Information Processing and Management 47 (2011) 287–299

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Using a new relational concept to improve the clustering performance of search engines ☆

Lin-Chih Chen *

Department of Information Management, National Dong Hwa University, No. 1, Sec. 2, Da Hsueh Road, Shou-Feng, Hualien 97401, Taiwan

ARTICLE INFO

Article history:
Received 8 April 2008
Received in revised form 17 November 2008
Accepted 14 April 2010
Available online 7 May 2010

Keywords:
Document clustering
Semantic relation
Relational concept
Web search engines
Web documents

ABSTRACT

In this paper, we present a novel clustering algorithm to generate a number of candidate clusters from other web search results. The candidate clusters generate a connective relation among the clusters and the relation is semantic. Moreover, the algorithm also contains the following attractive properties: (1) it can be applied to multilingual web documents, (2) it improves the clustering performance of any search engine, (3) its unsupervised learning can automatically identify potentially relevant knowledge without using any corpus, and (4) clustering results are generated on the fly and fitted into search engines.

1. Introduction

Document clustering is an important technology that can automatically discover groups of similar web documents in a set of the web documents returned by a search engine (Berkhin, 2002). Organizing web documents by clustering can help the user get a sense of the major subject areas covered in the web document set and help the user find the relevant web documents more quickly. Clustering techniques can be divided into hierarchical and partitional methods (Berkhin, 2002; Cios, Pedrycz, Swiniarski, & Kurgan, 2007).

Hierarchical methods yield an entire sequence of nested partitions, and these methods can be either agglomerative or divisive. Agglomerative methods yield a sequence of nested partitions starting with one-document clusters recursively merging two or more of the most appropriate clusters. Divisive methods start with one cluster of all documents and recursively split the most suitable cluster. Many clustering algorithms belong to hierarchical methods, such as HAC (Voorhees, 1986), STC (Zamir & Etzioni, 1998), and DIVCLUS-T (Chavent, Lechevallier, & Briant, 2007).

Partitional methods find the clusters by partitioning the entire document collection into either a predetermined or an automatically derived number of clusters. Partitional methods can be considered an optimization procedure that tries to create high quality clusters according to a particular criterion function to achieve high intra-cluster similarity and low inter-cluster similarity. Many clustering algorithms belong to partitional methods, such as K-means (MacQueen, 1967), SRE (Zha, He, Ding, Simon, & Gu, 2001), and k-Attractors (Kanellopoulos, Antonellis, Tjortjis, & Makris, 2007).

In this paper, we propose a novel clustering algorithm, called *On-The-Fly Document Clustering* (OTFDC). OTFDC is based on a connective semantic relation between the clusters (as defined as Section 3), and it has the following four properties: (1) it can be applied to multilingual web documents, (2) it improves the clustering performance of any search engine, (3) it uses

☆ The experimental search engines used in this paper is available at http://cayley.sytes.net/english and http://cayley.sytes.net/chinese.

* Tel.: +886 3 863 3111; fax: +886 3 863 3100.
E-mail address: lcchen@mail.ndhu.edu.tw

0306-4573/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2010.04.001

unsupervised learning to automatically identify potentially relevant knowledge without using any corpus, and (4) it provides clustering results generated on the fly and fitted into search engines.

This paper is organized as follows. Section 2 introduces related work in the area of document clustering. Section 3 discusses OTFDC in detail. We then present some preliminary simulation results in Section 4. Finally, Section 5 concludes the paper.

2. Related work

This section reviews some important literature related to the paper. First, we provide a literature review on different concepts of the document clustering. Second, we discuss some online academic document clustering systems. Third, we provide a short description of Search engine Vector Voting (SVV).

2.1. Different concepts of the document clustering

Document clustering is an important data exploration technique that groups similar documents into a cluster. However, document clustering is a difficult problem since the documents contain large amounts of unstructured text (Szczepaniak, Segovia, Kacprzyk, & Zadeh, 2003). Many researchers use different concepts and methods to solve the clustering problem.

Some researchers have used the fuzzy concept to solve the clustering problem. Tjhi and Chen (2007) combined the possibilistic and fuzzy formulations of co-clustering for automatic categorization of large document collections. The combined approach allows the word memberships to represent fuzzy word partitions, however it captures the natural word clusters structure. Saraçoğlu, Tütüncü, and Allahverdi (2007) designed a two-layer structure and let the documents pass through it to find the similarities. In two-layer structures, they used predefined fuzzy clusters to extract feature vectors of related documents. The similarity measure is estimated based on these feature vectors.

Other researchers have used the domain concept to solve the clustering problem. Kerne, Koh, Sundaram, and Mistrot (2005) presented an iterative method for generative semantic clustering of related terms in the documents. Their method for generating semantic clustering is based on a quantitative model that represents mutual information between each new domain and the domains already in the document. Dai, Xue, Yang, and Yu (2007) presented a co-clustering based method to learn an unlabeled data set in an unknown domain. The learning process starts with a labeled data set from a known domain which is called an in-domain, and applies the learning knowledge to a related but different domain which is called an out-of-domain.

Further researchers have used the query concept to solve the clustering problem. Chang, Kim, and Raghavan (2006) constructed the query concepts to express the user's information needs, rather than trying to reformulate the initial queries. To construct the query concepts, they extracted all document features from each document and then clustered these document features into primitive concepts that are used to form query concepts. Li, Chung, and Holt (2008) used the query sequence to rearrange clustering results. They claimed a query sequence is frequent if it occurs in more than a certain percentage of the documents in all document sets.

2.2. Online document clustering systems

Several online document clustering systems have been proposed in literature. WebCat (Giannotti, Nanni, Pedreschi, & Samaritani, 2003) uses Transactional K-Means to generate desirable clustering results. CIIRarchies (Lawrie & Croft, 2003) uses a probabilistic technique to extract sentences and builds the cluster labels in a hierarchy via a recursive algorithm. Lingo (Osinski & Weiss, 2004) uses a combination of phrases and Singular Value Decomposition (SVD) on a term-document matrix to find meaningful long cluster labels. SHOC (Zhang & Dong, 2004) uses the suffix array for sentences extraction and presents the cluster labels in a hierarchy via Suffix Tree Clustering (STC). FIHC (Fung, Wang, & Ester, 2003) uses the frequent itemset-like approach to produce a hierarchical topic tree for clusters. SnakeT (Ferragina & Guli, 2008) also uses an approach similar to the frequent itemset to extract meaningful labels and builds the cluster labels with the bottom-up hierarchy construction process. But none of these systems are available on the Internet.

On the other hand, many of online document clustering systems are available on the Internet. Credo (Carpineto & Romano, 2004a, 2004b) uses a Formal Concept Analysis (FCA) to construct the cluster label in a hierarchy form that allows the user to query documents and see retrieval results organized in a browsable concept lattice. However, the results of Credo are not always a sentence form. Carrot2 (Osinski & Weiss, 2005; Weiss & Stefanowski, 2003) is an implementation of STC with extensions such as stop-words and stemming. It used sentences with variable length words as the cluster label, but these sentences were drawn as contiguous portions of the source document. CatS (M. Radovanović & Ivanović, 2006) uses a separate category tree derived from the dmoz Open Directory topic hierarchy to arrange all the cluster labels, but its results are restricted to a fixed set of topics. Highlight (Wu & Chen, 2003) adopts lexical analysis and a probabilistic technique to construct a concept hierarchy on the cluster label, but its classification is very rough. Vivisimo (Segev, Leshno, & Zviran, 2007) uses the Concise All Pairs Profiling (CAPP) to match all pairs of possible clusters in order to distinguish between any two clusters.

