# Chapter II
# Background and Related Work

In this chapter, an overview of clustering and user feedback is presented. Firstly, different clustering related topics are introduced such as: clustering technique, web clustering approaches, and common document clustering algorithms. Secondly, user feedback, its usage in information retrieval (IR) and clustering, and recent techniques that employ user feedback are introduced.

## 2.1 Clustering

Clustering is an unsupervised discovery process for separating unrelated data and grouping related data into clusters in a way to increase intra-cluster similarity and to decrease inter-cluster similarity [5, 6]. In this section, a review of clustering techniques, web clustering approaches, and document clustering algorithms is presented.

### 2.1.1 Clustering Techniques

Clustering techniques are divided into two main categories; namely hierarchical and partitional clustering techniques [7]. Hierarchical clustering works by grouping data objects into a tree of clusters, the root of this tree is a cluster which includes all clusters in the lower levels. Each intermediate level in the tree represents a cluster which subsumes all clusters which are located beneath it [7, 8]. Figure 2.1 shows a pictorial representation of hierarchical clustering.

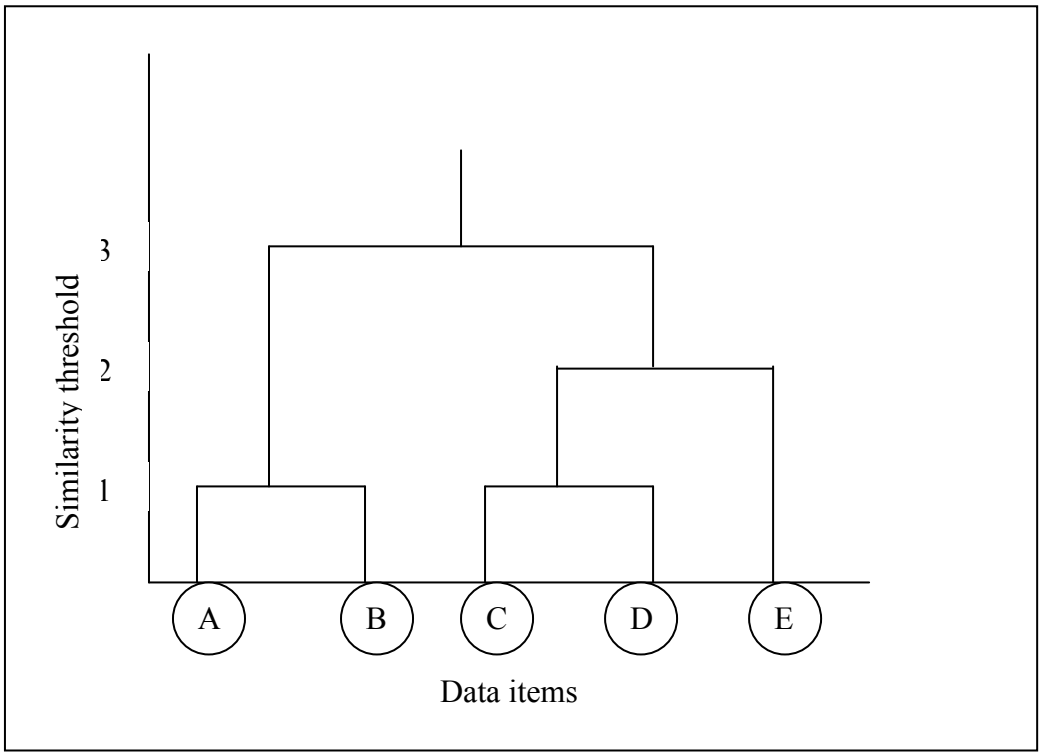



Figure 2.1. A Pictorial representation of hierarchical clustering

Hierarchical clustering can be divided into two basic approaches:

*Agglomerative clustering*: (a bottom-up technique) starts by placing each data object in its individual cluster, at consequent steps, merging most similar clusters into larger ones. *Divisive clustering*: (a top-down technique) starts with one inclusive cluster, and then it subdivides this cluster into smaller ones. a simple agglomerative clustering algorithm which is described in [7] works as follows:

1. Create a similarity matrix by calculating similarity between all pairs of clusters.
2. Merge most similar pairs.
3. Re-construct similarity matrix between the new clusters and old clusters.
4. Repeat steps 2 and 3 until only one cluster remains.

