$D' = \frac{D^s - D_{min}^s}{D_{max}^s - D_{min}^s}$$

$$D'' = \frac{\alpha \times D' + \alpha \times D' \times \beta \times S_i'}{\alpha + \beta}$$

Where S_i^s is the score of a selected local server, S_{min}^s and S_{max}^s are the minimum and maximum scores of the selected servers, S_i' is the normalized score of the local server. D^s is the original relevance score of a document given by the local server, D_{min}^s and D_{max}^s are the minimum and maximum document scores assigned by the local server. D' is the normalized document score. Finally, D'' is the final

document score that is normalized between *0* and *1*. It uses heuristic weighting schemes, such as a weight $\alpha = 1$ for the normalized document score and a weight $\beta = 0.4$ for the normalized local server score. The constant β also shows how much importance is given to resource selection scores. If $\beta = 0$ then there is no importance is given to the resource selection scores.

H.W10: Try to search the Internet about the other standard RS & RM techniques, give more details about them.

Clustering in information retrieval

The clustering in information retrieval means that the documents in the same cluster behave similarly with respect to relevance to information needs.

For example, if there is a document from a cluster that is relevant to a search request, then it is likely that other documents from the same cluster are also relevant. This is because clustering puts together documents that share many terms.

Why we need to document clustering?

Clustering the documents with similar topics is a key concept to cope with huge collections of text documents such as the Internet, news repositories, or digital libraries. It can serve different purposes:

1- Increasing the performance of the retrieval process. The respective technology is often called cluster- based search and can improve both the efficiency and the effectiveness of full search.

2- Improving the user interface for browsing document sources. In particular, clustering can help to navigate, inspect, and organize document collections.

3- Automatic text generation. A high-quality document clustering can form the basis for a further processing like summarization, or as a knowledge base for reasoning and documentation systems.

Clustering exploits knowledge about the similarity among the objects to be clustered. The similarity of two documents, *d1*,*d2*, is computed as a function of the distance between the corresponding term vectors *d1* and *d2*. Various measures exist for similarity computation, from which the cosine-measure proved to be the most successful for document comparison.

It is defined as follows.

$$\cos(d1, d2) = \frac{\overrightarrow{d1}.\overrightarrow{d2}}{|\overrightarrow{d1}|.|d2|} = \frac{\sum_{t=1}^{n} w_t^{d1}.w_t^{d2}}{\sqrt{\sum_{t=1}^{n}(w_t^{d1})^2}.\sqrt{\sum_{t=1}^{n}(w_t^{d2})^2}}$$

Flat Clustering

Flat clustering gives you a single grouping or partitioning of data. These require you to have a prior understanding of the clusters as we have to set the resolution parameter. Examples of flat clustering algorithms are K-means, K- medoids, DBSCAN, etc.

Hierarchical Clustering

Hierarchical clustering gives you a sort of nested relationship between the data. It doesn't require you to have prior knowledge of the cluster as it creates a kind of natural hierarchy over the clusters. These algorithms assume each point as a cluster to group every point in a single cluster. A dendrogram generally represents this hierarchy. An example of these algorithms is the linkage-based algorithms.

The k–means clustering

1) What are the main features of k–means clustering?

- One of the simplest unsupervised learning algorithms
- Works with numeric data only
- Number of clusters fixed a priori
- The objective it tries to achieve is to minimize total intra-cluster variance.

