國 立 交 通 大 學

資訊管理研究所

碩　士　論　文

工作階段知識支援：探勘與支援工作相關知識

Task-stage *K*-Support: Mining and Supporting Task-relevant Knowledge

研 究 生： 陳 韋 孝

指導教授： 劉 敦 仁 博士

中華民國九十四年七月

工作階段知識支援：
探勘與支援工作相關知識

Task-stage *K*-Support: Mining and Supporting Task-relevant Knowledge

研 究 生：陳 韋 孝　　**Student:** Wei-Hsiao Chen

指導教授：劉 敦 仁　　**Advisor:** Duen-Ren Liu

國立交通大學

資訊管理研究所

碩士論文

A Thesis

Submitted to Institute of Information Management

College of Management

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Business Administration

in

Information Management

July 2004

Hsinchu, Taiwan, the Republic of China

中華民國九十四年七月

Task-stage $\mathcal{K}$-Support: Mining and Supporting Task-relevant Knowledge

Student: Wei-Hsiao Chen **Advisor:** Duen-Ren Liu

Institute of Information Management

National Chiao Tung University

Abstract

Organizations mainly conduct knowledge-intensive tasks to achieve organizational goals. In such task-based environments, knowledge workers usually accomplish their tasks by stages, while their task-needs may be different at various stages of task performance. Accordingly, an important issue of deploying Knowledge Management Systems is to extract knowledge from the historical task and further support task-relevant knowledge according to workers' task-needs at different task-stages. This work proposed a task-stage mining method for discovering task-stage needs from historical task executions. The proposed method adopts information retrieval techniques and a modified hierarchical agglomerative clustering (HAC) algorithm to identify task-stage needs by analyzing codified knowledge (documents) accessed or generated during task performance. Task-stage profiles are generated to model workers' task-stage needs and are used for delivering task-relevant knowledge at various task stages. Finally, we conduct empirical evaluations to demonstrate that the proposed method provides effective knowledge support based on task-stages.

Keywords: Knowledge management, Knowledge support based on task-stage, Hierarchical agglomerative clustering, Text mining, Task-stage profile

工作階段知識支援：
探勘與支援工作相關知識

研究生：陳 韋 孝　　　　　　　　　　　指導教授: 劉敦仁 博士

國立交通大學資訊管理所

摘要

組織主要是處理知識密集工作來達成組織目標，而在工作為基礎的環境中，知識工作者常常以階段性完成工作。在工作執行過程中，不同階段有不同的工作需求。因此如何根據知識工作者在不同工作階段之需求，提供工作相關知識，為建構知識管理系統之重要議題。本研究改良階層式凝聚分群演算法，提出一工作階段探勘方法，從歷史工作資料發掘工作階段知識需求。依據所萃取之工作階段知識需求，進而建構工作階段需求特徵檔，提供知識工作者執行工作時各階段之相關知識。本文最後進行實驗比較以評估所提的方法在提供知識支援之有效性。

關鍵字：知識管理、工作階段知識支援、階層式凝聚分群、文件探勘、工作階段需求特徵檔

誌謝

首先非常感謝指導教授劉敦仁老師，在這兩年來給予許多建議與指導，不管是學業上或是論文研究上，使我能夠順利完成碩士學位。也要感謝口試委員李瑞庭博士與楊千博士在口試時給予許多寶貴的意見，讓本篇論文得以更加完善。

另外也要感謝同實驗室的學長姐們，美玉、志坤、孟蓉、怡瑾、嘉源、昆學、錦慧，在生活上與課業上對我的提攜與幫助，尤其特別感謝怡瑾學姊與昆學學長，在平常與非常時期一直對我非常的照顧；感謝實驗室的同學們，桄瑋、秋雯、柏村，一起研究課業，分享學習心得，共同分擔在學期間所遭遇到的問題。

最後，要感謝我的家人與朋友，支持我往自己感興趣的方向發展，讓我得以順利完成碩士學位。

Table of Content

Table of Content iv

1. Introduction **1**
1.1. Research Background & Motivation 1
1.2. Research Objectives 2
1.3. Thesis Organization 3

2. Related Works 4
2.1 Task-based Knowledge Management 4
2.2 Information Retrieval and Information Filtering 4
2.3 Document Clustering 6

3. Task-stage Knowledge Support 8
3.1. Problem Formulation 8
3.2. Process of Mining Task-stage Needs 9
3.3. Term Definition 10

4. Mining of Task-stage Needs **12**
4.1. Data Pre-processing 12
4.2. Process of Mining Task-stage Needs 13

5. An Illustrative Example and Discussion **20**
5.1. Example 1 20
5.2. Example 2 22

6. Experiments **25**
6.1. Experimental Setup 25
6.2. Experimental Result 27

7. Conclusions and Future Works **39**
References **39**
Appendix A1 .... **42**
Appendix A2-1 .... **43**
Appendix A2-2 .... **44**

1. Introduction

1.1. Research Background & Motivation

In task-based business environments, an important issue in deploying Knowledge Management Systems (KMSs) is providing task-relevant information to fulfill the information needs of knowledge workers. As the operations and management activities of enterprises are mainly task-based, Knowledge Management Systems (KMSs) focus on providing task-relevant knowledge to workers engaged in knowledge-intensive tasks (Abecker et al., 2000; Fenstermacher, 2002; Fischer & Ostwald, 2001; Staab & Schnurr, 2000). The KMS developed in the *KnowMore* project is an example of proactive delivery of task-specific knowledge according to the contexts of tasks within a process (Abecker et al., 2000). The *Kabiria* system supports knowledge-based document retrieval in office environments, allowing users to conduct document retrieval according to the operational context of task-associated procedures (Celentano et al., 1995)

Among different type of knowledge, the repository of structured and explicit knowledge, especially in document form, is a codified strategy for managing knowledge (Davenport & Prusak, 1998; Wei et al., 2004; Zack 1999). Empirical findings indicate that codifying intellectual content into a knowledge repository makes workers highly exploit existing organizational resources (Gray, 2001). The Gartner Group (1999) noted that knowledge retrieval is a core component of KMSs.

This work focuses on providing knowledge support for knowledge-intensive tasks within organizations. Examples of knowledge-intensive tasks include thesis works and research projects in academic organizations, project management in firms, research work and product development in R&D departments, and the like. In such task-based environments, reusing knowledge assets extracted from historical task executions is the key to providing effective knowledge support for conducting tasks. In our previous study, we built a knowledge support system from the perspective of task execution. Information Retrieval techniques are adopted to generate task profiles from text documents accessed and generated by historical task executions. The task profiles model workers' task needs, and can be used as the kernel of knowledge retrieval to provide needed task relevant knowledge during task performance. Moreover, the system identifies task-based peer groups to provide a communication portal of knowledge sharing among knowledge workers. Consequently, knowledge reuse can be achieved through knowledge retrieval and knowledge sharing (Liu et al., 2005). Figure 1 illustrates the concepts of our proposed task-based organizational environments (Yang, 2004).

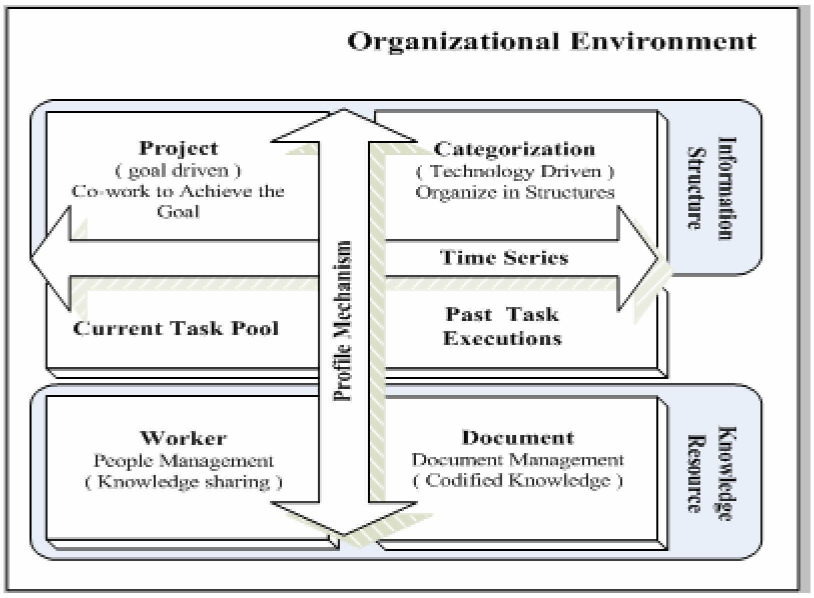


Figure 1: Concept of Task-based Organizational Environment

Moreover, we found that knowledge workers usually require a longer time to accomplish knowledge-intensive tasks. For such long-term tasks, the information needs of knowledge workers may vary according to different stages of progress towards task performance. Namely, with the growth understanding of perceived task, the worker's information needs and information seeking activates will change throughout various stages of task performance. This observation is in accordance with the pilot studies of task-based information retrieval (Kuhlthau, 1993; Vakkari 2000, 2003). Therefore, the aim of this work is to support task-relevant knowledge according to workers' task-needs at different task-stages.

1.2. Research Objectives

This work investigates the issues of deploying knowledge support systems based on task-stages. The research directions mainly include:

- Investigating the characteristics of knowledge-intensive tasks, and further proposing a system framework of supporting task-relevant knowledge based on task-stages;
- Employing data mining and information retrieval techniques to analyze workers' task-needs, which are modeled as profiles, at different task-stages;
- Conducting the experiments to evaluate the effectiveness of the proposed methods.

Our previous work did not consider knowledge workers' stages of progress towards task performance. Accordingly, this work proposed a task-stage mining method for discovering task-stage needs from historical task executions. The proposed method adopts information retrieval techniques and a modified hierarchical agglomerative clustering (HAC) algorithm to identify task-stage needs by analyzing codified knowledge (documents) accessed or generated during task performance. Task-stage profiles are generated to model workers' task-stage needs and are used for delivering task-relevant knowledge at various task stages. A ***task-stage profile*** specifies the key subjects of a task stage, and is constructed to model the information needs of knowledge workers during a task stage of task execution. Finally, we conduct empirical evaluations to demonstrate that the proposed method provides effective knowledge support based on task-stages. Experiments are executed to verify the effectiveness of the proposed task-stage method compared with task-based, called non-stage method.

1.3. Thesis Organization

This paper is organized as follows: The literature review is given in Section 2. Section 3 describes the process of task-stage knowledge support. The method of mining task stage needs is described in Section 4. An illustrative example to explain the proposed idea is presented in Section 5. The experimental evaluations are presented in Section 6. Conclusions and future works are finally made in Section 7.

2. Related Work

2.1 Task-based Knowledge Management

Managing Knowledge within and across organizations is considered as a prominent activity for creating sustainable competitive advantages in today's business environments. Knowledge Management (KM) is a cycle, sometimes repeated process, which generally includes creation, management and sharing activities (Davenport & Prusak 1998; Gray 2001; Wiig 1993). The operations and management activities of enterprises are mainly based on tasks, in which organizational workers perform various tasks to achieve business goals. In task-based business environments, an important issue of deploying KMS is how to support task-relevant knowledge by considering the characteristics of tasks in organizations (Abecker et al., 2000; Fenstermacher, 2002; Fischer & Ostwald, 2001; Liu ea al, 2005, Staab & Schnurr, 2000).

Organizations mainly conduct knowledge-intensive tasks to achieve organizational goals. In such task-based environments, knowledge workers usually accomplish their tasks by stages, while their task-needs may be different at various stages of task performance. Accordingly, another challenge of deploying KMS is how to support task-relevant knowledge according to workers' task-needs at different task-stages. The Vakkari studies (2000), which focus on a user's information seeking activities during task performance (e.g., writing a proposal, completing a project, etc.), show that information needs vary according to different task stages. Namely, with the growth understanding of perceived task, the worker's information needs and information seeking activates will change throughout various stages of task performance. For example, the type of information needed may vary from general to specific information, and the choice of search terms may vary from broader terms to specific terms.

Accordingly, the aim of this work is to support task-relevant knowledge according to workers' task-needs at different task-stages by the proposed mining technique. Furthermore, this work tries to explore the possible application to adopt the proposed task-stage knowledge support method in knowledge-intensive tasks, such as project management activities, academic research, and industry analysis.