2.3. Search engine vector voting

In our previous research (Chen & Luh, 2005), we proposed a MetaSearch engine, called *Search engine Vector Voting* (SVV), which can simultaneously retrieve the search results from several search engines. In order to calculate the final ranking of the web documents, we first need to gather all the relevant web documents and their ranking which appear in the search results of the search engines. Next, we calculate the weight of all relevant web documents by calculating the dot product. We need to define a new function called *User Behavior Function* (UBF) that can be used to calculate the weight of all relevant web documents, as shown in the following equation:

$$\mathrm{UBF}(l, obj) = \alpha l_{obj}^{\beta} \quad (\text{where } \alpha > 0 \text{ and } \beta < 0) \tag{1}$$

where α is the user preference of first item; l_{obj} is the relative order of the object *obj* in an item list l; and β is an important quality control parameter. A strong user's preference with a large β value leads UBF to decrease less as illustrated in the case of $\beta = -0.2$ (Fig. 1).

Next, the weight of all the relevant web documents can be derived from UBF, as shown in the following equation:

$$W_{wp_{i,m},i} = \sum_{s=1}^{n} \alpha_{s,i} x_{s,wp_{i,m},i}^{\beta} \tag{2}$$

where n is the number of search engines; $w_{wp_{i,m},i}$ is the weight of a web document $wp_{i,m}$ when the user query is i ($1 \leqslant m \leqslant n$); $\alpha_{s,i}$ is the user preference of first web document when the user query is i returned from the search engine s; $x_{s,wp_{i,m},i}$ is the ranking of $wp_{i,m}$ when the user query is i returned from s; and β is a control parameter.

According to UBF, if a web document obviously has large weight either it wins more votes from the search results of the search engines or its rank is relatively high in at least one search engine. To calculate $w_{wp_{i,m},i}$, SVV needs to generate two random numbers, which represent $\alpha_{s,i}$ and β, respectively. According to literature, human behavior is a random process (Malerba, Esposito, & Ceci, 2002; Zipf, 1949) and moreover, human preferences are values ranging between a lower bound and an upper bound (Yao, 1995). Thus, we use random numbers to simulate user behavior. The initial ranges of α and β are obtained from our own evaluation experiments and the user accessing log. Then both $\alpha_{i,q}$ and β are be updated regularly according to performance evaluation results.

3. On-The-fly Document Clustering

In this section, we present a novel clustering algorithm, called *On-The-Fly Document Clustering* (OTFDC), which produces a number of candidate clusters from other web search results. The candidate clusters not only generate a connective relation between the clusters, but also the relation is a semantic one.

Definition 1 (*SR*). ($Cluster_i$,$Cluster_t$) have a *semantic relation* (SR) feature when $Cluster_i$ and $Cluster_t$ satisfy at least one of the following three relations: (a) equivalence, (b) hierarchy, and (c) association (NISO, 2005). If there is an equivalent relation between $Cluster_i$ and $Cluster_t$, these two clusters express the same concept. For example, ("photocopies", "Xeroxes") and ("freedom", "liberty") exist in an equivalence relation, respectively. A hierarchical relation is based on degree of superordination and subordination, where the superordinate cluster may represent a class or a whole, and the subordinate

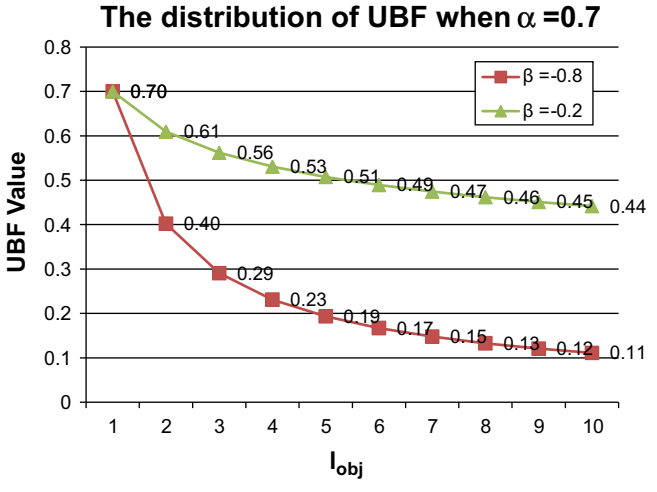

Fig. 1. The distribution of the UBF function.

cluster refer to its members or parts. For example, (“mountain regions”, “Alps”) and (“nervous system”, “brain”) exist in a hierarchical relation, respectively. An associational relation includes associations between clusters that are neither equivalent nor hierarchical, though the clusters are semantically associated to such an extent that the relation between them should be made explicit. For example, (“flour”, “wheat”) and (“Greek civilization”, “Socratic method”) exist in an association relation, respectively.

Definition 2 (HSR). *Clusteri,Clustert* exist in a *high-level SR* (HSR) feature *if and only if* ($Cluster_i$,$Cluster_t$) exist in a SR feature and $T \leqslant R(Cluster_i,Cluster_t) \leqslant 1$, where $R(Cluster_i,Cluster_t)$ is the correlation coefficient between two clusters (as defined later in this section) and T is the threshold value (as defined later in Section 4). HSR implies that $Cluster_i$ and $Cluster_t$ not only exist in a SR feature but also $R(Cluster_i,Cluster_t)$ is greater than or equal to the lower bound of T. For example, (“p2p”,“peer-to-peer”) exist in a HSR feature since we can find a “p2p is an abbreviation for peer-to-peer” relation and R(“p2p”,“peer-to-peer”) is 100% which is greater than or equal to T (According to the follow-up example and simulation results, we obtain R(“p2p”,“peer-to-peer”) is 100% and T is 0.801).

We use the relational concept to express OTFDC. There is a relational path with length n from $Cluster_i$ to $Cluster_t$ *if and only if* ($Cluster_i$,$Cluster_t$) $\in HSR^n$. The connectivity relation HSR^* consists of the pairs ($Cluster_i$,$Cluster_t$) within a relational path between $Cluster_i$ and $Cluster_t$ in *HSR*. Therefore, HSR^n consists of the pairs ($Cluster_i$,$Cluster_t$), and there is a relational path within length n from $Cluster_i$ to $Cluster_t$ following HSR^* (the union of all sets of HSR^n). In other words, $HSR^* = \cup_{n=1}^{\infty} HSR^n$. In Fig. 2, we show how OTFDC is implemented by the connectivity relation HSR^*.