In contrast to the hierarchical clustering technique, the partitional clustering technique constructs *K* partitions of the data (*K* is the desired number of clusters), where each partition represents a cluster; *N* is the number of data objects.

An example of a simple partitioning technique follows (as described in [7]):

1. Choose *K* values as the initial clusters' centroids.
2. Assign each data object to the closest cluster based on the mean value of the entries in the cluster.
3. Re-calculate the cluster's means.
4. Repeat steps 2 and 3 until the data objects stop shifting between clusters.

### 2.1.2 Web Clustering Approaches

Web clustering is a clustering process where its desired data set contains web pages only. It is widely used because of its importance to search engines and internet applications. There are two main approaches that are used in web clustering algorithms namely: keyword and hyperlink based clustering.

In keyword based clustering, documents are represented as vectors of keywords that they contain, and then the similarity is calculated by using the document vectors. Many similarity functions are in use today to measure the distances between vectors like the Cosine-similarity, the Jaccard-similarity, and Dice-similarity [1].

Keyword based document clustering has provided interesting results when applied to offline text documents, but it has some drawbacks such as:

- Web pages are created randomly by different authors so different words may be used to express the same idea, here the similarity which is based on keywords will fail [10].
- Multimedia content is increasing in web pages, so small portion of text is used, therefore representing a document as a vector of keywords is not enough to capture the semantics of that document.
- Web pages do not adhere to common structure as would text documents do [12].

Hyperlink based clustering is an additional criterion coming for handling limitations of using keywords. The hypothesis is that hyperlinks connect related documents and when the author of a web page creates a hyperlink pointing to another web page this can be the author's endorsement of the other page [1]. This technique uses two types of web pages:

- An authoritative web page: is a one pointed to by many web pages related to the same topic.
- A hub web page: is a one that points to many good authoritative pages.

But there are some difficulties in using hyperlinks in clustering such as:

- Not every hyperlink represents the endorsement; some links may be created for advertisement or navigation purposes.
- For commercial or competitive issues, one authority will rarely have its web page points to its rival authorities in the same field, for example Microsoft prefers not to endorse its competitor Sun Microsystems by not linking to Sun Microsystems web pages.

### 2.1.3 Document Clustering Algorithms

This section provides a review of some of the well known document clustering algorithms.

**Hierarchical Agglomerative Clustering Algorithms: (Using Keywords)**

These algorithms start with assigning each document to its own cluster, and then they merge the two most similar clusters. This process is repeated until the desired number of clusters is achieved [14].

These algorithms are slow when they are applied to large document sets; the difference between these algorithms is based on the measure used to compute the similarity between clusters. The most common Hierarchical Agglomerative Clustering algorithms are

listed below (All these algorithms have $O(n^2)$ running time, where $n$ is the number of objects) :

- The single-link algorithm: this algorithm defines the similarity of two clusters as the minimal similarity of any pair of items, one from each cluster.
- The complete-link algorithm: this algorithm defines the similarity of two clusters as the maximal similarity of any pair of items, one from each cluster.
- The group-average algorithm: this algorithm defines the similarity of two clusters as the average similarity of all pairs of items.

**K-means: (using keywords)**

This algorithm starts by randomly selecting initial clusters centroids then iteratively it allocates each document to its proper cluster. After that, the K-means algorithm recalculates the clusters' centroids and re-assigns objects to clusters based on the new centroids. It has a linear complexity that's equal to $O(nkt)$ where $n$ is the number of documents, $k$ is the number of desired clusters, and $t$ is the number of iterations [14].

**Contents-Link Coupled Clustering: (using keywords and hyperlinks)**

This algorithm was proposed in 2005 by Yitong Wang and Masaru Kitsuregawa [10], it combines two web clustering approaches and introduces very good results when applied to clustering web pages. The algorithm starts by creating three vectors $D_{in}$, $D_{out}$, and $D_{keywords}$ where $D_{in}$ contains all in-links, $D_{out}$ contains all out-links, while $D_{keywords}$ contains all keywords in data set $D$. Each web page $q$ in the data set D is represented by three vectors: $q_{out}$, $q_{in}$, $q_{keywords}$ , where the $i^{th}$ item in the $q_{out}$ vector contains 1 if the document has an outlink as the $i^{th}$ one in $D_{out}$, otherwise it contains 0. $q_{in}$ is filled in the same way as $q_{out}$. $q_{keywords}$ stores web page items' frequency. Figure 2.2 shows an example of these vectors.