2.2 Information Retrieval and Information Filtering

Information retrieval (IR) deals with the representation, storage, organization of, and access to information items (Baeza-Yates & Ribeiro-Neto, 1999). In the past twenty years, Information Retrieval technology mainly focused on two dimensions: indexing and searching from a large number of documents. Document pre-processing is important before indexing documents, including term transformation and term weighting (Salton, 1988). Term transformation includes case-folding, stemming and stop word omission or further thesaurus substitution. By the empirical study of Salton (1988), the most effective term-weighting approach considers three factors, which are term frequency, inverse document frequency and normalization. The vector model accomplishes the term weighting work by assigning non-binary term weight to each index term in documents (Baeza-Yates & Ribeiro-Neto, 1999). It is easy to represent documents in the form of vectors in vector space through the document preprocessing. Usually, users accomplish search by conducting query, represented as a set of keywords. The keyword set represents the user information needs. The query vector can be defined as $\vec{q} = \{w_{1,q}, w_{2,q,} \ldots, w_{n,q}\}$ and the document vector can be defined $\vec{d}_j = \{w_{1,d_j}, w_{2,d_j}, \ldots, w_{n,d_j}\}$ where n is the total index terms in the system. Documents can be retrieved and presented to users according to the degree of similarity calculated by cosine similarity measure.

In the past few years, research subjects discussed in the field of Information Retrieval were more general, like Document Classification, Document Categorization, User Modeling, Information Filtering, User Interfaces and so on. The Information Filtering technology is acknowledged to be an effective way to reduce the information overload and provide personalized information. With the dynamically changed environments, uncertainties associated with changing information needs of the user and the dynamic information stream must be handled efficiently. Information filtering stresses on maintaining a promising user profile, in which the system routes the proper information relevant to the user profile (Baeza-Yates & Ribeiro-Neto, 1999; Shapira, Shoval & Hanani, 1999). Information filtering technology relies on the support of traditional kernel technology of information retrieval, but this technology put more emphasis on methods to maintain and learn user profiles to find relevant documents (Baeza-Yates & Ribeiro-Neto, 1999; Widyantoro, Ioerger, &Yen, 2001). Various method for learning user interests or preferences from text documents or Web pages has been proposed in real-world applications in recommender systems, for example: e-mail-filtering systems (Mostafa et al., 1997), personalized online newspaper (Billsus & Pazzani, 1999), and so on.

Information retrieval and information filtering technologies applied in document management systems are generally the first pace of knowledge management initiatives, since textual data such as articles, reports, manual, know-how documents and so on are treated as valuable and explicit knowledge within organizations (Nonaka, 1994). Information retrieval and information filtering are considered as the core techniques to achieve knowledge retrieval. In addition, information retrieval provides not only text processing technique, but also document classification technology to help organizations collect and process documents to achieve the goal of knowledge reuse (Sebastiani, 1999). With the evolution of information filtering, it not only reduces the problem of information overloading but also provides relevant and needed information to users to accomplish their tasks. Accordingly, maintaining and learning user profiles is important in modern Knowledge Management Systems.

2.3 Document Clustering

Clustering is an unsupervised process of grouping the data (objects) into classes or clusters. The data (objects) within a cluster have similarity in comparison to one another, whereas dissimilar to data (objects) in other clusters (Han and Kamber ,2000; Jain et al., 1999). The clustering technique have been addressed in many contexts and by researchers in many domains such as pattern recognition, text categorization, image processing, market research and so on. Therefore, clustering is a useful tool to analyze data items and has been applied broadly.

Generally, clustering techniques can be divided into two classes: one is partitioning clustering and the other is hierarchical clustering (Jain et al., 1999). Partition method outputs only one partition while hierarchical method produces a nested partitions. *K-means* clustering which is commonly used to partition a set of data into groups is one of examples of partition method (MacQueen, 1967). The clustering procedure follows a way to classify the given data (object) through a certain number of clusters, i.e. k clusters. The main idea is first to define k centroids, one for each cluster. After k new centroids have been selected, a loop has been generated. The algorithm assigns a data sample d_i to the cluster c_j such that the distance from d_i to the center of c_j is the minimum over all k clusters. The k centroids will change their location step by step until no more changes are done. The hierarchical clustering method creates a hierarchical decomposition of the given data set (Johnson, 1967). From the viewpoint of algorithmic structure and operation, there are two types of clustering approach, one is agglomerative and another is divisive. The agglomerative method starts with each data item as singleton cluster. It continuously merges clusters together according to the merging criterion until a stopping condition is satisfied. The divisive method starts with all data items as only one cluster. It divides until coming to a stopping condition. Since we adopt Hierarchical agglomerative clustering (HAC) in this work, we take length to introduce the method.

Hierarchical agglomerative clustering (HAC) is a class of clustering approaches. This method produces clusters by merging, depended upon similarity of two existing clusters. The process repeats combining the two most proximity clusters; likes a binary tree, starts at leaves and grows up to root. The operations of HAC can be summarized into three steps as the following.

(1) Each data item is considered as a singleton item initially

(2) Select the nearest pair of clusters and merge them

(3) Repeat step 2 until meeting the stopping condition

In step 2, we have to find the nearest pair. Therefore, a similarity measure has to be chosen. The similarity between any pair of cluster Ci and Cj, there are four well-known inter-cluster similarity function can be used to measure the similarity.

- The *single-linkage function* computes the largest similarity between two objects in the pair of clusters.
- The *complete-linkage function* computes the smallest similarity between two objects in the pair of clusters.
- The *average-linkage function* computes the average of all similarities among all objects in the pair of clusters.
- The *centroid function* computes the similarity between the centroid of the two clusters.

Generally, the single-linkage method produces isolated and may not be cohesive clusters. Inversely, the complete-linkage method produces cohesive and may not be isolated clusters. The average-linkage method is a trade-off between single-linkage and complete-linkage method. Another commonly used measure is the centroid method, especially for text clustering.

In this work, data items are time-variant. In order to consider the time factor, we modified the traditional HAC algorithm by setting a time window to select candidates for merging. Only the clusters in the time window can be merged (perform the merging operation).

3. Task-stage Knowledge Support

Section 3.1 presents the problem formulation. Section 3.2 describes an overview of the process of mining task-stage needs. The terms and notations used in this work are defined in Section 3.3.

3.1. Problem Formulation

This work broadly refers to a task as a unit of work in organizations, such as a project, research work, or activity. Moreover, this work uses profiling approach to model workers' task-stage needs, namely information needs (profiles) on task-stages. Task-stage profiles are generated to model worker's task-stage needs and are used further to provide task-relevant knowledge at various task stages. A ***task-stage profile*** specifies the key subjects of a task stage and can be represented as a feature vector of weighted terms. The profile is used to retrieve relevant codified knowledge in the repository. The key contents of codified knowledge (textual data; documents), namely document profiles, are also represented as a feature vector of weighted terms. Relevant documents can be retrieved to provide knowledge support for task execution according to the similarity measures (e.g. cosine measures) between task-stage profiles and document profiles.

A task class is defined to specify a type of tasks with similar properties (similar tasks, for brevity). Task classes are identified based on existing tasks. A task that belongs to a task class is called a task instance of the task class. Task class/instance resembles the concept of object class/instance. Tasks in different task classes generally have different task-stage needs, while tasks belonging to the same task class may have similar task-stage needs. Accordingly, task-stage profiles generated by analyzing existing similar tasks of the same task class are useful to provide needed relevant knowledge in supporting the execution of on-going tasks.

The term *virtual task* is used to represent a task class of similar tasks. A virtual task may have more than one task instances. For example, a virtual task "Research on recommender systems" has several task instances including "Hybrid approaches for product recommendations" and "Comparisons of collaborative filtering for recommendations". Each existing task instance is associated with documents accessed/generated during task performance. Accordingly, the problem of identifying task-stage needs of a virtual task is described as follows.

Given a virtual task and a set of task instances with associated documents, find the number of task stages and the corresponding profile of each task-stage.

3.2. Process of Mining Task-stage Needs

This work employs information retrieval techniques to conduct text processing and data mining techniques to analyze workers' task-needs, which are modeled as profiles at different task-stages. Figure 2 illustrates the process of mining task-stage needs. The operational procedure is described as follows.

The document database and the system log record the historical tasks with associated documents and usage data. The documents of a task are preprocessed and organized as a task document sequence (TDS) according to their accessed/generated time during task performance. Notably, each task has its own task document sequence.

Clustering and information retrieval techniques are then employed to cluster the task document sequences of similar tasks belonging to the same task class. A task-stage hierarchical clustering *(TSHC)* algorithm is proposed to cluster the task document sequences based on similarity measures and retrieval time of documents. Each cluster represents a task stage with associated documents of a task. More details are addressed in Section 4.2. Finally, the feature vector of each task stage is extracted from each cluster of documents to construct the task-stage profile. The task-stage knowledge retrieval module can then retrieve relevant codified knowledge (documents) to provide knowledge support for task execution according to the similarity measures (e.g. cosine measures) between task-stage profiles and document profiles.

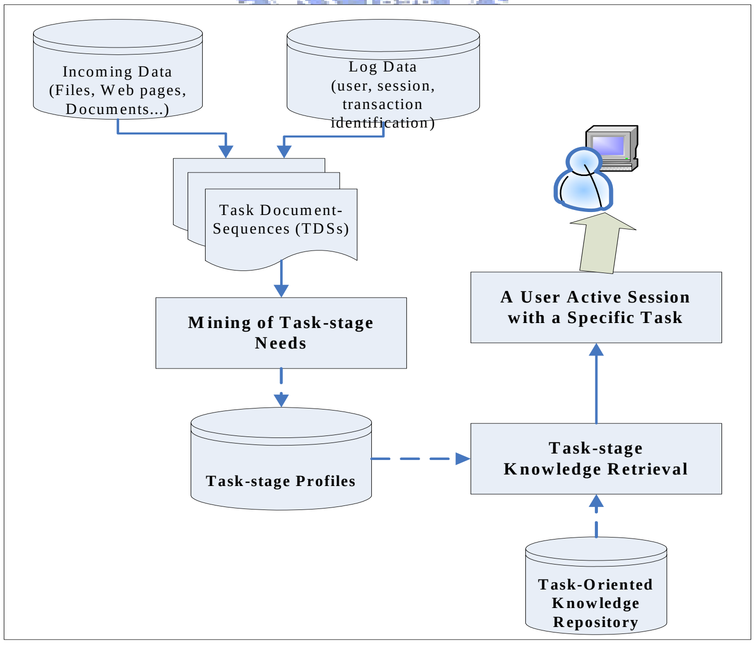

Incoming Data (Files, Web pages, Documents...)
Log Data (user, session, transaction identification)
Task Document-Sequences (TDSs)
Mining of Task-stage Needs
Task-stage Profiles
A User Active Session with a Specific Task
Task-stage Knowledge Retrieval
Task-Oriented Knowledge Repository

Figure 2. Process of discovering task-stage needs

3.3. Term Definition

This section lists the definitions of terms used in this work.

(1) Definition I: Task

The task is the fundamental unit in business. This work broadly refers to a task as a unit of work in organizations, such as a project, research work, or activity.

T: task set	A task is either an executing-task (on-going task) or an existing task in the task-based working environment. T = {t_1, t_2,…, t_r, …, t_n}
$\boldsymbol{t_r}$**:** existing-task	An existing-task is a historical task accomplished within the organization.
$\boldsymbol{t_v}$**:** virtual task	A virtual task represents a task class of similar tasks
task instance	A task instance of t_v is an existing-task that belongs to the class of t_v.
$\mathbf{T}_v$**:** set of task instances	A set of task instances of the virtual task t_v

(2) Definition II: Task Document Sequence

A task document sequence is a sequence of documents retrieved (accessed or generated) during task performance. Documents are sorted according to their retrieval time. Each task instance has its own task document sequence.

TDS($\boldsymbol{t_r}$**):** Task Document Sequence of t_r	A sequence of documents accessed/generated while conducting a task t_r. TDS(t_r)=<d_1, d_2,…,d_m>

(3) Definition III: Task Stage

Task document sequences of task instances are clustered into stages based on similarity measures and retrieval time of documents. The clustering result forms task stages of a task.