Definition 3 (*CHSR*). ($Cluster_i$,$Cluster_t$) exist in a *connective HSR* (CHSR) feature *if and only if* ($Cluster_i$,$Cluster_t$) exist in a HSR feature and ($Cluster_i$,$Cluster_t$) $\in HSR^*$. That is, we can find an intermediate cluster $Cluster_d$ so ($Cluster_i$,$Cluster_d$) $\in HSR^*$ and ($Cluster_d$,$Cluster_t$) $\in HSR^*$, where ($Cluster_i$,$Cluster_d$) and ($Cluster_d$,$Cluster_t$) exist in a HSR feature, respectively. For example, (“p2p”,“bittorrent”) exist in a CHSR feature since we can find (“p2p”,”peer-to-peer”) $\in HSR^*$ and (“peer-to-peer”,“bittorent”) $\in HSR^*$. Namely, (“p2p”,”peer-to-peer”) can be found in “*p2p* is an abbreviation for *peer-to-peer*” and “one of the most successful *peer-to-peer* applications is *bittorrent*” relations.

The pseudo code of OTFDC is listed as follows:

```
1   Algorithm OTFDC (Cluster_i as String)
2   {
3     (Cluster_t, R(Cluster_i,Cluster_t)) = Call OTFDC_Core (Cluster_i);
4     Foreach (Cluster_t)
5   {
6     Append Cluster_t into FinalClusterSet;
7     Append R(Cluster_i,Cluster_t) into FinalCorCoefSet;
8     (FinalClusterSet, FinalCorCoefSet) = Call OTFDC (Cluster_t);
9   } End of Foreach;
10  (FinalClusterSet, FinalCorCoefSet) = Sort FinalClusterSet according to its correlation coefficient FinalCorCoefSet;
11    Return (FinalClusterSet, FinalCorCoefSet);
12  } End of Algorithm;
```

OTFDC is recursively invoked on OTFDC_Core that generates a number of the candidate clusters for each recursive call. The main assumption of OTFDC_Core is if many important clusters link to an important web document, then OTFDC_Core can use the reverse links of an important web document to find these important clusters. For example, in Fig. 3, “p2p”, “peer-

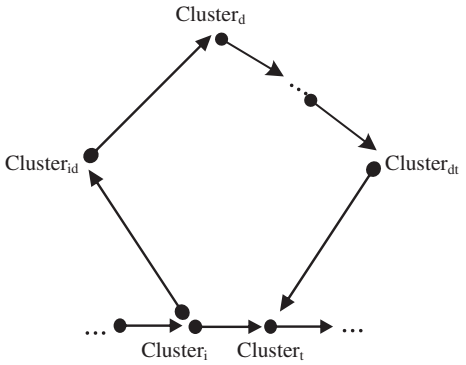

Fig. 2. The connectivity relation HSR^* consists of the pairs ($Cluster_i$,$Cluster_t$).

Fig. 3. The relationship of the clusters and a web document. OTFDC assumes that many important clusters link to an important web document, and then OTFDC can use the reverse links of an important web document to find these important clusters.

to-peer", and "decentralized p2p" clusters link to an important web document "en.wikipedia.org/wiki/Peer-to-peer". Thus, OTFDC_Core can use the reverse links of "en.wikipedia.org/wiki/Peer-to-peer" to find the "p2p", "peer-to-peer", and "decentralized p2p" clusters.

The pseudo code of OTFDC_Core is listed as follows:

```
1  Algorithm OTFDC_Core (Cluster_i as String)
2  {
3  (wp_{i,m}, w_{wp_{i,m},i}) = Call SVV (Cluster_i);
4  (wp_{i,m}, w_{wp_{i,m},i}) = Sort wp_{i,m} according to its weight w_{wp_{i,m},i};
5  Normalize the weight w_{wp_{i,m},i};
6  (CandidateCluster_t) = Use the reverse links of an important web document wp_{i,m} to find out all the candidate
   clusters.
7  Foreach (CandidateCluster_t)
8  {
9  (wp_{t,m}, w_{wp_{t,m},t}) = Call SVV (CandidateCluster_t);
10 (wp_{t,m}, w_{wp_{t,m},t}) = Sort wp_{t,m} according to its weight w_{wp_{t,m},t};
11 If (w_{wp_{t,m},t} < T)
12    Continue;
13 Normalize the weight w_{wp_{t,m},t};
14 Compute the correlation coefficient;
15 If (R(Cluster_i,CandidateCluster_t) ⩾ T and CandidateCluster_t ∉ FinalClusterSet)
16 {
17    Append CandidateCluster_t into FinalClusterSet;
18    Append R(Cluster_i,CandidateCluster_t) into FinalCorCoefSet;
19 } End of If;
20 } End of Foreach;
21 (FinalClusterSet, FinalCorCoefSet) = Sort FinalClusterSet according to its correlation coefficient FinalCorCoefSet;
22 Return (FinalClusterSet, FinalCorCoefSet);
23 } End of Algorithm;
```

Then, we use the following example to illustrate how to use OTFDC_Core to solve the single-pass clustering problem.

Example 1. OTFDC_Core refers to our previous project which developed the search engine vector voting (SVV) method described in Section 2.3. Table 1 shows the rank-order distribution and the partial SVV results among four major search engines when the user query is "p2p" (Lines 3 and 9). In order to calculate $w_{wp_{i,m},i}$, $\alpha_{s,i}$ and β need to be defined. Initially, the two random numbers are generated respectively from their own range, as shown in Table 2. Then, the weight of "en.wikipedia.org/wiki/Peer-to-peer" is 3.62054 ($0.92241 * 1^{-0.43501} + 0.86957 * 1^{-0.43501} + 0.93472 * 1^{-0.43501} + 0.89384 * 1^{-0.43501}$).

Table 3 shows the sorted list of the partial SVV results according to its weight (Lines 4 and 10). Based on the literature, human behavior is a random process (Malerba et al., 2002; Zipf, 1949). To reduce random measurement noise, OTFDC_Core normalizes $w_{wp_{i,m},i}$ expressed in percentage form by the following equation (Lines 5 and 13), as shown in Table 3:

$$Norm_i = w_{wp_{i,m},i} \Big/ \sum_{j=1}^{n} \alpha_{s,i} \tag{3}$$

Table 1
The rank-order distribution among four major search engines when the user query is "p2p".

	Google	Yahoo	MSN	Askjeeve	$w_{wp_{i,m},i}$
en.wikipedia.org/wiki/Peer-to-peer	1	1	1	1	3.62054
isohunt.com	2	2	3	2	2.56627
openp2p.com	0	4	0	4	0.96483
en.wikipedia.org/wiki/P2P	6	7	2	5	1.93127

A "0" denote a web document cannot be found by any search engine.
All the examples presented in this paper were collected on 3/1/2008.

Table 2
The values of $\alpha_{s,i}$ and β.

	Google	Yahoo	MSN	GAIS
$\alpha_{s,i}$ range	0.85–0.95	0.85–0.95	0.85–0.95	0.85–0.95
β range	−1 to −0.3			
$\alpha_{s,i}$	0.92241	0.86957	0.93472	0.89384
β	−0.43501			

Table 3
The sorted list of the partial SVV results and its corresponding normalized weight when the user query is "p2p".