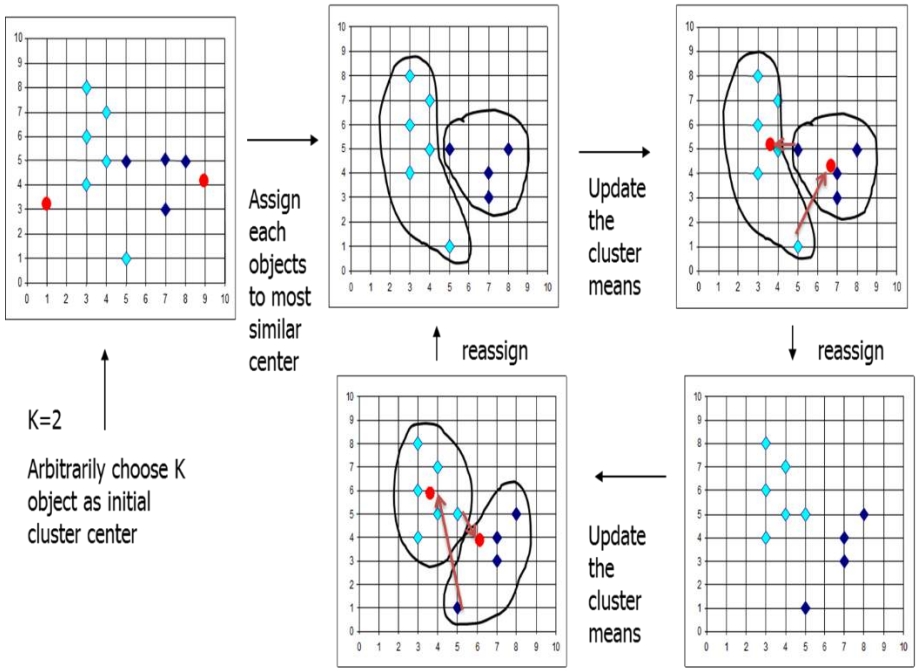
2) What is the pseudo code of the k–means clustering algorithm?

- ***Initialization***
  - ***Arbitrarily choose k objects as the initial cluster centers (centroids)***
- ***Iteration until no change***
  - ***For each object O_i***
    - ***Calculate the distances between O_i and the k centroids***

- ***(Re)assign O_i to the cluster whose centroid is the closest to O_i***

– ***Update the cluster centroids based on current assignment***

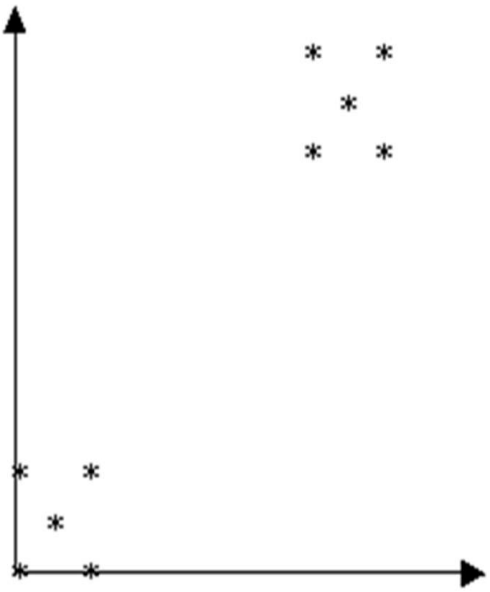
3) How does k–means work?



4) K-means example:

Give data objects plotted in the figure below . Choose $k = 2$ and use Manhattan distance function *dis* = $|x_2-x_1|+|y_2-y_1|$ for clustering.

$$O = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0.5 & 0.5 \\ 5 & 5 \\ 5 & 6 \\ 6 & 6 \\ 6 & 5 \\ 5.5 & 5.5 \end{bmatrix}$$



Step 1: Chose randomly *k* clusters (2 objects)

$$O = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0.5 & 0.5 \\ 5 & 5 \\ 5 & 6 \\ 6 & 6 \\ 6 & 5 \\ 5.5 & 5.5 \end{bmatrix}$$

Let the first two objects are the seed points of the clusters: clusters 1={(0,0)} and clusters 2={(0,1)}.

And represent the centroids points at the same time.

$C_1 = (0,0)$ and $C_2 = (0,1)$.

Step 2, Iteration-1:

Calculate the distance between each object and each cluster center, assigning the object to the closest cluster.

Using $dis = |x_2-x_1|+|y_2-y_1|$

$dis(O3, C_1) = |1-0| + |1-0| = 2$

$dis(O3, C_2) = |1-0| + |1-1| = 1$ less distance

so this object (1,1) is assigned to $Cluster_2$.

After calculating the distance for all points. the clusters contain the following objects:

$Cluster_1$ = {(O1,O4,O5} and

$Cluster_2$ = {O2,O3,O6,O7,O8,O9,O10}.

Update Centroids

Step 2, Iteration-1:

$$O = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0.5 & 0.5 \\ 5 & 5 \\ 5 & 6 \\ 6 & 6 \\ 6 & 5 \\ 5.5 & 5.5 \end{bmatrix}$$

$Cluster_1$ = {(O1,O4,O5}

$Cluster_2$ = {O2,O3,O6,O7,O8,O9,O10}.