TS($\boldsymbol{t_r}$**):** task-stages of a task instance	A task t_r comprises several task stages. **TS**(t_r)=<ts_r[1], ts_r[2],…, ts_r[k]>, where ts_r[i] denotes the task-stage i of t_r .
TS($\boldsymbol{t_v}$**):** task-stages of a virtual task	Task-stages of a virtual task t_v which are derived from task stages of task instances in T_v. **TS(**t_v**)**=<ts_v[1], ts_v[2],…, ts_v[k] >

(4) Definition IV: Task Stage Documents and Profiles

Each cluster of documents represents a task stage with associated documents, named task stage documents. Documents in task stage k of task t_r is denoted as ts_r[k].docs, while documents in task stage k of virtual task t_v is denoted as ts_v[k].docs. Each task-stage profile can be derived from the feature vectors of documents in each task-stage. Let ts_r[k].profile denote the profile of task stage k of task t_r, and ts_v[k].profile denote the profile of task stage k

of virtual task t_v.

ts**$_r$**[*k*].docs: task-stage documents	Documents in task stage *k* of task t_r.
d.profile	The feature vector (profile) of document *d*.
ts**$_r$**[*k*].profile: task-stage profile	The profile of task stage *k* of task t_r. A task-stage profile is the vector obtained by averaging the feature vectors of documents in ***ts**$_r$**[*k*].docs***.
ts**$_v$**[*k*].docs: task-stage documents	Documents in task stage *k* of virtual task t_v.
ts**$_v$**[*k*].profile: virtual task-stage profile	The profile of task stage *k* of virtual task t_v. A virtual task-stage profile is the vector obtained by averaging the feature vectors of documents in ***ts**$_v$**[*k*].docs***.

4. Mining of Task-stage Needs

For mining the task-stage needs, Section 4.1 presents how to collect and analyze the contents of documents and workers' usage data. Section 4.2 illustrates the proposed task-stage hierarchical clustering *(TSHC)* algorithm that is used to cluster the task document sequences based on similarity measures and retrieval time of documents.

4.1. Data Pre-processing

The data pre-processing phase is the most fundamental step of knowledge discovery process. Two types of valuable information sources of knowledge acquisition: content data and usage data are addressed as follows.

Content Data: The key contents of a codified knowledge item (document) can be represented as a feature vector of weighted terms in *n*-dimensional space. The term transforming and term weighting steps are employed to find the most discriminating words among a set of documents (Baeza-Yates & Ribeiro-Neto 1999). The term transforming step includes case folding, stemming, and stop word removing. The *tf-idf* approach is a well-known approach for term weighting in the information retrieval literature (Salton 1988). Let the term frequency $tf_{i,d}$ be the occurrence frequency of term kw_i in d, and let the document frequency df_i represent the number of documents that contain term kw_i. The importance of term kw_i to a document d, $w_{i,d}$ is proportional to the term frequency and inversely proportional to the document frequency, which is expressed as Eq. 1. According to the empirical experiments of Salton (1988), if there are many technical vocabularies in the data collection and the vocabulary is not very varied, normalized term frequency is a proper term weighting component. We use the normalized term frequency to derive the term weight, as shown in Eq. 1.

$$w_{i,d} = (0.5 + \frac{0.5 \times tf_{i,d}}{\max tf_{i,d}}) \times (\log \frac{N}{df_{i,d}} + 1) \quad \textbf{(1)}$$

where N is the number of documents, and $\log(N/df_i)$ is the inverse document frequency, idf_i. The feature vector of document d is represented as $\vec{d} = <w_{1,d}, w_{2,d}, \ldots, w_{n,d}>$.

Usage Data: Each log record is parsed to extract important information, such as user name, role type, TaskID, DocID, request type and system time. Usage log processing engine is useful to collect and manage task-relevant information by tracking user's document access behaviors in conducting a specific task. Similar tasks of the same task class are identified. Moreover, task document sequences of task instances are generated according to the usage log records. A time interval may be specified to select documents from usage logs. Documents with retrieval time within a specified time interval are collected to generate task document sequences.

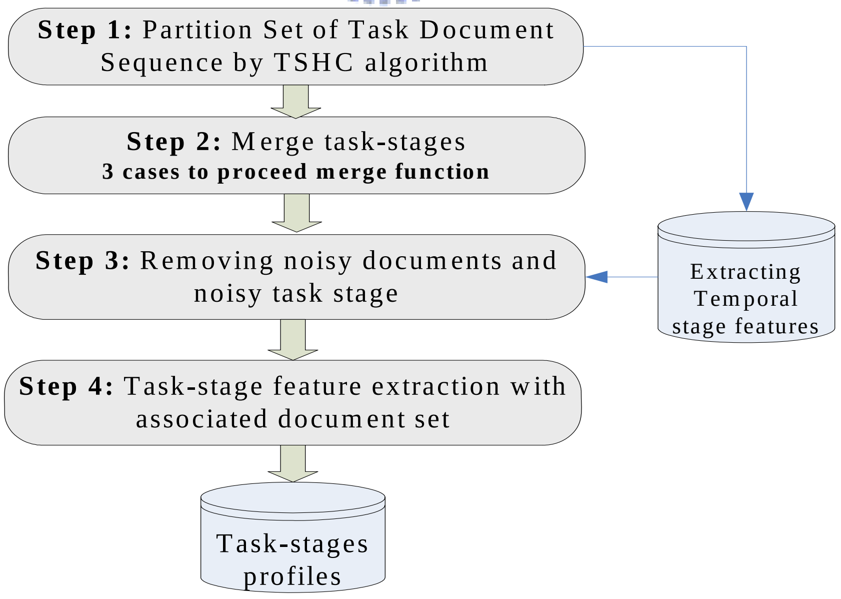
4.2. Process of Mining Task-stage Needs

Given a virtual task t_v and a set of task instances with associated documents, the mining process identifies the number of task stages and the corresponding profile of each task-stage for t_v. Notably, a virtual task t_v denotes a task class of similar tasks and has several task instances. A task instance of t_v is an existing-task that belongs to the class of t_v. The documents of a task instance are preprocessed and organized as a task document sequence (TDS) according to their accessed/generated time during task performance.

There are four steps to discover task-stage needs, as shown in Fig. 3. The mining procedure is described step by step. An example is given to explain the proposed idea.

Step1. Identifying task-stages of each task instance

This step identifies the task-stages of each task instance. A *TSHC* algorithm, task-stage hierarchical clustering algorithm, is proposed to cluster each task document sequence into task stages. The proposed algorithm modifies the hierarchical agglomerative clustering (HAC) algorithm by considering *time variant* to determine clusters. Table 1 shows the *TSHC* algorithm. Let TDS(t_r)=<d_1, d_2,…,d_m> be the task document sequence of a task instance t_r. The task document sequence is clustered into several clusters, which represent the stage information of the target task t_r. A task t_r comprises several task stages. **TS**(t_r)=<ts_r[1], ts_r[2],…, ts_r[k]>, where ts_r[i] denotes the task-stage i of t_r. Each cluster represents a task stage with associated documents, where ts_r[i].docs denotes documents in task stage i of task t_r.



Step 1: Partition Set of Task Document Sequence by TSHC algorithm
Step 2: Merge task-stages
3 cases to proceed merge function
Step 3: Removing noisy documents and noisy task stage
Step 4: Task-stage feature extraction with associated document set
Extracting Temporal stage features
Task-stages profiles

Figure 3: Process of mining task-stage needs

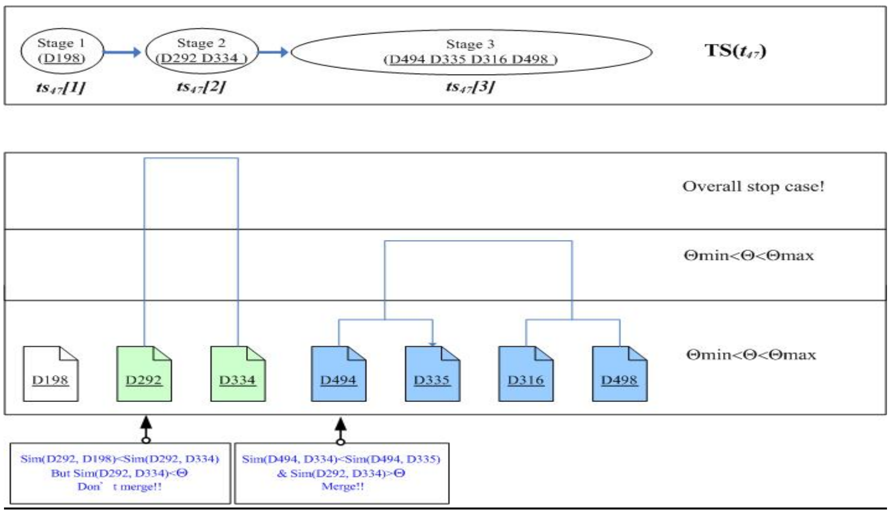


Figure 4: An example of clustering result

Fig. 4 illustrates the clustering result. Three clusters are generated after conducting the proposed clustering algorithm; namely, three stages are identified. In the following, we will explain the *TSHC* algorithm in detail.

TSHC algorithm**:** The traditional Hierarchical Agglomerative Clustering (HAC) algorithms start with each individual item in its own cluster and iteratively merge clusters until all items belong in one cluster. For each level, the HAC algorithms determine new clusters by merging clusters from the previous level. Several techniques can used to determine if two clusters are to be merged according to the distance threshold and the distance between any two points in the two target clusters. Two clusters are merged, if the minimum (maximum or average) distance between any two points in the two target clusters is less than or equal to the distance threshold. The HAC algorithm examines all clusters in the previous level to determine new clusters, and does not consider the time dimension.

The proposed TSHC algorithm adopts the HAC algorithm by considering the time dimension, since the task document sequence (TDS) is generated by ordering documents based on their retrieval time during task performance. A time window size *w* is set to define the scope of candidate clusters to be merged. Notably, the index position in the sequence rather than the actual time is used to decide if the candidate cluster is within the time window.

Initially, each document in the TDS forms its own cluster. The TSHC algorithm then iteratively merges clusters until the number of clusters is less than or equal to *MinNumStages* (minimum number of stages). The algorithm finds clustering result with number of stages no less than *MinNumStages* and no greater than *MaxNumStages* (maximum number of stages). Due to the characteristics of our research problem, we prefer to cluster each *TDS* into three to six clusters. A clustering examination procedure is conducted to select the level of clustering that can derive the best clustering result by intra-similarities and inter-similarities examination.

In the following, we explain the algorithm line by line. Note that the algorithm is listed in Table 1.

During the procedure of clustering, each document is regarded as a single cluster initially. Accordingly, TDS is a sequence of clusters. For merging two target clusters c_i and c_j into a new cluster *c'*, where $i < j$, the index position of *c'* is set to *i*. A new sequence of clusters is generated by ordering the new clusters according to their index positions. A specified time window size *w* is used as the scope to select candidate clusters for merging. For each cluster c_i, clusters with index positions within the time window size from $i - w$ to $i + w$ are the candidates to be merged with c_i (line 3 to 8). For a cluster c_j within the time window of c_i, if the similarity between c_i and c_j is greater than the *Threshold*, c_i and c_j are regarded as a pair of merging candidates; and they are added to the merging list with their similarity measure (line 8). The similarity measure between two cluster c_i and c_j is defined as the average of all pairwise similarities among the documents in c_i and c_j. The similarity between two documents is derived using the cosine of their feature vectors.

The time window is moved from the first cluster to the last one in TDS to determine the merging candidates for each cluster. Then, all merging candidates will be merged as follows. In the merging process (line 9 to 12), the pair of merging candidates with maximum similarity will be selected from the merge list. If both merging candidates in the selected pair had not been merged with another merging candidate in any other pair, then the corresponding two in TDS are merged. If there is no clusters can be merged during the iteration, the *Threshold* is decreased by a constant (In our case, the constant is set to 0.01); otherwise, the threshold is not changed (line 13 to 14). The step avoids that the similarity between a cluster and a merged cluster is higher than the decreased threshold.

The algorithm performs hierarchical clustering; thus various levels of clustering result can be derived. The algorithm records those levels of clustering results with the number of clusters within *MinNumStages* to *MaxNumStages*. As mentioned previously, we prefer to cluster each task document sequence (*TDS)* into three to six clusters. The quality measure, *Q* value, is derived to indicate the quality of each clustering result (line 15 to 17). We select the clustering result (among three to six clusters) that leads to the best quality value. The clusters of *TSHC* algorithm are the stages of a task instance. The profile of each task stage can be derived by averaging the feature vectors of documents in the corresponding cluster. Notably, the *task-stage profile* specifies the key subjects of a task stage.

In the following, we explain the *Q* value that is used verify the quality of clusters.