$wp_{i,m}$	$w_{wp_{i,m},i}$	$\sum_{s=1}^{n}\alpha_{s,i}$	$Norm_i$, %
en.wikipedia.org/wiki/Peer-to-peer	3.62054	3.62054	100
isohunt.com	2.56627		70.881
en.wikipedia.org/wiki/P2P	1.93127		53.342
openp2p.com	0.96483		26.649

where $Norm_i$ is the normalized weight of $wp_{i,m}$.

OTFDC_Core uses the reverse links of an important web document to find all the important clusters (Line 6). For example, Table 4 shows the partial candidate clusters ($CandidateCluster_t$) of "en.wikipedia.org/wiki/Peer-to-peer".

In OTFDC_Core analysis, we also assumed the candidate clusters derived from an important web document were more likely to be related to the source cluster. Thus, OTFDC_Core only considers the candidate cluster t when the weight of a web document $wp_{t,m}$ is greater than a Threshold T (Lines 11 and 12). For example, OTFDC_Core does not care if $Candidate\text{-}Cluster_t$ is "mfpnet" since the weight of $wp_{t,m}$ ("en.wikipedia.org/wiki/Peer-to-peer") is 0.65124 which is less than T (0.801).

To realize the similarity degree of any two clusters, OTFDC_Core uses the following equation to calculate the correlation coefficient between any two clusters (Line 14).

$$R(Cluster_i, Cluster_t) = \min(Norm_i, Norm_t)/Norm_i \tag{4}$$

where $Norm_t$ is the normalized weight of $wp_{t,m}$. The candidate cluster is t and $R(Cluster_i,Cluster_t)$ is the correlation coefficient between $Cluster_i$ and $Cluster_t$. For example, in Table 4, R("p2p",$CandidateCluster_t$) is the correlation coefficient between "p2p" and all the partial candidate clusters $CandidateCluster_t$.

According to Eq. (4), we can guarantee the upper bound of $R(Cluster_i,Cluster_t)$ is less than or equal to 1 while $\min(Norm_i,Norm_t)$ is less than or equal to $Norm_i$. OTFDC_Core also uses T as the lower bound of $R(Cluster_i,Cluster_t)$ to achieve HSR, as shown in the following equation.

$$R(Cluster_i, Cluster_t) = \begin{cases} [T, Norm_t, /Norm_i], & if\,(Norm_t \le Norm_i) \\ 1, & otherwise \end{cases} \tag{5}$$

Table 4
The parameters of the partial candidate clusters for $wp_{t,m}$ is "en.wikipedia.org/wiki/Peer-to-peer".

$CandidateCluster_t$	$w_{wp_{t,m},t}$	$w_{wp_{t,m},t} > T$?	$\sum_{s=1}^{n}\alpha_{s,t}$	$Norm_t$, %	R("p2p", $CandidateCluster_t$), %
Peer-to-peer	3.45639	√	3.45639	100	100
Decentralized p2p	2.67197	√	3.33127	80.209	80.209
mfpnet	0.65124	×			
p2ptv	2.04671	√	3.51793	58.179	58.179
p2p	3.62054	√	3.62054	100	100

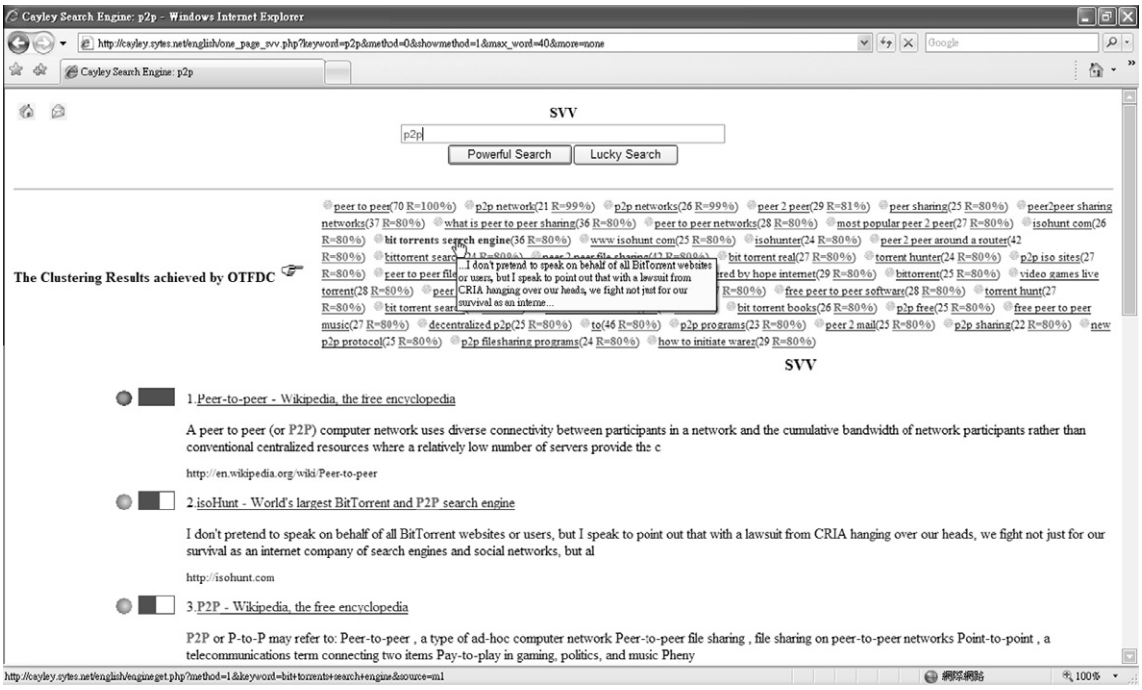

Fig. 4. A sample run of OTFDC (Source is English web documents) when the user query is "p2p". The experiment search engine (http://cayley.sytes.net/english) uses OTFDC to produce a number of candidate clusters when the source of OTFDC is English web documents.

That is, OTFDC_Core only selects the candidate clusters from the correlation coefficient of the clusters that have not yet been considered greater than or equal to T (Line 15). For example, in Table 4, the correlation coefficient of "peer-to-peer", "decentralized p2p", and "p2p" is greater than or equal to T, yet "p2p" has been considered. At the end of this step, OTFDC_Core considers all the partial candidate clusters as "peer-to-peer" and "decentralized p2p".

Finally, OTFDC_Core sorts all the candidate clusters according to its corresponding correlation coefficient (Line 21). For example, the sorted list of the partial OTFDC_Core results are ("peer-to-peer", "decentralized p2p") when the source of OTFDC_Core is "p2p".

OTFDC uses a series of recursive OTFDC calls to generate the SR feature. In the first recursive call, a source cluster, Cluster_i, generates some candidate clusters, $Cluster_t$. Hence, $Cluster_i$ is equivalent to $Cluster_t$ since they share same concept (web document) $wp_{i,m}$. In the second recursive call, the source $Cluster_t$ generates some candidate $Cluster_k$. That is, $Cluster_i$ and $Cluster_k$ are in a hierarchical relation since the subordinate cluster $Cluster_k$ can be accessed from the superordinate $Cluster_i$ through an intermediate $Cluster_t$. In the nth recursive call, the source $Cluster_k$ generates some candidate $Cluster_m$. Therefore, $Cluster_i$ is associated with $Cluster_m$ because there is a path with length n from $Cluster_i$ to $Cluster_m$. For each recursive call, OTFDC defines the target cluster with a "good label" property generated from the source cluster when the correlation coefficient between the source cluster and the target cluster is greater than or equal to a threshold T value (Line 15 in OTFDC_Core).