Compute new cluster center for each cluster:

New center for $Cluster_1$ = {(O1,O4,O5} is

(0+1+0.5)/3 = 0.5 , (0+0+0.5)/3 = 0.16 ⟶ $C_1 = (0.5,0.16)$

New center for $Cluster_2$ = {O2,O3,O6,O7,O8,O9,O10}.

(0+1+5+5+6+6+5.5)/7 = 4.1

(1+1+5+5+6+6+5.5)/7 = 4.2

⟶ $C_2 = (4.1,4.2)$

$$O = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0.5 & 0.5 \\ 5 & 5 \\ 5 & 6 \\ 6 & 6 \\ 6 & 5 \\ 5.5 & 5.5 \end{bmatrix}$$

Go to step 2, Iteration -2:

- C_1 = (0.5,0.16)
- C_2 = (4.1,4.2)

New centers C_1 = (0.5,0.16) and C_2 = (4.1,4.2) differ from old centers C_1= (0,0) and C_2= (0,1),

so the loop is repeated. We get new clusters.

$Cluster_1$ = {O1,O2,O3,O4,O5}

$Cluster_2$ = {O6,O7,O8,O9,O10}

Compute new cluster center for each cluster

New center for C_1 = (0.5,0.5)

New center for C_2 = (5.5,5.5)

$$O = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0.5 & 0.5 \\ 5 & 5 \\ 5 & 6 \\ 6 & 6 \\ 6 & 5 \\ 5.5 & 5.5 \end{bmatrix}$$

Go to step 2, Iteration -3:

- C1 = (0.5,0.5)
- C2 = (5.5,5.5)

New centers C_1 = (0.5,0.5) and C_2 = (5.5,5.5) differ from old centers C_1 = (0.5,0.16) and C_2 = (4.1,4.2),

so the loop is repeated.

$Cluster_1$ = {O1,O2,O3,O4,O5}

$Cluster_2$ = {O6,O7,O8,O9,O10}

Compute new cluster centers. Here the algorithm is done:

Centers are the same as in Iteration-2,

so algorithm is finished.

Result is same as in Iteration-2.

$Cluster_1$ = {O1,O2,O3,O4,O5}

$Cluster_2$ = {O6,O7,O8,O9,O10}

5) What are the advantages of k–means clustering?

- Easy to understand and implement.
- Efficient in processing large data sets.

6) What are the disadvantages of k–means clustering?

- Works only when mean is defined (what about categorical data?).
- Need to specify k clusters in advance.
- Unable to handle noisy data (too sensible to outliers).
- Results depend on the metric used to measure distances and on the value of k.

The k-medoids clustering

In the k-means algorithm described in the previous section, each cluster is represented by the mean of its vectors. In the k-medoids methods, discussed in this section, each cluster is represented by a vector selected among the elements of X, and we will refer to it as the medoid. Apart from its medoid, each cluster contains all vectors in X that:

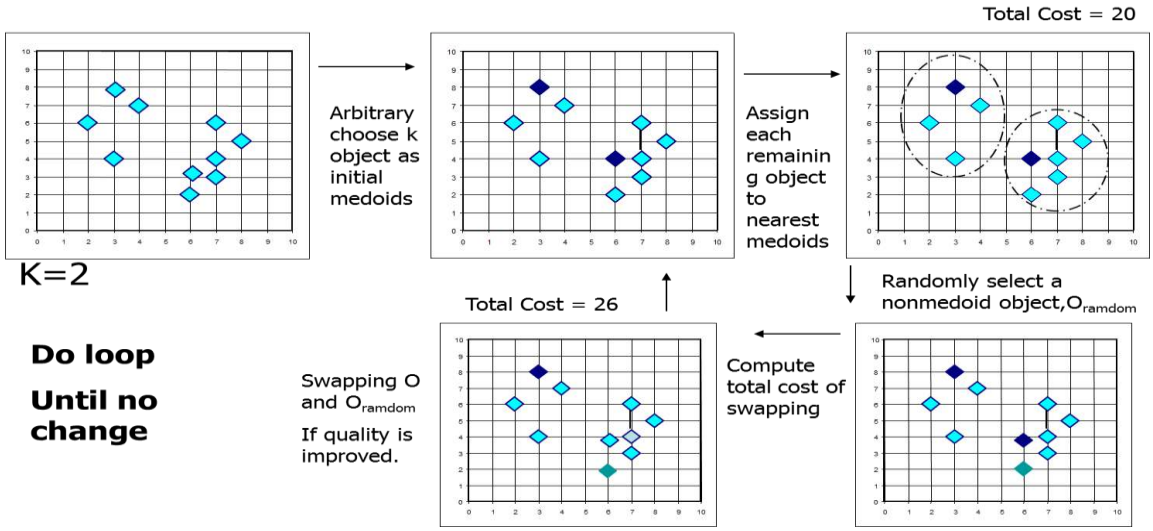
(a) are not used as medoids in other clusters and