- **Clustering quality:** Let *C* be the set of clusters derived from a particular level of clustering. The best clustering result is to maximize the intra-similarities of all those clusters and minimize the inter-similarities among the clusters produced at the level (Chuang & Chien, 2004). Let the inter-similarity between two cluster c_i and c_j be defined as the average of all pairwise similarities among the documents in c_i and c_j, which is denoted as *sim(c_i, c_j)*. Let the intra-similarity within the cluster c_i be defined as the average of all pairwise similarities within c_i, denoted as *sim(c_i, c_i)*.

$$Q(C) = \frac{1}{|C|} \sum_{c_i \in C} \frac{sim(c_i, \bar{c}_i)}{sim(c_i, c_i)} \quad where \ \bar{c}_i = \cup_{k \neq i} c_k \tag{2}$$

Table 1. Algorithm of Task Stage Hierarchy-based Clustering with Time-variant

```
Input: TDS(t_r): Task Document Sequence of t_r
Output: Task stages of target task
     function TSHC(TDS) {
1         Set a constant offset be 0.01;
2         do {
3              foreach cluster c_i in TDS {
4                   c_temp = {c_{i-w}, c_{i-w+1}, …, c_{i-1}, c_i , c_{i+1}, …, c_{i+w-1}, c_{i+w}};
                    // According to time window size to determine clusters
5                   foreach cluster c_j in c_temp where c_i <> c_j {
6                        Calculate similarity of c_i and c_j as similarity;
7                        if (similarity is greater than Threshold) then
8                             add {c_i, c_j, similarity} to the merge_list;
                    }
               }
9              do while (merge_list is not empty) {
10                  Select and remove the pair (c_i, c_j) with maximum similarity from merge_list;
11                  if (c_i and c_j had not been merged with another cluster) then
12                       Merge c_i and c_j in TDS
               }
13             if (there's no cluster merged) then
14                  decrease the Threshold by offset;

15             if (number of clusters in TDS ≤ MaxNumStages) then {
16                  Let Q value be the quality of clustering value of TDS;
17                  Add {TDS, Q} to the list result;
               }
          }
18        while (number of clusters in TDS > MinNumStages)

19        return result;
     }
```

Step2. Merging Task-stages.

Step 1 uses the *TSHC* algorithm to generate the task stages of each task instance. This step generates the task stages of the virtual task t_v by merging the stages of task instances of t_v. $\mathbf{T}_v$ is the set of task instances of t_v. A kernel task instance t_r is selected as the representative of the virtual task t_v. Then, the stages of other task instances are merged to the stages of t_r. The kernel task is the task instance with the best average Q value of clusters among task instances. During the merging process, documents from the stages of other task instances are merged to the stages of the kernel task instance.

Task Stage Merging algorithm: The algorithm is shown in Table 2. First, the task instance t_r with the best average Q value of clusters is chosen as the kernel task (line 1). Then, for each other task instance t_f, documents from the stages of t_f are merged to the stages of t_r. Let n_r and n_f be the number of stages (clusters) of t_r and t_f, respectively. Let $ts_f[i]$.docs be the documents in task stage i of task t_f and $ts_r[k]$.profile denote the profile of task stage k of t_r. Notably, $ts_r[k]$.profile is the vector obtained by averaging the feature vectors of documents in $ts_r[k]$.docs.

Generally, several candidate stages in t_r are possible to merge a document of t_f. For each stage i of t_f, each document d in $ts_f[i]$.docs is merged into the candidate stage k of t_r that has the maximum similarity measure $cos(d.\text{profile}, ts_r[k].\text{profile})$. There are three cases to select the candidate stages of t_r for merging a stage i of t_f.

Case 1: $n_f > n_f$: The number of stages of t_f is greater than the number of stages of t_r. The candidate stages of t_r selected are from stages i to $i+(n_r-n_f)$, where $n_r - n_f + 1$ stages of t_f are chosen as the candidate stages. Figure 5 is an illustrative example of Case 1, where $n_f = 3$ and $n_r = 4$. Two stages of t_r are selected as the candidate stages to merge the documents of t_f. For example, stages 2 and 3 of t_r are the candidate stages to merge the documents in stage 2 of t_f.

Case 2: $n_f = n_f$: The number of stages of t_f is equal to the number of stages of t_r. The candidate stages of t_r selected are from stages i-1 to i+1, where three stages of t_f are chosen as the candidate stages.

Case 3: $n_f > n_f$: The number of stages of t_f is less than the number of stages of t_r. The candidate stages of t_r selected are from stages $max(1, i-(n_f-n_r))$ to $min(i, n_r)$. Figure 5-2 is an illustrative example of Case 3, where $n_f = 5$ and $n_r = 3$. For example, stages 1 and 2 of t_r are the candidate stages to merge the documents in stage 2 of t_f. Stages 1 to 3 of t_r are the candidate stages to merge the documents in stage 3 of t_f. Stages 2 and 3 of t_r are the candidate stages to merge the documents in stage 4 of t_f.

Once the stage k with maximum similarity measure is determined, the pair (d, $ts_r[k]$) is added to the merge list indicating that document d will be merged to stage k of t_r (line 7, 14, 20). After the merging stages of all documents from other task instances are decided, the merging process conducts the merge (e.g. adding a document d to $ts_r[k]$.docs) according to the merge list (line 25). The final documents and corresponding task stages of the kernel task instance t_r are regarded as the documents and stages of the virtual task t_v. Note that the number of task stage of t_v is equal to the number of stages of t_r.

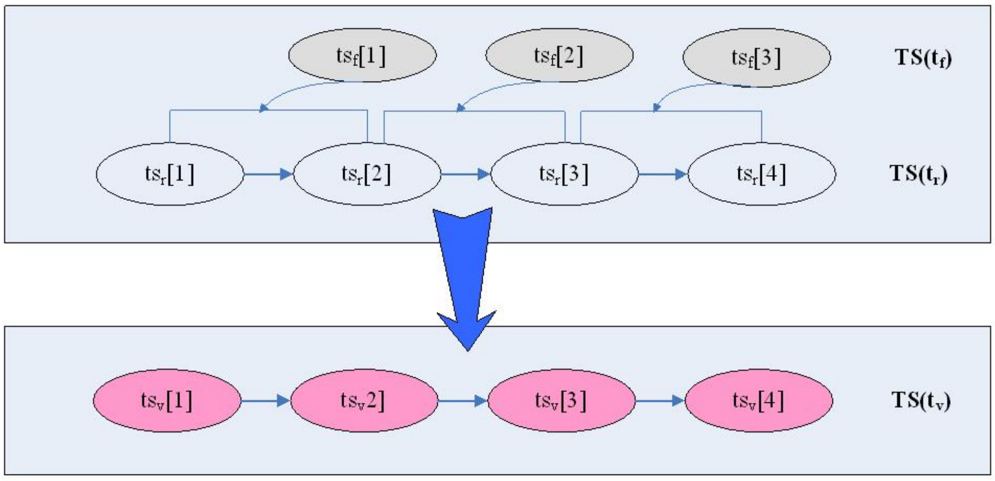


Figure 5: An illustrative example of task stage merge of Case 1.

Table 2. Algorithm of Task Stage Merge

```
Input: $T_v$ // a set of task instances of $t_v$
Ouput: TaskStages of $t_v$ // TS($t_v$)

function TaskStagesMerge($T_v$) {
    Select the kernel task instance which has the best Q value as $t_r$ from $T_v$;
    foreach task $t_f$ in $T_v$ {
        Let $n_r$ and $n_f$ be the number of stages in $t_r$ and $t_f$, respectively.
        case 1: ($n_r > n_f$)
            foreach task stage $i$ in $t_f$ {
                foreach document $d$ in $ts_f[i]$.docs {
                   Add ($d$, $ts_r[k]$) to the merge_list where $cos(d.\text{profile}, ts_r[k].\text{profile})$ is
                   maximum for $i \le k \le i + (n_r - n_f)$
                }
            }
        case 2: ($n_r = n_f$)
            foreach task stage $i$ in $t_f$ {
                foreach document $d$ in $ts_f[i]$.docs {
                   Add ($d$, $ts_r[k]$) to the merge_list where $cos(d.\text{profile}, ts_r[k].\text{profile})$ is
                   maximum for $i - 1 \le k \le i + 1$
                }
            }
        case 3: ($n_r < n_f$)
            foreach task stage $i$ in $t_f$ {
                foreach document $d$ in $ts_f[i]$.docs {
                   Add ($d$, $ts_r[k]$) to the merge_list where $cos(d.\text{profile}, ts_r[k].\text{profile})$ is
                   maximum for $max(1, i-(n_f-n_r)) \le k \le min(i, n_r)$
                }
            }
        }
    }
    Merge all documents in the merge_list to the corresponding task stage;

    return TS($t_f$) as TS($t_v$);
}
```

Step3. Reassigning documents.

This step reorganizes the documents in task stages of t_v to enhance the homogeneousness of each task-stage. Documents may be reassigned to other task stages according to the similarity measures (cosine measures) of document profiles and task-stage profiles. A document *d* is assigned to the stage *k* of t_v that has the maximum similarity measure, i.e., *cos*(*d*.profile, ts_v[*k*].profile), among all task stages of t_v. Some noisy or irrelevant documents may be removed from t_v, if their maximum similarity measures are below a defined threshold. After the reassignment, task stages with zero or one document will be removed. Consequently, we derive the final document set at each task stage of the virtual task t_v. Notably, for some application domains, knowledge workers may need different types (categories) of codified knowledge during a particular stage of task performance. In such scenarios, documents need to be categorized to generate different categories of profiles at each task-stage. Accordingly, the process of mining task-stage needs must be modified to tackle such scenarios. It will be interesting to explore such issues in future work.

Step 4: Extracting Task-stage profiles.

The profile of each task stage can be derived by averaging the feature vectors of documents in the corresponding stage. Notably, the *task-stage profile* specifies the key subjects of a task stage. For a stage *k* of virtual task t_v, a task-stage profile ts_v[*k*].profile is the vector obtained by averaging the feature vectors of documents in **ts_v[*k*].docs**.

4.3. Knowledge Support

Relevant documents can be retrieved to provide knowledge support for task executions according to the similarity measures (e.g. cosine measures) between task-stage profiles and document profiles. For conducting an on-going task t_e, the system first identifies the task class t_v that t_e belongs to. Then, the task-stage profiles of t_v are used to provide task-relevant knowledge at each stage of executing t_e. A document *d* is provided to knowledge workers at the stage of *k*, if the similarity measure *cos*(*d*.profile, ts_v[*k*].profile) is greater than a defined threshold. Alternatively, top-*N* approach can be used to provide *N* documents with *N*-highest similarity measures at each task stage.

5. An Illustrative Example and Discussion

We give two examples to explain the proposed task-stage knowledge support approaches. The first example sets time window to 1 (*w*=1), whereas the second example sets time window=2 (*w*=2). Example one illustrates the case 3 of the task-merge algorithm and example two explains the case 1 of the task-merge algorithm. We explain the proposed approach step by step as follows.

5.1. Example 1

A virtual task 3 named “Task-based knowledge support model and system” is adopted as the illustrative example to explain the proposed idea. Two task instances belonging to this virtual task are listed in Table 3. We will explain how to extract task stage needs according to the proposed task-stage mining approach.

Table 3. Information of the virtual task 3

Virtual task 3: Task-based knowledge support model and system $T_3=\{t_{31},t_{47}\}$		
t_{31}	2003/09/15 ~ 2004/04/13	**A Collaborative Relevance Feedback Approach to Task-driven Recommendation**
t_{47}	2003/09/18 ~ 2004/05/18	**Implementation of Task-based Knowledge Support System**

Step1. Identifying Task-stages. Five and six stages are identified from the task document sequences (TDS) of task 47 and task 31, respectively. The clustering process of task 47 is illustrated in Figure 6.

- **Cluster examination:** We adopted the proposed *TSHC* algorithm listed in Table 1 to conduct the clustering.