$D_{keyword}$
( course, java, compiler, windows )
$D_{in}$
(www.just.edu.jo,www.ju.edu.jo,www.inu.edu.jo)
$D_{out}$
(cis.just.edu.jo,www.sfu.ca,www.yahoo.com,www.google.com)

$q_{keyword}$ (0,3,2,8)
$q_{in}$ (1,0, 1)
$q_{out}$ (0,1,0,1)

The 1 in location 2 of $q_{out}$ indicates that page $q$ has an outlink that is identical to the 2nd outlink in the $D_{out}$ vector.

Figure 2.2. An example of $q_{keyword}$, $q_{in}$ , and $q_{out}$ vectors.

After these vectors are created, the similarity between two web pages $q$ and $r$ is calculated as follow:

$$P_{out} \times S(q_{out}, r_{out}) + P_{in} \times S(q_{in}, r_{in}) + P_{term} \times S(q_{keywords}, r_{keywords}) \quad (2.1)$$

Where $P_{out}$, $P_{in}$, and $P_{term}$ are weighting factors and must satisfy the equation:

$P_{out} + P_{in} + P_{term} = 1$, and $S$ is the similarity function.

After generating the similarity matrix between the new page and existing positive examples in the training data, a modified version of the K-means clustering method is applied to cluster web pages. It is described as follow:

i. Define the similarity threshold, (a value specified experimentally)

ii. Filter irrelevant pages (only relevant pages join the clustering process)

iii. Assign each relevant page to the top $C$ existing cluster(s) based on the similarities between the page and the centroids which are above the similarity threshold

iv. A page will form one cluster by itself if no existing clusters meet step iii

v. Recompute the centroids of the clusters

vi. Repeat steps iii to v until all relevant pages are assigned to clusters and the centroids do not change.

**Frequent Itemset-based Hierarchical Clustering (FIHC)**

This algorithm was introduced in 2003 by Benjamin Fung [12]. It's the best document clustering algorithm when it is compared with the K-means, UPGMA (Unweighted Pair-Group Method using Arithmetic Averages), and HFTC (Hierarchical Frequent Term based Clustering) clustering algorithms.

The FIHC uses the notion of frequent itemsets, which comes from association rule mining, for document clustering. Its main features are:

- Reduced dimensionality: it uses only frequent items that appear in a minimum fraction of documents in the document vectors.
- High clustering accuracy: it is the best document clustering algorithm in terms of accuracy, and it is robust when applied to large and complicated document sets.
- Number of clusters is an optional input parameter: it takes the number of clusters as an optional input parameter and can achieve optimal clustering quality without predetermining this value.
- Easy to browse with meaningful cluster description: it browses clusters as a tree, and each cluster in the tree has a set of frequent itemsets as that fit the description of the cluster.

The following define the terminology used by the FIHC algorithm (Figure 2.3 shows an overview of the FIHC algorithm):

- ***Global frequent itemset*** refers to a set of items (words) that appear together in more than a user-specified fraction of the document set.

- ***Global support*** of an itemset is the percentage of documents containing the itemset.
- ***Global frequent item*** refers to an item that belongs to some global frequent itemset.
- ***Cluster frequent item*** a global frequent item is cluster frequent in cluster $C_i$ if the item is contained in some minimum fraction of documents in $C_i$.

The FIHC algorithm works as follows:

Step1: Create normal document vectors (word-based),

Step2: Generate frequent itemsets based on the minimum global support defined by the user.

Step3: Construct initial clusters, construct a cluster for each global frequent itemset. All documents containing this itemset are included in the same cluster. This means that a given document may belong to several clusters.

Step4: Make clusters disjoint; assign a document to the "best" initial cluster. A cluster $C_i$ is good for a document $doc_j$ if there are many global frequent items in $doc_j$ that appear in many documents in $C_i$. Assign each $doc_j$ to the initial cluster $C_i$ that has the highest $score_i$ as shown in equation 2.2:

$$Score(C_i \leftarrow doc_j) = \left[\sum_x n(x) \times cluster\_\sup port(x)\right] - \left[\sum_{x'} n(x') \times global\_\sup port(x')\right] \quad (2.2)$$

Where:

- $x$ represents a global frequent item in $doc_j$ and the item is also cluster frequent in $C_i$.
- $x'$ represents a global frequent item in $doc_j$ but the item is not cluster frequent in $C_i$.
- $n(x)$ is the frequency of item x in the feature vector of $doc_j$.
- $n(x')$ is the frequency of $x'$ in the feature vector of $doc_j$.