(b) lie closer to its medoid than to the medoids representing the other clusters.

What is the pseudo code of the K-medoids algorithm?

- ***Initialization***
  - ***Arbitrarily choose k objects as the initial medoids***
- ***Iteration until no change, do***
  - ***(Re)assign each object to the cluster with the nearest medoid***
  - ***Improve the quality of the k-medoids (Randomly select a nonmedoid object, O_{random}, compute the total cost of swapping a medoid withO_{random})***

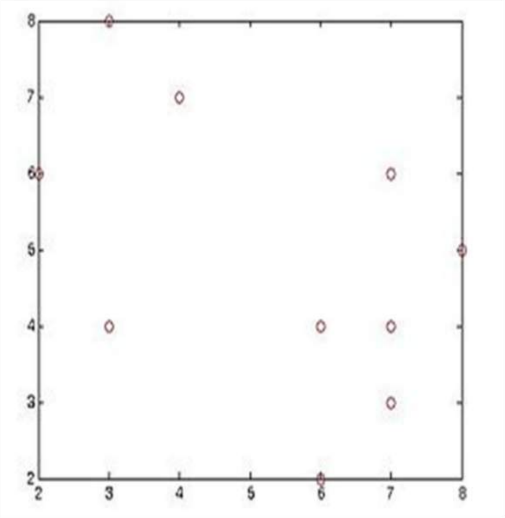
How does K-medoids work?



K-Medoids Example:

Cluster the following ten objects into two clusters i.e. k = 2. Use Manhattan distance to calculate the cost.

X_1	2	6
X_2	3	4
X_3	3	8
X_4	4	7
X_5	6	2
X_6	6	4
X_7	7	3
X_8	7	4
X_9	8	5
X_{10}	7	6



distribution of the data

X_1	2	6
X_2	3	4
X_3	3	8
X_4	4	7
X_5	6	2
X_6	6	4
X_7	7	3
X_8	7	4
X_9	8	5
X_{10}	7	6

Step 1:

Choose randomly two medoids :

Let X2 = (3,4) and X8 = (7,4).

Calculate distance so as to associate each data object to its nearest medoid.

Using Manhattan distance .

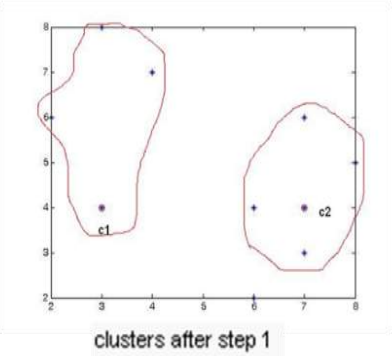
$$d(i,j)=|x_{i1}-x_{j1}|+|x_{i2}-x_{j2}|+\ldots+|x_{ip}-x_{jp}|$$

Costs to the nearest medoid are shown in the tables below.

C1=X2 = (3,4) and C2=X8 = (7,4).

i	c_1		Data objects (X_i)		Cost (distance)
1	3	4	2	6	3
3	3	4	3	8	4
4	3	4	4	7	4
5	3	4	6	2	5
6	3	4	6	4	3
7	3	4	7	3	5
9	3	4	8	5	6
10	3	4	7	6	6

dis(C1 ,X1) = |3-2| + |4-6| = 3



clusters after step 1

i	c_2		Data objects (X_i)		Cost (distance)
1	7	4	2	6	7
3	7	4	3	8	8
4	7	4	4	7	6
5	7	4	6	2	3
6	7	4	6	4	1
7	7	4	7	3	1
9	7	4	8	5	2
10	7	4	7	6	2

dis(C2,X1) = |7-2| + |4-6| = 7

i	Cost (distance)
1	3
3	4
4	4
5	5
6	3
7	5
9	6
10	6

C1 = (3,4)

i	Cost (distance)
1	7
3	8
4	6
5	3
6	1
7	1
9	2
10	2

C2= (7,4).