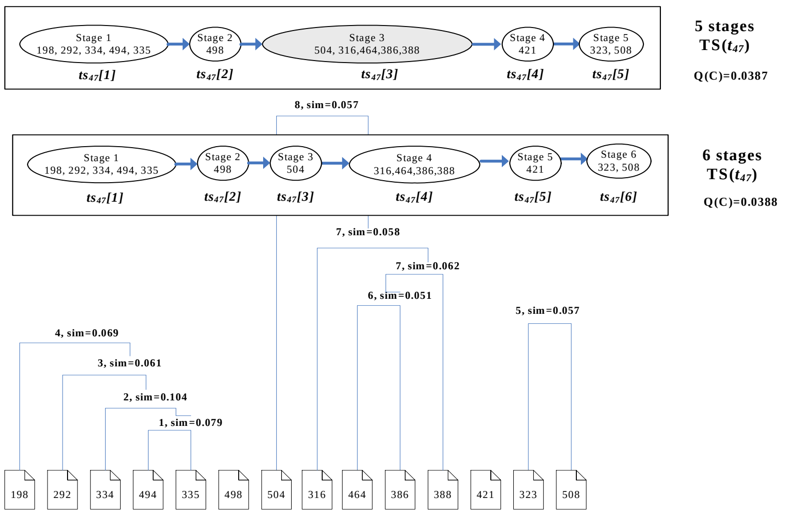
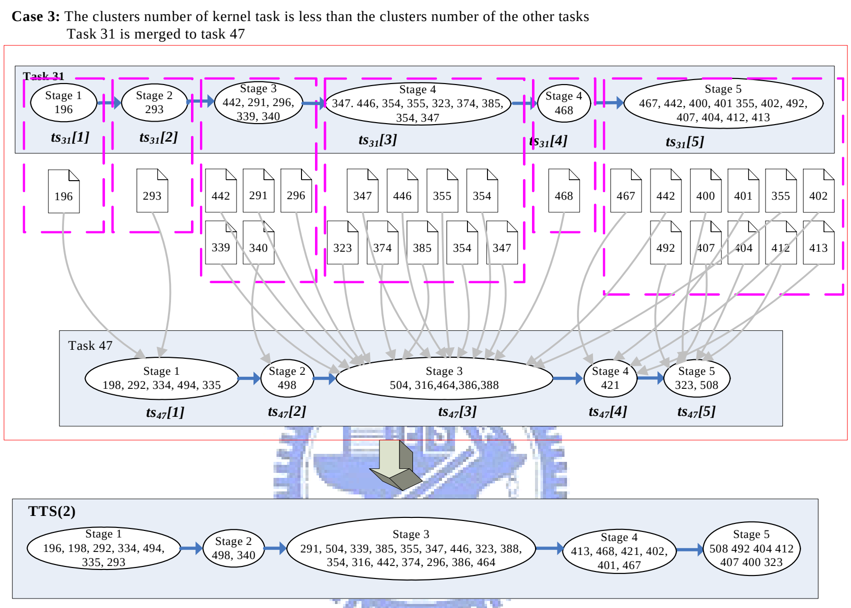


Figure 6: Clustering process of task 47 (Step 1)

Table 4. Clustering examination

Quality value of stages	Task 31	Task 47
Q(C) of 6 stages	**0.168**	0.039
Q(C) of 5 stages	0.191	**0.038**
Q(C) of 4 stages	0.280	0.050
Q(C) of 3 stages	0.478	0.123

Step2. Merging Task-stages. Task 47 is selected as the kernel task, since it has the best *Q* value. The process of merging the stages of task 31 to the stages of task 47 is shown in Figure 7.



Case 3: The clusters number of kernel task is less than the clusters number of the other tasks
Task 31 is merged to task 47
Task 31
Stage 1 196
Stage 2 293
Stage 3 442, 291, 296, 339, 340
Stage 4 347. 446, 354, 355, 323, 374, 385, 354, 347
Stage 4 468
Stage 5 467, 442, 400, 401 355, 402, 492, 407, 404, 412, 413
ts31[1]
ts31[2]
ts31[3]
ts31[4]
ts31[5]
196
293
442
291
296
339
340
347
446
355
354
323
374
385
354
347
468
467
442
400
401
355
402
492
407
404
412
413
Task 47
Stage 1 198, 292, 334, 494, 335
Stage 2 498
Stage 3 504, 316,464,386,388
Stage 4 421
Stage 5 323, 508
ts47[1]
ts47[2]
ts47[3]
ts47[4]
ts47[5]
TTS(2)
Stage 1 196, 198, 292, 334, 494, 335, 293
Stage 2 498, 340
Stage 3 291, 504, 339, 385, 355, 347, 446, 323, 388, 354, 316, 442, 374, 296, 386, 464
Stage 4 413, 468, 421, 402, 401, 467
Stage 5 508 492 404 412 407 400 323

Figure 7: Task stage merge process (Step 2)

Step3. Reassigning documents. The reassigned result is shown in Table 5. Notably, in this case, no document or stage is removed. The details of stage document set are listed in Appendix A(2).

Step 4. Extracting Task-stages profiles. Finally, the feature set of each task stage is extracted from each cluster.

Table 5. Final document set at each task stage of the virtual task 3 (Step 3)

TS(t_3)	
ts_3[1].docs	D0000000335 D0000000494 D0000000198 D0000000334 D0000000292 D0000000196 D0000000293
ts_3[2].docs	D0000000498 D0000000340
ts_3[3].docs	D0000000386 D0000000316 D0000000464 D0000000388 D0000000504 D0000000291 D0000000296 D0000000442 D0000000339 D0000000347 D0000000354 D0000000355 D0000000385 D0000000446 D0000000374
ts_3[4].docs	D0000000421 D0000000468
ts_3[5].docs	D0000000508 D0000000323 D0000000402 D0000000412 D0000000407 D0000000400 D0000000492 D0000000404 D0000000413 D0000000401 D0000000467

5.2. Example 2

A virtual task 2 named “Context-aware knowledge service and application” is adopted as the illustrative example to explain the proposed idea. Two task instances belonging to this virtual task are listed in Table 6.

Table 6. Information of the virtual task 2

Virtual task 12: Context-aware knowledge service and application **$T_2=\{t_{43},t_{48}\}$**		
t_{43}	2003/09/16 ~ 2004/06/21	**Applying Topic Maps and Data Mining to Deploy Composite E-service Platform**
t_{48}	2003/09/23 ~ 2004/04/30	**Implementation of Personalized Recommendations for Composite E-service**

Step1. Identifying Task-stages. Three and four stages are identified from the task document sequence of task 48 and task 43, respectively. The clustering process of task 48 is illustrated in Figure 8.

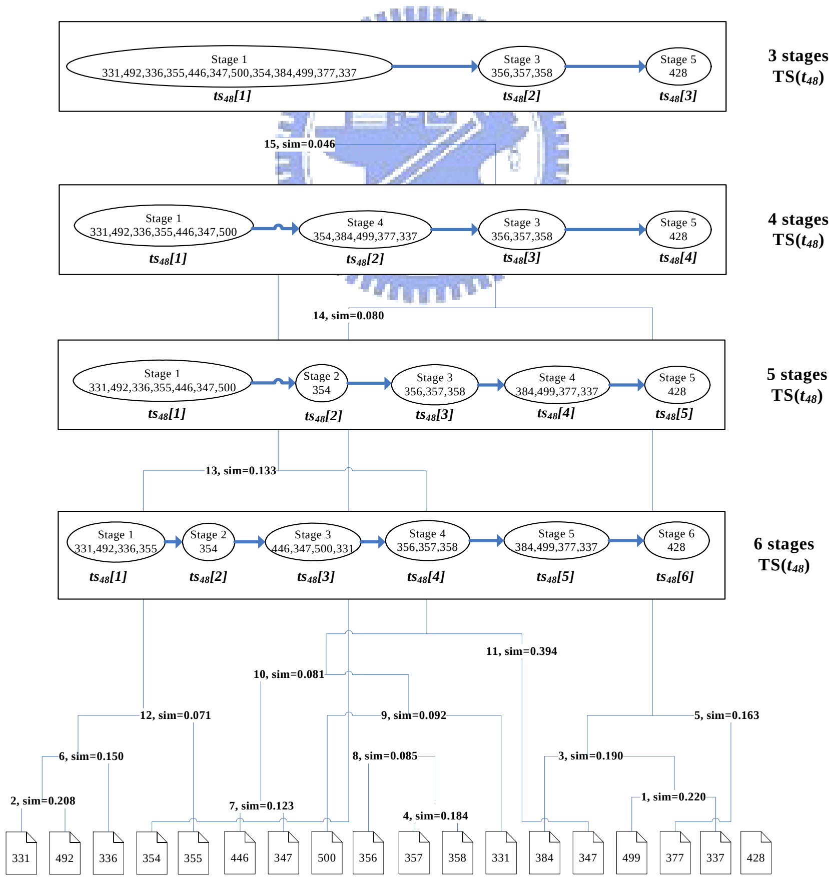


Figure 8: Task stage clustering process (Step 2)

- **Cluster examination:** We adopted the proposed *TSHC* algorithm listed in Table 1 to conduct the clustering.

Step2. Merging Task-stages. Task 43 is selected as the kernel task, since it has the best *Q* value. The process of merging the stages of task 48 to the stages of task 43 is shown in Figure 9.

Table 7. Clustering examination

Quality value of stages	Task 43	Task 48
Q(C) of 6 stages	0.125	0.088
Q(C) of 5 stages	0.139	0.084
Q(C) of 4 stages	**0.074**	0.082
Q(C) of 3 stages	0.110	**0.076**

Case 3: The clusters number of kernel task is greater than the clusters number of the other tasks
Task 48 is merged to task 43

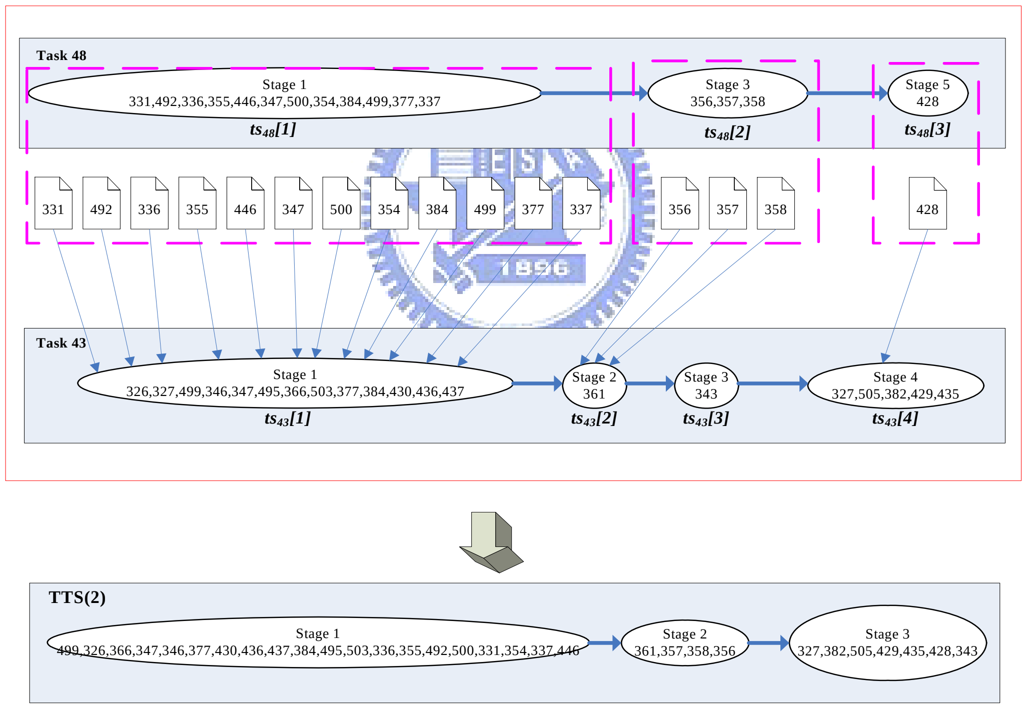


Figure 9. Task stage merge process (Step 2)

Step3. Reassigning documents: The reassigned result is shown in Table 8. In this case, one stage is removed. Stage 3 contains only one document D0000000343. After calculating the similarity between neighborhoods, the document D0000000343 is merged to the stage 4. Then the stage 3 is removed.

Table 8. Final document set at each task stage of target task 1 (Step 3)

TS(t_2)	
ts$_2$[1].docs	D0000000499 D0000000326 D0000000366 D0000000347 D0000000346 D0000000377 D0000000430 D0000000436 D0000000D437 D0000000384 D0000000495 D0000000503 D0000000336 D0000000355 D0000000492 D0000000500 D0000000331 D0000000354 D0000000337 D0000000446
ts$_2$[2].docs	D0000000361 D0000000357 D0000000358 D0000000356
ts$_2$[3].docs	D0000000327 D0000000382 D0000000505 D0000000429 D0000000435 D0000000428 D0000000343

Step 4. Extracting Task-stages profiles: Finally, the feature set of each task stage is extracted from each cluster.

6. Experiments

6.1. Experimental Setup

Three experiments were performed to evaluate whether the proposed task-stage knowledge support approach based on the *TSHC* algorithm can achieve better performance than our previous task-based knowledge support approach (Liu et al., 2005). Section 6.1.1 reviews the objective of the experiments and the experimental design. Meanwhile, Section 6.1.2 and Section 6.1.3 describe the data, participants, and evaluation metrics, respectively.