Step5: Construct the tree; build a tree from bottom-up by choosing a parent for each cluster (start from the cluster with the largest number of items in its cluster label).

Choosing a parent for cluster $C_i$ in level $K$ can be done by looking for all the clusters in level $K$-1 that have the cluster label being a subset of $C_i$'s cluster label, then to determine the best parent of $C_i$, all the documents in the subtree of $C_i$ are merged into a single conceptual document $doc(C_i)$ and then compute the score of $doc(C_i)$ against each potential parent. The potential parent with the highest score would become the parent of $C_i$, finally remove any leaf cluster that does not contain any document.

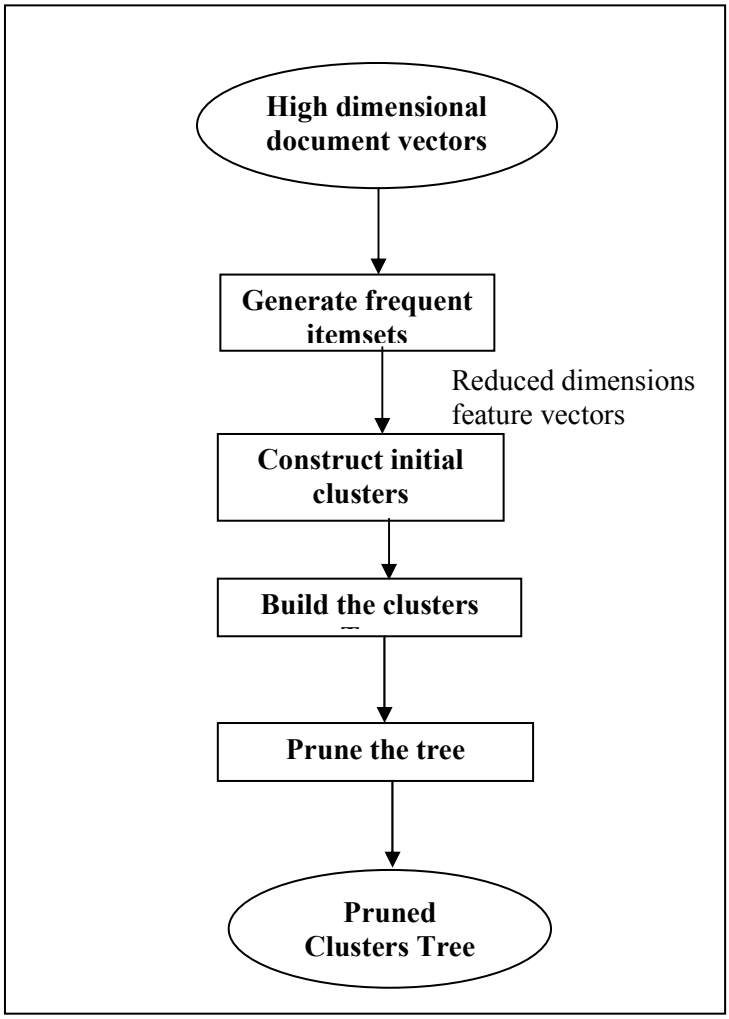



Figure2.3. The FIHC algorithm.

Step6: Prune the tree, the aim of tree pruning is to merge similar clusters in order to produce a natural topic hierarchy for browsing and to increase the clustering accuracy.

This step is divided into two phases: Child Pruning and Sibling Merging

But, before describing child pruning and sibling merging, inter-cluster similarity should be described first. Inter-cluster similarity is the main value for merging clusters.