A sample procedure of OTFDC has been conducted to produce a number of candidate clusters with the correlation coefficient in response to the user query called "p2p", as illustrated in Fig. 4. OTFDC sorts all the candidate clusters according to their corresponding correlation coefficient (R). For example, R is 100% for "peer-to-peer".

Moreover, OTFDC also can be applied to solve multilingual clustering problems when the source of OTFDC is multilingual web documents. Fig. 5 is the clustering results of "mlb" derived from the source OTFDC of Chinese web documents.

4. Experimental results

Three types of performance were performed using various clustering algorithms. First, 12 students were selected from National Dong Hwa University to judge the quality of clustering results for 39 top queries on the Internet during 2007. Second, we use a computer simulation to determine a suitable *Threshold* (T) value to use in OTFDC. Third, we also simulate OTFDC applied to Google, Yahoo, and Vivisimo in order to verify whether OTFDC can improve the clustering performance of any search engines or not.

4.1. User study

In this experiment, we pay attention to how the users rate OTFDC compared with other online systems. We designed an online questionnaire to ask 10 undergraduate and two graduate students from National Dong Hwa University to compare

Fig. 5. A sample run of OTFDC (Source is Chinese web documents) when the user query is "mlb". The experiment search engine (http://cayley.sytes.net/chinese) uses OTFDC to produce a number of candidate clusters when the source of OTFDC is Chinese web documents.

OTFDC (Chen, 2008a) results against those provided by Credo (Carpineto & Romano, 2004), Carrot2 (Weiss & Osinski, 2004), CatS (Miloš Radovanović, 2006), Highlight (Wu, 2003), and Vivisimo (Vivisimo, 2004) from September 25, 2008 to October 5, 2008[1]. The underlying systems here are discussed in detail in Section 2. In addition, we selected 39 of the most searched queries on Google (Google, 2007), Yahoo (Saremi, 2007) and Lycos (Lycos, 2007), respectively.

The questionnaire was been manually answered by 12 students who judged the relevance of the results with respect to the 39 queries.[2]

For each of the 39 queries, the 12 students evaluators computed the precision at the first N cluster labels associated with queries generated by each system. Precision at top N is defined as

$$P@N = \frac{M@N}{N} \tag{6}$$

where $M@N$ is the number of cluster labels that have been *manually tagged* as relevant among the first N cluster labels computed by each system.

The results of the experiment for $P@3$ are shown in Table 5.[3] The number in every digit cell (except the last row) is the average $P@3$ score of each system on *a given evaluated query* for 12 students. For example, OTFDC has an average $P@3$ scored 80.95% when the query is "anna nicole smith". The number in the last row is the average $P@3$ score of each system on *all evaluated queries* for the 12 students.

The average scores of OTFDC, Credo, Carrot2, Cats, Highlight, and Vivisimo are 78.51%, 54.29%, 72.98%, 49.27%, 69.62%, and 74.48%, respectively. Moreover, we have extended $P@3$ analysis to $P@5$, $P@7$, and $P@10$ analyses. According to Table 6, we can find the lowest score, Cats, for all analyses since its results are restricted to a fixed set of topic and the second lowest score, Credo, for all analyses since its results are not always in sentence form.

Next, we use SPSS 11.0 for Windows to analyze the results of above experiment. Because the results of SPSS are tedious, and thus we refer the reader to (Chen, 2008b) for a full report. Considering the performance of OTFDC, Credo, Carrot2, Cats, Highlight, and Vivisimo. We used the statistical methodology, Analysis of Variance (ANOVA) analysis to show that $F_{(PR@3)} = 3055.950$, $F_{(PR@5)} = 2759.889$, $F_{(PR@7)} = 2834.710$, and $F_{(PR@10)} = 3045.559$ (Table 7) are all greater than

[1] The online questionnaire will remember your settings so you do NOT need to refill the questionnaire in the next time and its URL is http://cayley.sytes.net/OTFDC_questionnaire/questionnaire.php.

[2] Two of the evaluators are advanced internet users, whereas the remaining 10 were just normal internet users. The queries were randomly partitioned among the evaluators.

[3] All of the experiment results are shown in http://cayley.sytes.net/OTFDC_questionnaire/calculate_quest.php.

Table 5
P@3 over 39 most searched queries during 2007.

Query	OTFDC, %	Credo, %	Carrot2, %	Cats, %	Highlight, %	Vivisimo, %
Anna nicole smith	80.95	55.48	77.62	55.24	69.76	71.43
Badoo	73.81	57.14	76.19	45.24	66.67	73.81
Beyonce	73.81	52.38	75.95	47.62	66.67	71.43
Britney spears	76.19	59.52	71.43	52.38	69.05	71.43
Club penguin	73.81	52.38	69.05	45.24	64.29	76.19
Dailymotion	76.19	54.76	73.81	47.62	64.29	71.43
Disney	73.81	50.00	76.19	47.62	73.81	71.43
Ebuddy	76.19	54.76	69.05	45.24	71.43	76.19
Facebook	78.57	52.38	69.05	52.38	64.29	73.81
Fantasy football	83.33	57.14	69.05	50.00	71.43	71.43
Fergie	78.57	57.14	69.05	50.00	73.81	78.57
Fifa	80.95	50.00	69.05	52.38	73.81	71.43
Golf	76.19	59.52	73.81	47.62	71.43	76.19
Heroes	73.81	57.14	76.19	47.62	69.05	78.57
Hi5	76.19	57.14	73.81	47.62	71.43	78.57
Iphone	83.33	50.00	69.05	52.38	64.29	76.19
Jessica alba	78.57	54.76	73.81	45.24	69.05	76.19
Kazaa	76.19	54.76	76.19	50.00	66.67	73.81
Lindsay lohan	80.95	50.00	76.19	50.00	71.43	73.81
Mozart	73.81	57.14	73.81	52.38	73.81	78.57
Mp3	80.95	50.00	69.05	52.38	66.67	76.19
Myspace	80.95	57.14	73.81	52.38	73.81	78.57
Naruto	73.81	52.38	69.05	52.38	64.29	73.81
Paris hilton	78.57	50.00	76.19	47.62	64.29	76.19
Pokemon	83.33	50.00	76.19	47.62	71.43	78.57
Poker	76.19	50.00	69.05	45.24	64.29	76.19
Rebelde	80.95	54.76	71.43	47.62	73.81	71.43
Rune scape	78.57	54.76	69.05	52.38	66.67	71.43
Second life	78.57	57.14	76.19	50.00	71.43	76.19
Shakira	80.95	52.38	73.81	52.38	66.67	78.57
Sudoku	83.33	59.52	69.05	47.62	71.43	71.43
Tmz	76.19	50.00	73.81	45.24	71.43	71.43
Transformers	80.95	52.38	71.43	52.38	71.43	71.43
Webdetente	83.33	52.38	71.43	45.24	73.81	71.43
Webkinz	80.95	57.14	73.81	52.38	69.05	73.81
World cup	83.33	54.76	73.81	50.00	69.05	73.81
Wwe	80.95	59.52	76.19	50.00	73.81	76.19
Xanga	76.19	57.14	76.19	45.24	73.81	73.81
Youtube	78.57	52.38	78.33	49.76	71.43	73.81
Average	78.51	54.29	72.98	49.27	69.62	74.48

Table 6
Average *P*@3, *P*@5, *P*@7, *P*@10 scores for different online systems.