Since the points X1, X3 and X4 are closer to medoid C1=X2 = (3,4) and X5,X6,X7, X9,X10 are closer to medoid C2=X8 = (7,4) , so we form the following clusters.

$Cluster_1$ = {X1,*X2*,X3,X4} *with cost= 3+4+4*

$Cluster_2$ = {X5,X6,X7,*X8*,X9,X10} *with cost= 3+1+1+2+2*

Total cost=(*3+4+4)+(3+1+1+2+2)=20*

i	c_1		Data objects (X_i)		Cost (distance)
1	3	4	2	6	3
3	3	4	3	8	4
4	3	4	4	7	4
5	3	4	6	2	5
6	3	4	6	4	3
7	3	4	7	4	4
9	3	4	8	5	6
10	3	4	7	6	6

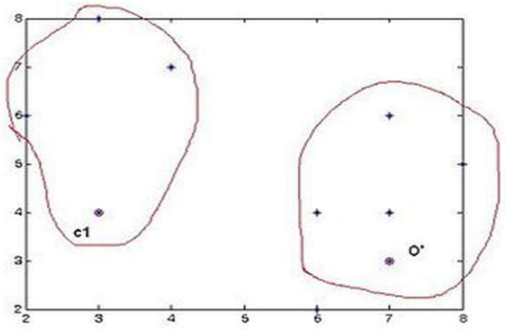
i	O'		Data objects (X_i)		Cost (distance)
1	7	3	2	6	8
3	7	3	3	8	9
4	7	3	4	7	7
5	7	3	6	2	2
6	7	3	6	4	2
7	7	3	7	4	1
9	7	3	8	5	3
10	7	3	7	6	3

Previous clusters :
$Cluster_1$ = {X1,*X2*,X3,X4}
$Cluster_2$ = {X5,X6,X7,*X8*,X9,X10}

Step 2

Select one of the non-medoids X_i
Let us assume O' =X7 = (7,3)
So now the medoids are
C1=(3,4) and O' =(7,3)

We will calculate distance and the total cost involved as the previous.



clusters after step 2

i	c_1		Data objects (X_j)		Cost (distance)
1	3	4	2	6	3
3	3	4	3	8	4
4	3	4	4	7	4
5	3	4	6	2	5
6	3	4	6	4	3
7	3	4	7	4	4
9	3	4	8	5	6
10	3	4	7	6	6

i	O′		Data objects (X_j)		Cost (distance)
1	7	3	2	6	8
3	7	3	3	8	9
4	7	3	4	7	7
5	7	3	6	2	2
6	7	3	6	4	2
7	7	3	7	4	1
9	7	3	8	5	3
10	7	3	7	6	3

Previous clusters :
$Cluster_1$ = {X1,*X2*,X3,X4}
$Cluster_2$ = {X5,X6,X7,*X8*,X9,X10}

Total cost = 3+4+4+2+2+1+3+3=22,
***so the cost of swapping mediod from C2 to O′ is*:**
S=current total cost - past total cost=
22-20=2>0.
So moving to O′ would be bad.

For more iterations we still find that our first choice was the best.

So there is no change in the medoids and the clusters are the same as the previous:
$Cluster_1$ = {X1,X2,X3,X4}
$Cluster_2$ = {X5,X6,X7,X8,X9,X10}

What are the advantages of k-medoids clustering?

- More robust than k-means in the presence of noise and outliers; because a medoid is less influenced by outliers or other extreme values than a mean.

What are the disadvantages of k-medoids clustering?

- Relatively more costly and not so much efficient.
- Need to specify k, the total number of clusters in advance.
- Result and total run time depend upon initial partition.

H.W11: Try to test the k-means by a program to see how the documents are clustered.

H.W12: Try to test the k-medoids by a program to see how the documents are clustered.

References:

1- R. Baeza-Yates and B. Ribiero-Neto, *Modern Information Retrieval*. Addison Wesley,Longman Publishing Co. Inc, 1999.
2- C. D. Manning, P. Raghavan, and H. Schutze, *An Introduction to Information Retrieval*. 2008.