Inter-cluster similarity between two clusters $C_a$ and $C_b$ is calculated by measuring the similarity of $C_a$ to $C_b$, it is done by treating one cluster as a conceptual document (by combining all documents in the cluster) and by calculating its score against the other cluster by using the following score equation:

$$\frac{Score(C_a \leftarrow C_b) = Score(C_a \leftarrow doc(C_b))}{\sum_x n(x) + \sum_{x'} n(x')} + 1 \quad (2.3)$$

Where:

- $x$ represents a global frequent item in $doc(C_b)$ and the item is also cluster frequent in $C_a$.
- $x'$ represents a global frequent item in $doc(C_b)$ but the item is not cluster frequent in $C_a$.
- $n(x)$ is the frequency of item x in the feature vector of $doc(C_b)$ .
- $n(x')$ is the frequency of x' in the feature vector of $doc(C_b)$ .

Then inter similarity defined as:

$$Inter - sim(C_a \leftrightarrow C_b) = \left[sim(C_a \leftarrow C_b) \times sim(C_b \leftarrow C_b)\right]^{1/2} \quad (2.4)$$

The two phases of tree pruning are describe as follow:

- **Child Pruning**: it starts by scanning the tree in bottom-up order. During this scan, for any non-leaf node calculates inter-similarity between this node and its children; and each child with inter-similarity greater than 1 is pruned.

- **Sibling Merging**: it merges similar clusters at level 1, starting by calculating the inter-similarity for each pair of clusters at level 1 and merging the cluster pair that has the highest inter-similarity, the children of the two clusters become the children of the merged cluster. Sibling merging stops when all inter-similarity between each pair are less than or equal to 1.

In this thesis we are going to employ the FIHC algorithm for creating the initial clusters as described in Section 3.1.

### 2.1.4 Clustering Applications

Some of the applications that use clustering algorithms are image segmentation, object and character recognition, and information retrieval.

Image segmentation is a basic component in many computer vision applications [13]. Clustering is used as a powerful utility for obtaining classifications of image pixels since 1960. There are many issues that must be considered before using clustering as design segmenters such as:

- Choice of pixel measurement.
- Choice of similarity measure.
- Identification of a clustering algorithm.
- Development of strategies for feature and data reduction.
- Identification of necessary pre- and post-processing techniques.

Object recognition means grouping views of 3D objects. Where a view refers to a range image of an unlocked object obtained from a viewpoint [13].

Each input image of an object forms a feature vector which characterizes that view. And then the views of each object are clustered, based on the similarity between their vectors using a clustering algorithm such as complete link hierarchical clustering.

Character recognition identifies lexemes in handwritten text for the purpose of writer-independent handwriting recognition. Here each digit is represented by a feature vector that contains its N proximities to the digits of the same class, and then a clustering algorithm is investigated to produce the best clustering for the given digits.

Information Retrieval (IR) field is concerned with automatic storage and retrieval of documents. Clustering is widely used in IR to enhance access to related information; for example clustering is widely used in library information retrieval systems to enhance access to books, articles, and other documents. One of the most common applications of information retrieval is search engine, clustering can be employed by search engines to resolve ambiguity of the query string. Search engine can employ clustering in a different way, for example, topic taxonomies of returned results (where returned web pages are grouped into several groups , each group concerns a specific topic) can enhance the readability of the returned results, and the search engine can retrieve more pages that belong to the user's interested group in the next time.

**2.2 User Feedback**

User feedback is employed to enhance clustering results, many recommendation systems were developed to enhance clustering algorithms but all these systems are visible to the user and get user feedback by requesting the user to fill in check boxes or other controls to determine if the document is relevant or not [3, 4, 7, 8, 9]. There are many types of user feedback such as: [12]

- The user might tell a system that a cluster is so bad, but he does not tell why,
- The user might indicate that some documents must be removed from a given cluster,
- The user might tell which cluster structure is good and which is bad.

In this section we will introduce different algorithms and technologies which are used to employ user feedback in enhancing search results and cluster quality.

### 2.2.1 Relevance Feedback in Information Retrieval

Relevance feedback is a classic information retrieval technique that reformulates a query based on documents identified by the user as relevant [13]. Its aim is to improve the retrieved set by removing irrelevant documents and adding more relevant ones without the user consciously constructing new search strategies, and by using only relevance or nonrelevance information obtained from the user.

The typical automatic relevance feedback operation involves an initial search with a user-supplied query and an initial retrieval of certain documents. Then, from a display (usually of titles or abstracts of the retrieved documents) the searcher identifies/chooses some relevant documents. Those documents are then used to modify the query by reweighting the existing query terms or by adding terms to the initial query that appear useful, or by deleting terms that do not. This process creates a new query which resembles the relevant documents more than the original query does [13].