	OTFDC, %	Credo, %	Carrot2, %	Cats, %	Highlight, %	Vivisimo, %
Average *P*@3	78.51	54.29	72.98	49.27	69.62	74.48
Average *P*@5	74.47	50.27	67.88	45.11	64.49	68.94
Average *P*@7	70.49	44.99	64.15	40.02	60.80	66.77
Average *P*@10	66.08	40.59	58.68	35.88	56.47	62.00

$F_{0.05}(5228) = 2.254$ (*F*-distribution). This provides extremely strong evidence against the null hypothesis, indicating that there is a *significant difference* in the performance of the online systems on the user study.

We then conducted multiple pair-wise comparisons using *Fisher's Least Significant Difference* (LSD) test at the 5% significance level. As illustrated in the results of SPSS, OTFDC was found to overwhelmingly better than Credo, Carrot2, Cats, Highlight, and Vivisimo.

4.2. Determine a suitable threshold value

We need to determine a suitable Threshold (T) value to use in OTFDC. In this simulation, we use DISTance (*DIST*) to measure the average distance of TK_i *and any one of the ECKs* j, shown as the following equation:

$$DIST_j = \frac{\sum_{wp_s} |1/R_{(WP_s \ in \ TK_i)} - 1/R_{(WP_s \ in \ TK_j)}|}{\#WP_s/\#WP_u} \tag{7}$$

Table 7
The distribution of the optimal MMDIST values and its corresponding T values.

#{TK}s	Optimal MMDIST	Optimal T	#{TK}s	Optimal MMDIST	Optimal T
50	3.271	0.800	550	3.276	0.750
100	4.420	0.800	600	3.178	0.800
150	3.638	0.800	650	4.577	0.825
200	3.878	0.775	700	4.012	0.800
250	3.133	0.825	750	4.005	0.750
300	4.474	0.775	800	3.888	0.850
350	4.502	0.825	850	4.574	0.825
400	4.236	0.800	900	4.587	0.775
450	3.130	0.850	950	4.560	0.800
500	4.501	0.775	1000	4.142	0.825

where *WPs* (*WPu*) are the intersection (union) of the top 10 web documents returned from TK_i and TK_j; $R_{(WPs\ in\ X)}$ is the rank of *WPs* in X if $R_{(WPs\ in\ X)} > 0$, and $1/R_{(WPs\ in\ X)} = 0$ if *WPs* cannot be found in X. For example, the web documents returned from TK_i = "p2p" and TK_j = "peer-to-peer" ("en.wikipedia.org/wiki/P2P", "openp2p.com") and ("en.wikipedia.org/wiki/Peer-to-peer", "en.wikipedia.org/wiki/P2P", "openp2p.com"), respectively. The rank of "en.wikipedia.org/wiki/P2P" ("openp2p.com") in "p2p" and "peer-to-peer" are 1 and 2 (2 and 3), respectively. Then, we obtain $DIST_{\text{"peer to peer"}} = (|1/1 - 1/2| + |1/2 - 1/3|)/(2/3) = 1$.

The motivation of T is to minimize the average distance of TK_i and TK_j. Thus, Eq. (7) implies the following three properties: (1) there are more *WPs* than *WPu*, which implies that the web documents returned from TK_i and TK_j are closer to each other; (2) $1/R_{(WPs\ in\ TK_i)} - 1/R_{(WPs\ in\ TK_j)}$ becomes much greater, which implies that the rank of *WPs* appearing in TK_i and TK_j are fairly similar; (3) the top rank of *WPs* have greater impact on $DIST_j$ than the rear rank, which implies that the dominant factor of $DIST_j$ is the rank of *WPs*.

Then, we use Mean of DIST (MDIST) to measure the average distances of *TK_i and all ECKs when the number of TK_i is 1*, as shown in the following equation:

$$\mathrm{MDIST}_{TK_i} = \frac{1}{\|ECK_s\|}\sum_{j=1}^{\|ECK_s\|} DIST_j \tag{8}$$

On the other hand, we also use Mean of MDIST (MMDIST) to judge the average distance of *TK_i and all of the ECKs when the number of TKi is not 1*, as shown in the following equation:

$$\mathrm{MMDIST}_{\{TK\}} = \frac{1}{\|\{TK\}\|}\sum_{j=1}^{\|TK_i\|} \mathrm{MDIST}_{TK_i} \tag{9}$$

Fig. 6 shows a MMDIST curve based on different T values when OTFDC randomly generated 50 {TK}s.[4] We find the minimum value of MMDIST is 3.271 when T attains 0.800. However, the value of MMDIST will be increased if T is greater than 0.800. This happens because the total number of *ECKs* will be proportionately reduced when T is increased. If T is increased to a certain value or greater (the total number of *ECKs* is reduced to a certain value or less), the value of MMDIST will be greatly increased. Hence, the total number of *ECKs* is too small, so that OTFDC has just a few candidate clusters to run OTFDC_Core.

Moreover, we also apply OTFDC using different {TK}s from 50 to 1000. Table 7 shows the optimal MMDIST values and their corresponding T values. For example, we can find the optimal MMDIST value is 3.271 and its corresponding T is 0.800 when the number of {TK}s is 50. In the end, the average over all possible values of the optimal T value is 0.801 for a final T parameter of OTFDC.

4.3. Applying OTFDC to several search engines

The source of OTFDC is a term-document matrix used to form the reverse links of a document. That is, if any one of the search engines has a term-document matrix, it can apply OTFDC to improve the clustering performance. We fetch the web documents by libcurl tool (Stenberg, 2008) and parse them by Sphinx tool (Aksyonoff, 2008); they are returned from Google, Yahoo, and Vivisimo in order to form a term-document matrix.

For the comparison sample, Google and Yahoo are derived from Google AdWords (Google, 2008) and Yahoo Search Marketing (Yahoo, 2008) keyword suggestion tools. The comparison sample of OTFDC-like, Google-OTFDC, Yahoo-OTFDC, and Vivisimo-OTFDC, are the clustering labels which are generated from OTFDC applied to its origin term-document matrix.

In this simulation, we also use MMDIST to judge the clustering performance. Table 8 shows the MMDIST achieved by Google, Google-OTFDC, Yahoo, Yahoo-OTFDC, Vivisimo, and Vivisimo-OTFDC when T is set to be 0.801, as described in previous

[4] In this paper, we randomly selected 1000 queries from metaspy (InfoSpace, 2008) which people are currently searching for in Internet. The 1000 random queries are listed in http://cayley.sytes.net/english/listing_all_testing_keywords.php.