6.1.1. Overview of the experimental objective and design

Three experiments are conducted to evaluate the effectiveness of the proposed task-stage mining method for supporting task-relevant knowledge. The objectives of experimental evaluations were threefold: (1) Experiment one evaluates the impact of our proposed method with different time window size; (2) the experiment two evaluates the effectiveness of knowledge support based on task-stage compared with the baseline method, non-stage knowledge support (task-based knowledge support). In this work, we refer our previous task-based knowledge support as non-stage knowledge support; and (3) experiment three is similar to experiment two except we use different answering set for evaluating the effectiveness of the proposed method under different condition. Note that the knowledge support based on task-stage means that a virtual task with task-stage profiles can be used to deliver task-relevant knowledge for an executing task belonging to the task class of the virtual task. Our baseline method, task-based knowledge support uses the task profile of the virtual task to deliver task-relevant knowledge for an executing task belonging to the task class of the virtual task.

Two kinds of answering set of needed documents are prepared. The first is the task-answering set of a virtual task. The second is the stage-answering set for each stage of a virtual task. Note that the task-answering set is the union of the stage-answering sets. Section 6.1.2 explains the procedure to acquire the answering set in detail. Accordingly, experiment two compares task-stage knowledge support with non-stage knowledge support based on the task-answering set (named overall match). Experiment three compares stage knowledge support with non-stage knowledge support based on the stage-answering set (named stage match).

6.1.2. Data and participants

Experiments were conducted using a real application domain on conducting research tasks in a laboratory of a research institute. The tasks concerned are writing research papers or conducting research projects. The real application domain restricts the sample size of the data and participants in the experiments.

The test document set used in our experiment is collected from the laboratory in the institute of information management department. Over 600 task-related documents were collected. The smallest meaningful components of document information elements, such as title, abstract, journal and author, were extracted from documents. Each document contained an average of ninety distinct terms after information extraction, and document pre-processing (e.g. case folding, stemming, and stop word removal).

Three virtual tasks are selected from our research domain to evaluate the effectiveness of the proposed method, including "Change mining and application", "Context-aware knowledge service and application", and "Task-based knowledge support model and system". Each virtual task has its own needed kernel documents. The task-answering set of a virtual task is derived from the knowledge workers who have executed similar tasks and domain experts who have the knowledge of the virtual task. Furthermore, we asked the domain expert to select the needed documents at each stage of a virtual task. Consequently, the stage-answering set can be acquired. The limitation of this work is that it is not easy to identify the stage-answering sets of each virtual task. Therefore, only three stages of answering set of each virtual task are acquired from domain experts.

6.1.3. Evaluation metrics

The effectiveness of knowledge support is measured in terms of precision, recall and F1-measure, as in the research area of information retrieval.

Precision and recall. Precision is the fraction of retrieved items (tasks or documents) that are relevant, while recall is the fraction of total known relevant items that are retrieved, defined as follows.

$$precision = \frac{|retrieved\ items\ that\ are\ relevant|}{|total\ retrieved\ items|} \qquad \textbf{(3)}$$

$$recall = \frac{|relevant\ items\ that\ are\ retrieved|}{|total\ known\ relevant\ items|} \qquad \textbf{(4)}$$

F-Measure. The F-metric (van Rijsbergen, 1979; Riloff & Lehnert, 1994) could be used to balance the trade-off between precision and recall. F1 metric assigned equal weight to precision and recall and was given by,

$$F - measure(\beta) = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall} \quad \textbf{(5)}$$

The value of β is used to adjust the relative importance of the recall in comparison to the precision. In this experiment, we set β=1 such that the recall and precision have the same importance.

6.2. Experimental Result

6.2.1. Experiment one: effect on task-stage knowledge support with different size of time window

(1) Objective

The objective of this experiment is to evaluate the effectiveness of our proposed method with different time window size. As we have mentioned in Section 4.2, a time window size *w* is set to define the scope of candidate clusters to be merged. The smaller the time window size is, the higher the timeliness is. When time window size is equal to *n* (the number of documents in *TDS* of a task instance), the proposed *TSHC* algorithm is the same as standard hierarchical clustering algorithm (HAC). Thus, we conduct experiment to evaluate the impact of the time window parameter. And then the best time window size *w* will be selected for the usage of the proposed stage knowledge support method.

(2) Experimental Result and Observations

Table 9 shows the experimental results with different time window size in terms of precision, recall, and F1-measure. Four window sizes are evaluated: *w*=1, *w*=2, *w*=3 and *w*=*n*. Three virtual tasks are evaluated. In addition, we also evaluated the effectiveness of the proposed method under various top-*N* (*N*=10, 20, and 30) knowledge support, i.e., providing top-N documents with N-highest ranked similarity measures.

Observation 1: Generally, the best performance can be achieved when time window equals to one (*w*=1). Bold type denotes the best performance with different time window size. Specifically, when conducting top-30 knowledge support, time window equals to one (*w*=1) has the best performance. In addition, it is also apparently that the greater the time window, the lower effectiveness of the proposed method.

Observation 2: Interestingly, the virtual task 3 has the same performance no matter w=1 or w=2. It implies that the time effect is not obvious in this virtual task. On the other hand, w=1 not always leading to best performance for virtual task 1. It indicates that higher variant of TDS (task document sequence) exists for task instances of virtual task 1.

Table 9. Result of knowledge support with different time window for three virtual tasks

Parameters		**Virtual Task 1**			**Virtual Task 2**			**Virtual Task 3**		
Top-N	**w**	**Pre.**	**Re.**	**F1-measure**	**Pre.**	**Re.**	**F1-measure**	**Pre.**	**Re.**	**F1-measure**
Top-30	***w*=*1***	**0.267**	**0.203**	**0.231**	**0.231**	0.084	**0.171**	**0.344**	**0.175**	**0.232**
	***w*=2**	0.255	0.161	0.197	0.166	0.065	0.127	**0.344**	**0.175**	**0.232**
	***w*=3**	None	None	None	0.156	0.063	0.121	0.244	0.124	0.165
	***w*=n**	0.167	0.116	0.137	0.133	0.019	0.033	None	None	None
Top-20	***w*=*1***	0.267	0.126	0.171	**0.217**	0.069	**0.152**	**0.350**	**0.118**	**0.177**
	***w*=2**	**0.350**	**0.152**	**0.212**	0.156	0.049	0.118	**0.350**	**0.118**	**0.177**
	***w*=3**	None	None	None	0.133	0.037	0.088	0.250	0.085	0.126
	***w*=n**	0.167	0.079	0.107	0.150	0.039	0.062	None	None	None
Top-10	***w*=1**	0.333	0.080	0.129	**0.300**	0.047	**0.144**	**0.433**	0.074	**0.126**
	***w*=2**	**0.367**	**0.080**	**0.132**	0.200	0.026	0.085	**0.433**	**0.074**	**0.126**
	***w*=3**	None	None	None	0.167	0.023	0.074	0.333	0.056	0.096
	***w*=n**	0.200	0.050	0.080	0.156	0.059	0.086	None	None	None

Note: "None" denotes the proposed method is unable to generate 3 stages (clusters)

Table 10 shows the knowledge support results with different time window size of different task stages in terms of precision, recall, and F1-measure. As we have mentioned in Section 5.1.2, due to the limitation of stage answering set, only three stages of each virtual task are identified. According to previous pilot study (Vakkari 2000, 2003*)*, stage one indicates the task pre-focus stage; stage two indicates the task focus formulation stage; and stage three indicates the task post-focus stage. Note that the measure of each stage is the average of three virtual tasks.

Observation 3: Generally, the best performance can be achieved for all virtual task at stage 2 and stage 3 while conducting Top-20 and Top-10 knowledge support, when the time window size is set to one (*w*=1). Bold type denotes the best performance with different time window size. However, for the same top-*N* knowledge support, the best performance of different task stage is achieved in different time window size.

Observation 4: Generally, the greater the time window size, the lower effectiveness of the proposed method, especially for stage 2 and stage 3. However, stage 1 seems abnormal compared with stage 2 and stage 3. That is the greater the time window size, the higher effectiveness of the proposed method, especially in providing more documents.

Table 10. Result of knowledge support under different time window size for task stages

Parameters		Stage1			Stage 2			Stage3		
Top-N	*w*	Pre.	Re.	F-measure	Pre.	Re.	F-measure	Pre.	Re.	F-measure
Top-30	*w=1*	0.333	0.183	0.119	0.222	0.131	0.163	0.233	0.147	0.176
	w=2	0.366	0.195	0.251	**0.233**	**0.098**	**0.137**	0.167	0.107	0.126
	w=3	0.234	0.095	0.133	0.117	0.055	0.075	**0.250**	**0.132**	**0.172**
	w=n	**0.367**	**0.253**	**0.200**	0.084	0.034	0.067	0.033	0.029	0.018
Top-20	*w*=1	0.333	0.122	0.177	**0.283**	**0.112**	**0.159**	**0.217**	**0.080**	**0.116**
	w=2	**0.433**	**0.249**	**0.179**	0.283	0.080	0.125	0.167	0.060	0.088
	w=3	0.225	0.064	0.098	0.125	0.040	0.060	0.117	0.055	0.075
	w=n	0.350	0.184	0.128	0.075	0.021	0.046	0.050	0.029	0.022
Top-10	*w=1*	0.400	0.070	0.119	**0.400**	**0.080**	**0.133**	**0.267**	**0.049**	**0.083**
	w=2	**0.433**	**0.089**	**0.147**	0.367	0.054	0.094	0.200	0.036	0.061
	w=3	0.350	0.053	0.091	0.250	0.040	0.068	0.150	0.027	0.029
	w=n	0.350	0.111	0.067	0.050	0.007	0.020	0.100	0.029	0.028

(3) Implications:

This experiment evaluates the impact of the time window size in the proposed *TSHC* algorithm. The result reveals that generally the smaller the time window size, the better the effectiveness (the higher precision, recall, and F-measure) of knowledge support. Accordingly, we set time window size to one (*w*=1) in the proposed algorithm.

Interestingly, when time window size is equal to *n* (number of document *TDS* of a task instance), the proposed *TSHC* algorithm is the same as standard hierarchical clustering algorithm (HAC). The time effect disappeared when time window size is equal to *n*. The result is similar to topics clustering, i.e., documents within the same cluster may discuss a similar topic without considering the effect of stage.

6.2.2 Experiment two: comparing task-stage knowledge support with non-stage knowledge support based on overall match

(1) Objective

The objective of this experiment is to evaluate the effectiveness of task-stage mining method for knowledge support. Task-stage profiles are generated to model workers' task-stage needs and are used for delivering task-relevant knowledge at various task stages. The *task stage knowledge support method* is compared with the baseline method, *non-stage knowledge support method* (task-based knowledge support). The *task-stage knowledge support* denotes providing needed documents based on task-stage profiles, whereas *non-stage knowledge support* denotes providing knowledge support based on task profiles. The task-stage profiles or task profiles are used to deliver task-relevant knowledge for an executing task that is similar to the virtual task. Note that this experiment compares *task-stage knowledge support* with *non-stage knowledge support* according to the task-answering set of a virtual task (named overall match). On the other hand, the experiment three compares *task-stage knowledge support* with *non-stage knowledge support* according to the stage answering set of a virtual task (named stage match).

(2) Result and Observations

Table 11 shows the result of *task-stage knowledge support method* and *non-stage knowledge support method* in terms of precision, recall, and F1-measure. Notably, three virtual tasks are evaluated under various top-*N* (*N*=10, 20, and 30) document supports. Table 12 shows the average result of *task-stage knowledge support method* and *non-stage knowledge support method*.

Observation 1: The precision, recall and F1-measure of *task stage knowledge support method* are greater than the *non-stage method* for virtual task one and two. However, for virtual task three, the precision, recall and F1-measure of *task stage knowledge support method* are lesser than *non-stage method.* The precision of *non-stage method* under Top-30 support is greater than that of the *task stage knowledge support method*.

Observation 2: Notably, for *task stage knowledge support method*, the fewer number of supporting documents, the higher precision value is. The rank of precision value is Top-10 > Top-20 > Top-30. The observation doest not apply to the *non-stage knowledge support method*. The precision values of *task stage knowledge support method* are greater than those of the *non-stage method* under top-10 knowledge support for three virtual tasks. The result reveals that *task stage knowledge support method* can provide more effective knowledge support than the *non-stage knowledge support method* when providing fewer number of task-stage relevant documents.