The steps involved in the user-system interaction in a simple relevance feedback are:

1. The user supplies an initial query to the system.
2. The system supplies document description (usually as a ranked set of documents) to the user.
3. The user supplies relevance judgments to the system.

In its simplest form, relevance feedback could be based on one document only. For example, after displaying a single document, the system could invite the user to see more documents like the one on display.

Another simple form is when judgment is based on a set as a whole. Here, a set derived from a previous search becomes the seed for the new query formulation. The method uses "searchonyms" [1], i.e. terms which might be regarded as synonyms for the purposes of a particular enquiry and derived from terms contained in the seed set.

Automatic relevance feedback, generally, can be implemented in various ways depending on the retrieval technique used, (e.g., vector space, probabilistic, etc.), and also on the methods used to select terms for the feedback query. We can distinguish four term selection methods for query reformulation and query expansion:

- The first relies entirely on the original query and uses only those terms in the new one [13]. This method has been successfully implemented in CIRT [1].
- The second method uses terms from the original query and also adds terms from some other sources, e.g., from all the adjacent terms in the maximum spanning tree [13] or nearest neighbor terms [1].
- The third method is a mixed method because it used combinations of the terms derived from the original query and from the documents retrieved and judged relevant as found in the work of Salton and his colleagues [13].
- Finally, the fourth method abandons the terms from the original query and uses only terms found in the retrieved set of documents [1].

In all cases, after the initial query formulation, the only form of feedback to the user is a set of documents, and the only user input is user choices of documents. The query reformulation and query expansion are entrusted entirely to the retrieval system. query expansion can be performed with or without term reweighting. Query expansion without term reweighting, may involve the addition of terms from a knowledge structure, such as thesauri, term classifications, etc. With reweighting a new weight is assigned to the

original set of keywords. Most research on relevance feedback and query expansion have been done using both query expansion and term reweighting [1].

### 2.2.2 Implicit User Feedback

Explicit user feedback does not apply easily in different real-world information and filtering applications [2]. Recently many researches are discussing and proposing techniques to infer user feedback implicitly. A study for Steve Fox [2] shows that implicit user feedback can substitute for explicit user feedback. This study proposes different measures that can be used to infer user feedback as shown in Table 2.1.

This thesis uses two of the measures shown in Table 2.1, namely duration in seconds, and visits (number of visitors) to calculate average browsing time to implicitly infer user feedback as in described in Chapter 3. Another technique is proposed in [25] where documents are represented in different ways (such as top ranked sentences (TRS), document title, summary sentence, sentence in context, and full text document). The idea of this technique is that the user starts by browsing TRS of the document if he is interested with it, he will browse the next representation which is the page title, and if he is still interested, he goes to the next representation until he reaches the document. The relevance degree is determined by observing the browsing path. As the user goes deeply in this path, the more relevant this document is.

### 2.2.3 Semi-Supervised Clustering

Using user feedback in information retrieval to enhance the result quality has encouraged researches in clustering to benefit from this method. This approach generates some constraints derived from user feedback, then the clustering algorithm attempts to satisfy these constraints in future iterations [12]. The algorithm uses initially an

unsupervised clustering algorithm to cluster documents, and then the user browses the resulting clusters and provides feedback to the system by saying:

- "This document doesn't belong in here".
- "Move this document to that cluster".
- "These two documents shouldn't be (or should be) in the same cluster".

Table 2.1. Implicit measures of user feedback

| **Measure** | **Description** |
|---|---|
| Time<br><br>Difference in seconds<br><br>Duration in seconds | Difference in seconds: represents the time elapsed from the moment the focus leaves the result pages to the time the focus is returned to that page. Presumably during that time the user will be browsing one of the pages returned as a result.<br><br>Duration in seconds: represents the time spent on browsing a given page. |
| Scrolled, scrolling count, average seconds between scrolls, total scroll time. | Scrolled is a Boolean value that determines if the user scrolled the page or not; scrolling count records the number of scrolls the user has done; average seconds between scrolls represents the time spent browsing each part in the document, total scroll time record the time spent on scrolling the page. |
| Time to first click, time to first scroll. | To obtain initial activity times. |
| Page position | The position of the page in the result list. |
| Visits | Number of visitors to the page. |
| Exit type | How the user ends the page visit, time out or close the window or navigate to a new page, etc. |
| Image count, page size, script count | Characteristics of the page |
| Added to favorites, printed | Added to favorites or printed meaning that this page is relevant and meeting user's query. |

Semi-supervised clustering is neither unsupervised nor supervised clustering technique. Unsupervised clustering takes an unlabeled collection of data and partitions it into groups with high intra-cluster similarity and low inter-cluster similarity. Supervised clustering, on the other hand, takes a set of data with predefined class labels and returns a function that maps data to class labels.