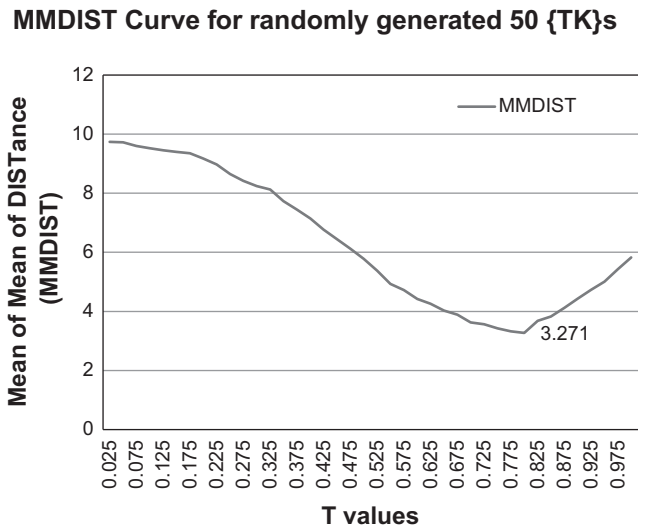

Fig. 6. Mean of Mean of DISTance Curve for 50 {*TK*}s. This curve estimates the average distance TK_i and all of *ECKs* when the number of TK_i is 50.

Table 8
The distribution of MMDIST when OTFDC applied to different search engines ($T = 0.801$).

#{*TK*}s	Google	Google-OTFDC	Improved rate, %	Yahoo	Yahoo-OTFDC	Improved rate, %	Vivisimo	Vivisimo-OTFDC	Improved rate, %
50	4.401	3.371	23.40	4.192	3.621	13.62	3.855	3.484	9.62
100	4.310	3.320	22.97	4.435	3.374	23.92	3.734	3.431	8.11
150	4.246	3.332	21.53	4.523	3.471	23.26	3.814	3.491	8.47
200	3.991	3.465	13.18	4.318	3.439	20.36	3.697	3.582	3.11
250	4.039	3.616	10.47	4.257	3.360	21.07	3.942	3.434	12.89
300	3.930	3.340	15.01	4.322	3.448	20.22	4.051	3.537	12.69
350	4.017	3.397	15.43	4.215	3.562	15.49	3.740	3.707	0.88
400	4.332	3.312	23.55	4.003	3.500	12.57	3.858	3.526	8.61
450	4.304	3.376	21.56	4.357	3.328	23.62	3.767	3.470	7.88
500	4.184	3.386	19.07	3.942	3.492	11.42	3.721	3.538	4.92
550	4.476	3.461	22.68	4.438	3.358	24.34	3.732	3.440	7.82
600	4.178	3.618	13.40	3.959	3.432	13.31	4.057	3.469	14.49
650	3.968	3.485	12.17	4.123	3.444	16.47	3.994	3.689	7.64
700	4.286	3.332	22.26	4.314	3.334	22.72	3.743	3.478	7.08
750	4.033	3.428	15.00	4.137	3.410	17.57	4.098	3.560	13.13
800	4.323	3.614	16.40	4.044	3.460	14.44	3.698	3.594	2.81
850	4.011	3.322	17.18	4.438	3.514	20.82	3.713	3.582	3.53
900	3.991	3.353	15.99	4.178	3.609	13.62	3.912	3.728	4.70
950	4.139	3.536	14.57	4.064	3.454	15.01	3.635	3.716	2.23
1000	4.074	3.519	13.62	3.921	3.597	8.26	4.042	3.599	10.96
Average	4.162	3.429	17.47	4.209	3.460	17.61	3.840	3.553	7.58

studies. Obviously, Google-OTFDC, Yahoo-OTFDC, and Vivisimo-OTFDC outperform their respective origins, Google, Yahoo, and Vivisimo. Specifically, when OTFDC is applied to Google and Yahoo, it can improve the performance of keyword suggestion tools by at least 17.47%. According to the simulation results of Vivisimo and Vivisimo-OTFDC, OTFDC also can improve the clustering performance of Vivisimo by 7.58%. That is, the simulation confirmed the HSR and CHSR feature are superior to the original SR feature.

5. Conclusions

In this paper, we presented a novel clustering algorithm, called *On-The-Fly Document Clustering* (OTFDC), to generate a set of candidate clusters from other web search results. OTFDC has the following four attractive properties. First, it can be applied to multilingual web documents. We have provided English and Chinese clustering search engines to illustrate how OTFDC can be applied to multilingual web documents.

Second, we simulate the combined search engines: ‘‘Google-OTFDC’’, ‘‘Yahoo-OTFDC’’, and ‘‘Vivisimo-OTFDC’’, to verify whether or not OTFDC can improve the clustering performance of any search engines.

Third, the clustering performance of OTFDC is mainly influenced by the search results returned from search engines since OTFDC is a post-processing algorithm. Moreover, OTFDC performs unsupervised learning to automatically identify potentially relevant knowledge without using any corpus since OTFDC does not need any predefined information on the distribution of the search results returned from search engines.

Fourth, although OTFDC is a recursive algorithm, it still generates candidate clusters on the fly in response to a user query. This is because OTFDC uses T to determine whether any two clusters are topically related. T is set to a high value so OTFDC usually needs only a small number of *ECKs* to run OTFDC_Core for each recursive call. This means *ECKs* with a relatively low T value cannot become a source of OTFDC for the next recursive call.

Acknowledgements

We would like to thank the project of libcurl and Sphinx to form the term-document matrix of various search engines. We would like to thank 12 students to evaluate the online questionnaire. We would also like to thank anonymous reviewers of the paper for their constructive comments which help us to improve the paper in several ways. This work was supported in part by National Science Council, Taiwan under Grant NSC 97-2221-E-259-026.