Table 11. Compare task stage knowledge support method with none-stage knowledge support method for three virtual tasks

		Task Stage Knowledge Support			Non-Stage (Task-based) Knowledge Support		
		Pre.	**Re.**	**F-measure**	**Pre.**	**Re.**	**F-measure**
Virtual Task 1	**Top-10**	**0.667**	**0.082**	**0.146**	0.600	0.078	0.138
	Top-20	**0.633**	**0.165**	**0.261**	0.600	0.156	0.248
	Top-30	**0.578**	**0.225**	**0.324**	0.567	0.220	0.317
	Average	**0.626**	**0.157**	**0.244**	0.589	0.151	0.234
Virtual Task 2	**Top-10**	**0.433**	**0.03**	**0.06**	0.300	0.021	0.039
	Top-20	0.350	0.05	0.09	**0.400**	**0.056**	**0.098**
	Top-30	0.322	0.07	0.11	**0.400**	**0.085**	**0.140**
	Average	**0.369**	**0.050**	**0.086**	0.367	0.054	0.093
Virtual Task 3	**Top-10**	**0.567**	**0.060**	**0.109**	0.500	0.053	0.096
	Top-20	**0.467**	**0.099**	**0.163**	0.450	0.096	0.158
	Top-30	0.444	0.142	0.215	**0.567**	**0.181**	**0.274**
	Average	0.493	0.100	0.162	**0.506**	**0.110**	**0.176**

Observation 3: Table 11 shows the details of the experiment result, which shows that virtual task one has best result. This may result from that the best clustering number for virtual task one is 3, but clustering number for virtual task two and three are not 3. As we have addressed in step 1 of Section 4.2, we use the *Q* value to determine the cluster quality for deciding the number of clusters of each task.

Observation 4: Table 12 shows the result of *task stage knowledge support method* and *non-stage knowledge support method*. The average precision, recall and F1-measure of *task stage knowledge support method* are greater than those of the *non-stage method* under top-10 and top-20 knowledge support. Meanwhile, for *task stage knowledge support method*, the fewer number of supporting documents, the higher precision value is. The rank of precision value is Top-10 > Top-20 > Top-30. For *non- stage knowledge support method*, the fewer number of supporting documents, the lower precision value is. The rank of precision value is Top-30 > Top-20 > Top-10.

Table 12. Results of knowledge support by overall match

	Stage knowledge support			**Non- stage knowledge support**		
β=1	Precision	Recall	F1-measure	Precision	Recall	F1-measure
Top-10	**0.556**	**0.058**	**0.104**	0.467	0.051	0.091
Top-20	**0.483**	**0.104**	**0.171**	0.483	0.103	0.168
Top-30	0.448	0.145	0.217	**0.511**	**0.162**	**0.244**
Average	**0.496**	**0.102**	**0.164**	0.487	0.105	0.168

Table 13, and Figure 10 (A), (B), and (C) shows the results of *task stage knowledge support method* with *none-stage knowledge support method* of different task stages. Note that the *none-stage knowledge support method* of different task stages has the same performance value. The reason is that the *non-stage knowledge support method* only has one task profile. In addition, this experiment evaluates the performance of two methods according to the task-answering set (named overall match) not the stage answering sets.

Observation 5: Generally, *task stage knowledge support method* has better performance than *non-stage method* under top-10 knowledge support for three task stages. In addition, no matter top-10, 20 or 30 knowledge supports, *task stage knowledge support method* has better performance than *non-stage method* in task pre-focus stage (stage 1).

Table 13: Compare task stage knowledge support method with none-stage knowledge support method under different task stages

		Task Stage-based Knowledge Support			**Non-Stage (Task-based) Knowledge Support**		
		Pre.	**Re.**	**F-measure**	**Pre.**	**Re.**	**F-measure**
Stage 1 (Pre-focus)	**Top-10**	**0.567**	**0.064**	**0.218**	0.467	0.051	0.177
	Top-20	**0.517**	**0.120**	**0.309**	0.483	0.103	0.278
	Top-30	**0.522**	**0.181**	**0.376**	0.511	0.162	0.357
	Average	**0.535**	**0.122**	**0.301**	0.487	0.105	0.270
Stage 2 (Focus)	**Top-10**	**0.500**	**0.051**	**0.179**	0.467	0.051	0.177
	Top-20	0.417	0.091	0.240	**0.483**	**0.103**	**0.278**
	Top-30	0.311	0.101	0.217	**0.511**	**0.162**	**0.357**
	Average	0.409	0.081	0.212	**0.487**	**0.105**	**0.270**
Stage 3 (Post-focus)	**Top-10**	**0.600**	**0.059**	**0.209**	0.467	0.051	0.177
	Top-20	0.517	0.102	0.281	0.483	0.103	0.278
	Top-30	0.511	0.153	0.343	**0.511**	**0.162**	**0.357**
	Average	**0.543**	**0.105**	**0.277**	0.487	0.105	0.270

Note: The details of stage knowledge support of each group are listed in Appendix B1

Observation 6: Figure 10(A), (B), and (C) depict the result of knowledge support based on various number of top-N knowledge support. Interestingly, stage 1 and stage 3 of task-stage knowledge support method has better performance than non-stage knowledge support. However, stage 2 of task-stage knowledge support method is not always better than non-stage knowledge support method under various top-N knowledge supports.

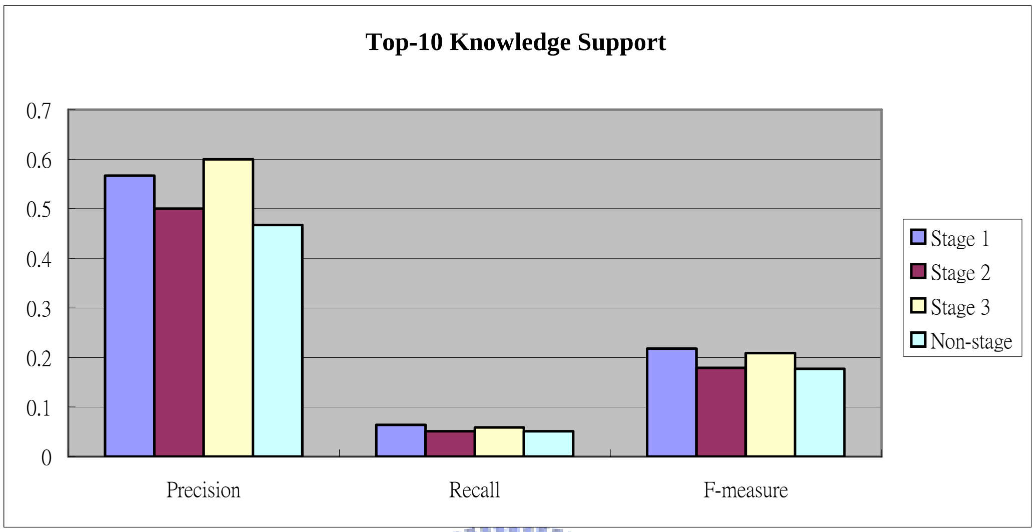


Figure 10(A): Result of top-10 knowledge support

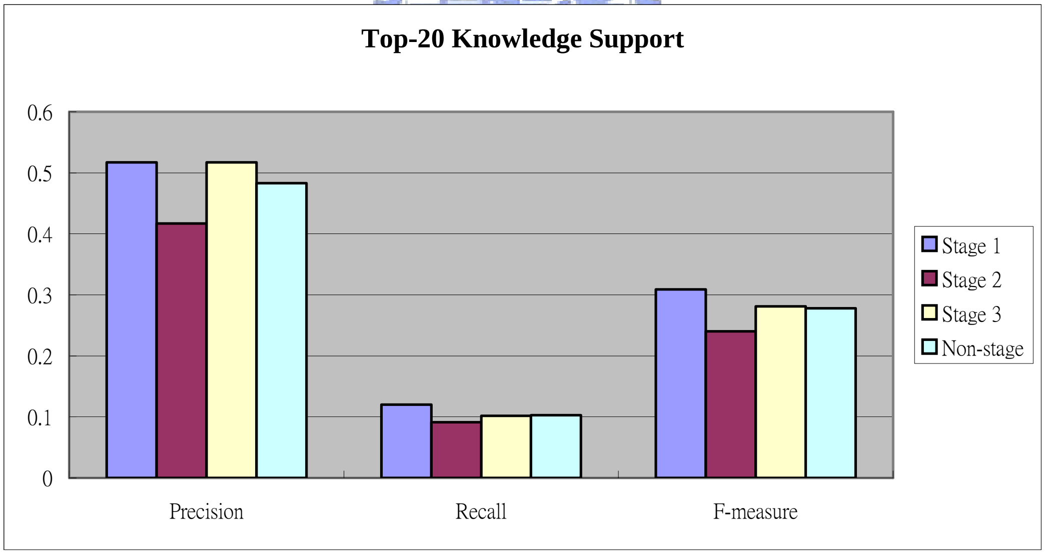


Figure 10(B): Result of top-20 knowledge support

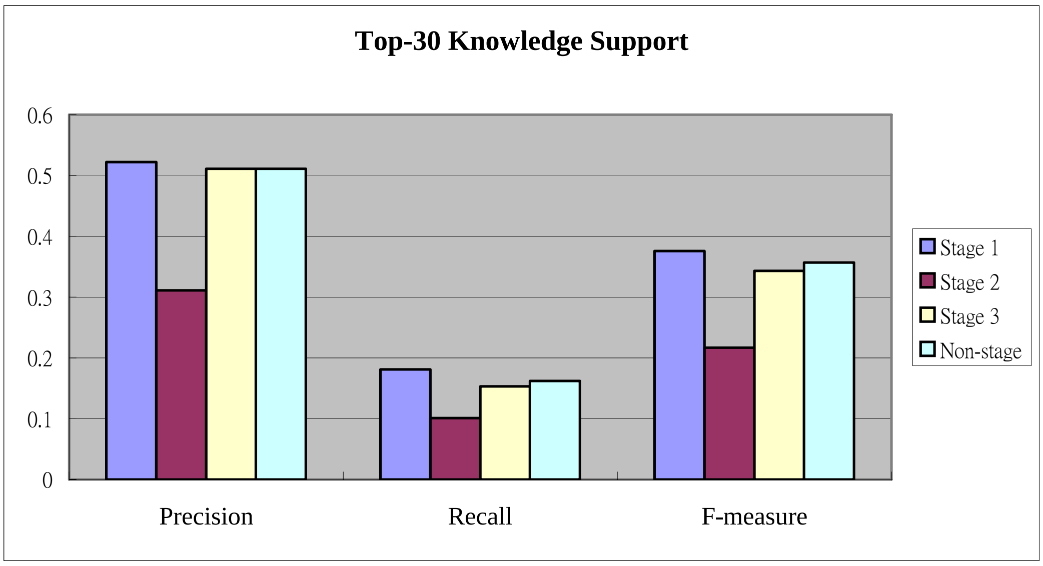


Figure 10(C): Result of top-30 knowledge support

(3) Implications:

This experiment evaluates the effectiveness of the proposed task-stage mining method for knowledge support. The experimental result reveals that generally the *task stage knowledge support method* is better than the baseline method, *non-stage knowledge support method* (task-based knowledge support). Notably, for *task stage knowledge support method*, the fewer number of supporting documents, the higher precision value is (Top-10 > Top-20 > Top-30). The result revels that the proposed stage mining method can provide more effective knowledge support. Meanwhile, knowledge workers generally do not need a lot of documents at each task stage; therefore, the system needs to determine appropriate number of needed documents to support task-execution. The experimental result is useful to suggest the number of supporting documents at each task stage.

6.2.3. Experiment three: comparing task-stage knowledge support with non-stage knowledge support method based on stage match

(1) Objective

The objective of this experiment is to evaluate the effectiveness of task-stage mining method for knowledge support. Experiment three compares *task stage knowledge support* with *non-stage knowledge support* according to the stage answering set (named stage match). This experiment is similar to experiment two except that we use different answering set (stage answering set) for evaluating the effectiveness of the proposed method.

(2) Result and Observations

Table 14 shows the result of *task stage knowledge support method* and *non-stage knowledge support method* in terms of precision, recall, and F1-measure. Notably, three virtual tasks are evaluated under various top-*N* (*N*=10, 20, and 30) document supports. Table 15 shows the average result of *task stage knowledge support method* and *non-stage knowledge support method*, respectively.