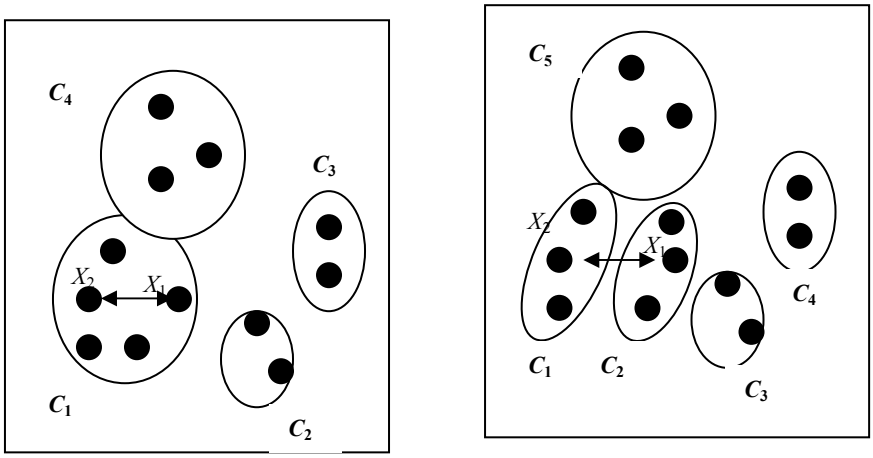
To illustrate semi-supervised clustering, assume that two documents $x_1$ and $x_2$ , were clustered into the same cluster by the employed algorithm, when the user browses these documents the user says "these two documents shouldn't be in the same cluster", consequently the clustering algorithm must modify the distance measure in the next iteration to increase the distance between $x_1$ and $x_2$ to separate them. Figure 2.4 shows how semi-supervised clustering works. Where $c_1$, $c_2$, $c_3$, and $c_4$ are clusters. $x_1$ and $x_2$ are documents. Figure 2.4(a) describes the initial distribution of documents to clusters. Before user feedback is taken into consideration the user notices that documents $x_1$ and $x_2$ belong to the same cluster and instructs the system to separate them. Figure 2.4(b) demonstrates how the system re-adjusts its distance measure to split $x_1$ and $x_2$ based on user feedback.

### 2.2.4 User Feedback-Driven Document Clustering

User feedback-driven clustering technique as described in [14] uses three phases to cluster a set of documents:

**Phase 1**: Pre-clustering (generation of fine-grained clusters):

The system partitions a given document collection into small clusters based on the distance between documents. The complete-linkage hierarchical agglomerative clustering is used as a basic clustering algorithm to generate fine-grained clusters.




(a) (b)

Clusters before user feedback Clusters after user feedback

Figure 2.4. An illustration of semi-supervised clustering.

**Phase 2**: Supervision phase (user feedback):

In this phase, two types of document bundles (i.e group of documents) are created: positive bundle and negative bundle. These bundles are developed by a relevance feedback program from interviews with the user. The interview program starts by extracting a set of documents randomly from a set of pre-clusters, then the user determines relevant documents to put them in the positive bundle and irrelevant documents to put them in the negative bundle.

**Phase 3**: Re-clustering (assigns each of the pre-clustered document to its nearest positive document bundle):

In this phase each of the pre-clustered documents is assigned to the positive bundle in which its nearest document is found. During assigning of pre-clusters to the positive bundle; the local centroid of the cluster is updated. At the same time the negative bundle is observed to prevent assigning any documents in the negative bundle to the positive bundle. If such document is assigned to the positive bundle then it is re-assigned to another cluster that has its second nearest document.

As it has been demonstrated above, all techniques that have investigated user feedback were based on explicit user feedback; Chapter 3 shows how implicit user feedback rather than explicit one was investigated to enhance cluster quality.