References

Aksyonoff, A. (2008). Sphinx – free open-source SQL full-text search engine. <http://www.sphinxsearch.com/> Retrieved 28.10.08.
Berkhin, P. (2002). Survey of clustering data mining techniques. <http://www.ee.ucr.edu/~barth/EE242/clustering_survey.pdf> Retrieved 28.10.08.
Carpineto, C., & Romano, G. (2004). CREDO. <http://credo.fub.it/> Retrieved 22.09.08.
Carpineto, C., & Romano, G. (2004a). *Concept data analysis: Theory and applications*. John Wiley and Sons Press.
Carpineto, C., & Romano, G. (2004b). Exploiting the potential of concept lattices for information retrieval with CREDO. *Journal of Universal Computer Science, 10*(8), 985–1013.
Chang, Y., Kim, M., & Raghavan, V. V. (2006). Construction of query concepts based on feature clustering of documents. *Information Retrieval, 9*(3), 231–248.
Chavent, M., Lechevallier, Y., & Briant, O. (2007). DIVCLUS-T: A monothetic divisive hierarchical clustering method. *Computational Statistics and Data Analysis, 52*(2), 687–701.
Chen, L. C. (2008a). Cayley search engine. <http://cayley.sytes.net/english/> Retrieved 22.08.09.
Chen, L. C. (2008b). LSD test results. <http://cayley.sytes.net/OTFDC_questionnaire/output.htm> Retrieved 28.10.08.
Chen, L. C., & Luh, C. J. (2005). Web page prediction from metasearch results. *Internet Research: Electronic Networking Applications and Policy, 15*(4), 421–446.
Cios, K. J., Pedrycz, W., Swiniarski, R. W., & Kurgan, L. A. (2007). *Data mining: Knowledge discovery methods*. Springer Press.
Dai, W., Xue, G. R., Yang, Q., & Yu, Y. (2007). Co-clustering based classification for out-of-domain documents. In *Paper presented at the Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*.
Ferragina, P., & Guli, A. (2008). A personalized search engine based on Web-snippet hierarchical clustering. *Software: Practice and Experience, 38*, 189–225.
Fung, B. C. M., Wang, K., & Ester, M. (2003). Hierarchical document clustering using frequent itemsets. In *Paper presented at the proceedings of the 3rd SIAM international conference on data mining*.
Giannotti, F., Nanni, M., Pedreschi, D., & Samaritani, F. (2003). Webcat: Automatic categorization of web search results. In *Paper presented at the proceedings of the 11th Italian symposium on advanced database systems*. Retrieved 09.01.08.
Google, (2007). 2007 Year-End Zeitgeist. <http://www.google.com/intl/en/press/zeitgeist2007/index.html> Retrieved 22.09.08.
Google, (2008). Google AdWords. <http://www.adwords.google.com/select/KeywordToolExternal?defaultView=2> Retrieved 22.09.08.
InfoSpace, (2008). MetaCrawler SearchSpy. <http://www.metacrawler.com/info.metac.spy/searchspy/> Retrieved 15.03.08.
Kanellopoulos, Y., Antonellis, P., Tjortjis, C., & Makris, C. (2007). k-Attractors: A clustering algorithm for software measurement data analysis. In *Paper presented at the proceedings of the 19th IEEE international conference on tools with artificial intelligence*.
Kerne, A., Koh, E., Sundaram, V., & Mistrot, J. M. (2005). Generative semantic clustering in spatial hypertext. In *Paper presented at the proceedings of the 2005 ACM symposium on document engineering*.
Lawrie, D. J., & Croft, B. (2003). Generating hierachical summaries for web searches. In *Paper presented at the proceedings of the 26th annual international ACM conference on research and development in information retrieval*.
Li, Y., Chung, S. M., & Holt, J. D. (2008). Text document clustering based on frequent word meaning sequences. *Data and Knowledge Engineering, 64*(1), 381–404.
Lycos, (2007). Top search terms for 2007. <http://50.lycos.com/121007.asp> Retrieved 22.09.08.
MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Paper presented at the proceedings of the 5th Berkeley symposium on mathematical statistics and probability*.
Malerba, D., Esposito, F., & Ceci, M. (2002). Mining HTML pages to support document sharing in a cooperative system. *Lecture Notes in Computer Science, 2490*, 794–798.
NISO, (2005). *ANSI/NISO Z39.19-2005 guidelines for the construction, format, and management of monolingual controlled vocabularies*. NISO Press.
Osinski, S., & Weiss, D. (2004). Conceptual clustering using lingo algorithm: Evaluation on open directory project data. In *Paper presented at the proceedings of new trends intelligent information processing and web mining*.
Osinski, S., & Weiss, D. (2005). A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems, 20*(3), 48–54.
Radovanović, M. (2006). CatS categorized search. <http://stribog.im.ns.ac.yu/cats/index.html> Retrieved 22.09.08.
Radovanović, M., & Ivanović, M. (2006). CatS: A classification-powered meta-search engine. *Advances in Web Intelligence and Data Mining, 23*, 191–200.
Saraçoğlu, R., Tütüncü, K., & Allahverdi, N. (2007). A fuzzy clustering approach for finding similar documents using a novel similarity measure. *Expert Systems with Applications: An International Journal, 33*(3), 600–605.
Saremi, S. (2007). Top search terms for 2007. <http://www.searchviews.com/index.php/archives/2007/12/top-search-terms-for-2007.php> Retrieved 22.09.08.
Segev, A., Leshno, M., & Zviran, M. (2007). Context recognition using internet as a knowledge base. *Journal of Intelligent Information Systems, 29*(3), 305–327.
Stenberg, D. (2008). Libcurl – The multiprotocol file transfer library. <http://curl.haxx.se/libcurl/> Retrieved 22.09.08.
Szczepaniak, P. S., Segovia, J., Kacprzyk, J., & Zadeh, L. A. (2003). *Intelligent exploration of the web*. Physica-Verlag GmbH Press.
Tjhi, W.-C., & Chen, L. (2007). Possibilistic fuzzy co-clustering of large document collections. *Pattern Recognition, 40*(12), 3452–3466.
Vivisimo, (2004). Clusty the clustering search engine. <http://www.clusty.com/> Retrieved 22.09.08.
Voorhees, E. M. (1986). Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. *Information Processing and Management, 22*(6), 465–476.
Weiss, D., & Osinski, S. (2004). Carrot clustering engine. <http://www.demo.carrot2.org/> Retrieved 22.09.08.
Weiss, D., & Stefanowski, J. (2003). Web search results clustering in Poish: Experimental evaluation of Carrot. In *Paper presented at the proceedings of the new trends in intelligent information processing and web mining conference*.

Wu, B. (2003). Highlight: Concept hierarchy developer. <http://highlight.njit.edu/> Retrieved 22.09.08.
Wu, Y.-F., & Chen, X. (2003). Extracting features from Web search returned hits for hierarchical classification. In *Paper presented at the proceedings of the international conference on information and knowledge engineering*.
Yahoo, (2008). Start advertising with Yahoo! search marketing. <http://wwwsignup13.marketingsolutions.yahoo.com/signupui/signup/loadSignup.do> Retrieved 22.09.08.
Yao, Y. Y. (1995). Measuring retrieval effectiveness based on user preference of documents. *Journal of the American Society for Information Science, 46*(2), 133–145.
Zamir, O., & Etzioni, O. (1998). Web document clustering: A feasibility demonstration. In *Paper presented at the proceedings of the 21st international ACM SIGIR conference on research and development in information retrieval*.
Zha, H., He, X., Ding, C., Simon, H., & Gu, M. (2001). Bipartite graph partitioning and data clustering. In *Paper presented at the proceedings of the 10th international conference on information and knowledge management*.
Zhang, D., & Dong, Y. (2004). Semantic, hierarchical, online clustering of web search results. *Lecture Notes in Computer Science, 3007*, 69–78.
Zipf, G. K. (1949). *Human behavior and the principle of least effort: An introduction to human ecology*. Addison Wesley Press.

Lin-Chih Chen is an assistant professor in the Department of Information Management at National Dong Hwa University, Taiwan. His research interests include Web Intelligent and Web Technology. He develops many Web Intelligent systems include Cayley Search Engine, On-The-Fly Document Clustering, Cayley Digital Content system, iClubs Community, and Language Agent.