Observation 1: The average precision, recall and F1-measure of *task stage knowledge support method* are greater than *non-stage method* for virtual task two and three. However, for virtual task one, the average precision, recall and F1-measure of *task stage knowledge support method* are lesser than the *non-stage method*. Only top-10 document support of *task stage knowledge support method* is better than the *non-stage method.*

Observation 2: Notably, for *task stage knowledge support method*, the fewer number of supporting documents, the higher precision value is. The rank of precision value is Top-10 > Top-20 > Top-30. On the other hand, this observation does not apply to the *non-stage knowledge support method*. The precision values of *task stage knowledge support method* are greater than those of the *non-stage method* under top-10 knowledge support for three virtual tasks. The result reveals that *task stage knowledge support method* can provide more effective knowledge support than the *non-stage knowledge support method* when supporting fewer number of task-stage relevant documents. This result is in accordance with the result of experiment two.

Observation 3: Table 15 shows the average result of *task stage knowledge support method* and *non-stage knowledge support method*. The average precision, recall and F1-measure of *task stage knowledge support method* are all greater than those of the *non-stage method*. Notably, if we compare this result with experiment 2 (Table 12), the performance value is not better than experiment 2. This result is reasonably because the number of documents of each stage answering set is far fewer than the number of documents of the task answering set. This experiment uses the stage-answering sets, whereas the experiment two uses only one task answering set. The task answering set of a virtual task is the union of stage answering sets of a virtual task.

Table 14. Compare task-stage knowledge support method with none-stage knowledge support method for three virtual tasks

		Task Stage Knowledge Support			**Non-Stage (Task-based) Knowledge Support**		
		Pre.	**Re.**	**F-measure**	**Pre.**	**Re.**	**F-measure**
Virtual Task 1	**Top-10**	**0.333**	**0.080**	**0.128**	0.300	0.074	0.128
	Top-20	0.267	0.126	0.171	**0.317**	**0.162**	**0.171**
	Top-30	0.267	0.203	0.229	**0.289**	**0.229**	**0.229**
	Average	0.289	0.136	0.176	**0.302**	**0.155**	**0.176**
Virtual Task 2	**Top-10**	**0.300**	**0.047**	**0.080**	0.233	0.033	0.133
	Top-20	**0.217**	**0.069**	**0.104**	0.167	0.051	0.117
	Top-30	**0.231**	**0.084**	**0.113**	0.178	0.079	0.122
	Average	**0.249**	**0.067**	**0.099**	0.193	0.055	0.124
Virtual Task 3	**Top-10**	**0.433**	**0.074**	**0.135**	0.233	0.039	0.167
	Top-20	**0.350**	**0.118**	**0.194**	0.300	0.102	0.200
	Top-30	0.344	0.175	0.232	**0.367**	**0.253**	**0.233**
	Average	**0.376**	**0.122**	**0.187**	0.300	0.131	0.200

Table 15. Average results of knowledge support for three virtual tasks by stage match

	Stage knowledge support			**Non- stage knowledge support**		
β=1	Precision	Recall	F1-measure	Precision	Recall	F1-measure
Top-10	**0.356**	**0.067**	**0.115**	0.256	0.049	0.143
Top-20	**0.278**	**0.105**	**0.156**	0.261	0.105	0.163
Top-30	**0.281**	**0.154**	**0.191**	0.278	0.187	0.195
Average	**0.305**	**0.108**	**0.154**	**0.265**	**0.114**	**0.167**

Table 16 shows the results of *task stage knowledge support method* with *none-stage knowledge support method* under different task stages.

Observation 4: Generally, *task stage knowledge support method* has better performance than the *non-stage method* under top-10 or top-20 knowledge support for three task stages. In addition, no matter top-10, 20 or 30 knowledge supports, *task stage knowledge support method* has better performance than the *non-stage method* in task pre-focus stage (stage 1).

Table 16. Compare task stage knowledge support method with none-stage knowledge support

method under different task stages (Stage match)

		Task Stage-based Knowledge Support			**Non-Stage (Task-based) Knowledge Support**		
		Pre.	**Re.**	**F-measure**	**Pre.**	**Re.**	**F-measure**
Stage 1 (Pre-focus)	**Top-10**	**0.400**	**0.070**	**0.119**	0.367	0.066	0.111
	Top-20	**0.333**	**0.122**	**0.177**	**0.333**	**0.126**	**0.181**
	Top-30	**0.333**	**0.183**	**0.234**	**0.333**	**0.248**	**0.277**
	Average	**0.355**	**0.125**	**0.176**	0.344	0.147	0.190
Stage 2 (Focus)	**Top-10**	**0.400**	**0.080**	**0.133**	0.233	0.043	0.072
	Top-20	**0.283**	**0.112**	**0.159**	0.233	0.091	0.129
	Top-30	0.222	0.131	0.163	**0.267**	**0.153**	**0.192**
	Average	**0.302**	**0.108**	**0.151**	0.244	0.096	0.131
Stage 3 (Post-focus)	**Top-10**	**0.267**	**0.049**	**0.083**	0.167	0.033	0.057
	Top-20	**0.217**	**0.080**	**0.116**	0.217	0.102	0.152
	Top-30	0.233	0.147	0.176	**0.234**	**0.186**	**0.247**
	Average	**0.239**	**0.092**	**0.125**	0.206	0.107	0.152

Note: The details of stage knowledge support of each group are listed in Appendix B2-1 and B2-2

(3) Implications:

This experiment evaluates the effectiveness of the proposed task-stage mining method for knowledge support. The results prove that the effectiveness of *task stage knowledge support method* by the *TSHC* algorithm is better than the non-stage method. This experiment has more significant improvement in *precision* by *task stage knowledge support* than that of experiment 2. Table 17 shows the comparison of Table 12 and Table 15 in experiment 2 and 3, respectively. Notably, in average, the *recall* value of *non-stage knowledge support* is better than that of the *task stage knowledge support*. But if we took further analysis, the recall value of top-30 *task stage knowledge support* is much lower than that of *non-stage knowledge support*. This result implies that the system can deliver more task-relevant documents by providing top-30 documents based on task profiles and providing top-10 or top-20 documents based on task-stage profiles.

Table 17. Average results of knowledge support for three virtual tasks by stage match

	Stage knowledge support			**Non- stage knowledge support**		
β=1	Precision	Recall	F1-measure	Precision	Recall	F1-measure
Average results of knowledge support for three virtual tasks by stage match						
Top-10	**0.356**	**0.067**	0.115	0.256	0.049	0.143
Top-20	**0.278**	**0.105**	0.156	0.261	**0.105**	0.163
Top-30	**0.281**	0.154	0.191	0.278	**0.187**	0.195
Average	**0. 305 (+15.09%)**	**0. 108 (-5.26%)**	**0. 154**	**0.265**	**0.114**	**0.167**
Average results of knowledge support for three virtual tasks by overall match						
Top-10	**0.556**	**0.058**	0.104	0.467	0.051	0.091
Top-20	**0.483**	**0.104**	0.171	0.483	0.103	0.168
Top-30	0.448	0.145	0.217	**0.511**	**0.162**	0.244
Average	**0.496 (+1.85%)**	**0.102 (-2.94%)**	**1. 0.164**	0.487	0.105	0.168

7. Conclusions and Future Works

Codifying knowledge into explicit form is a widely adopted strategy in contemporary knowledge management systems (KMS) to facilitate knowledge reuse. However, workers still encounter difficulty in accessing information from vast amount of codified knowledge. This work proposed a task-stage mining technique to tackle the problem. The valuable knowledge of each task-stage is extracted to build the task-stage profile for delivering task-relevant knowledge at various stages. Several experiments have been conducted to demonstrate the effectiveness of the proposed task-stage knowledge support approach.

In our ongoing work, we have proposed *a task-stage identification model* to determine the changes of a worker's task-stage (Wu et al., 2005). Accordingly, we will integrate the proposed task-stage mining technique with the task-stage identification model to investigate the contribution of the task-stage knowledge support model empirically. Meanwhile, we are seeking other application domain to apply the proposed model.

References

[1] A. Abecker, A. Bernardi, H. Maus, M. Sintek, and C. Wenzel, (2000), "Information Supply for Business Processes: Coupling Workflow with Document Analysis and Information Retrieval," *Knowledge Based Systems*, 13(1), pp. 271-284.

[2] R. Baeza-Yates, and B. Ribeiro-Neto, (1999), *Modern Information Retrieval*, Reading, MA: Addison-Wesley.A.

[3] A. Celentano, M.G. Fugini, and S. Pozzi, (1995), "Knowledge-based Document Retrieval in Office Environment: The Kabiria System," *ACM Transactions on Information Systems*, *13*(3), pp.237-268.

[4] S.-L. Chuang, and L.-F. Chien (2004), "A practical web-based approach to generating topic hierarchy for text segments," *CIKM*, pp.127-136

[5] T. H. Davenport, and L. Prusak, (1998), *Working knowledge: How Organizations Manages What They Know*. Boston MA: Harvard Business School Press.

[6] Kurt D. Fenstermacher, (2002), "Process-Aware Knowledge Retrieval," *Proc. of the 35th Hawaii Intl. Conf. on System Sciences*, Hawaii, USA.

[7] G. Fischer, and J. Ostwald, (2001), "Knowledge Management: Problems, Promises, Realities, and Challenges," *IEEE Intelligent Systems*, 16(1), pp.60–73.

[8] J. Han, and M. Kamber, (2000) *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publisher, San Francisco.

[9] A.K. Jain，M.N，Murty and P.J Flynn, (1999), ”Data Clustering：A Review,” *ACM computing Surveys*, 131(3).

[10] S. C. Johnson (1967) "Hierarchical Clustering Schemes," *Psychometrika*, 2:241-254

[11] C. Kuhlthau, (1993), *Seeking Meaning: A Process Approach to Library and Information Services*, Norwood, NJ: Ablex Publishing Corp.

[12] Gartner Group (Summer 1999), *Knowledge Management Reports*.

[13] P.H. Gray (2001), “Problem-solving Perspective on Knowledge Management Practices,” *Decision Support Systems*, 31(1), pp.87-102.

[14] D.-R. Liu, I.-C. Wu, and K.-S. Yang (2005), “Task-based K-Support System: Disseminating and Sharing Task-relevant Knowledge,” *Expert Systems with Applications*, 29 (2), pp.408-423.

[15] J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations,” *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability"*, Berkeley, University of California Press, 1:281-297.

[16] J. Mostafa, S. Mukhopadhyay, W. Lam, and M. Palakal, (1997), "A Multi-level Approach to Intelligent Information Filtering: Model, System and Evaluation," ACM Transactions on Information Systems, 15(4), pp. 368-399.

[17] I. Nonaka, (1994), “A Dynamic Theory of Organizational Knowledge Creation,” Organization Science, 5(1), pp.14–37.

[18] M. Pazzani, D. Billsus, (1997), “Learning and Revising User Profiles: The Identification of Interesting Web Sites,” *Machine Learning*, 27, pp. 313–331.

[19] van Rijsbergen, C.J., (1979). *Information Retrieval,* Second ed. Butterworths, London.

[20] E. Riloff and W. Lehnert. (1994), “Information Extraction as a Basis for High Precision Text Classification,” *ACM Transaction on Information System*, 12(3), pp.296-333.

[21] G. Salton, and C. Buckley (1988), “Term Weighting Approaches in Automatic Text Retrieval,” *Information Processing & Management*, 24(5), pp. 513–523.

[22] F. Sebastiani, (1999), “Machine Learning in Automated Text Categorization: a Survey,” *Technical report*, Istituto di Elaborazione dell'Informazione, C.N.R., Pisa, Italy.

[23] S. Staab, and H.-P. Schnurr, (2000), “Smart Task Support through Proactive Access to Organizational Memory,” *Knowledge-Based Systems*, 13(5), pp. 251-260.

[24] B. Shapira, P. Shoval and U. Hanani, (1999), "Experimentation with an Information Filtering System that Combines Cognitive and Sociological Filtering Integrated with User Stereotypes," *Decision Support Systems*, 27, pp.5–24.

[25] P. Vakkari (2000), "Cognition and changes of search terms and tactics during task performance: A longitudinal case study," *Proceedings of the RIAO'2000 Conference*. Paris: C.I.D., pp.894-907.

[26] P. Vakkari (2003), "Changes of Search Terms and Tactics While Writing a Research Proposal: A Longitudinal Case Study," *Information Processing and Management*, 39(3), pp. 445-463.

[27] H. Widyantoro, T. R. Ioerger, and J. Yen, (2001), "Learning User Interest Dynamics with a Three-Descriptor Representation," *Journal of the American Society for Information Science*, 52(3), pp.212-225.

[28] Wiig, K (1993), *Knowledge Management Foundation*. Schema Press.

[29] I.-C. Wu, D.-R. Liu, and W.-H. Chen (2005), "Task-stage Knowledge Support Model: Coupling User Information Needs with Task Stage Identification," *Proc. of the IEEE International Conference on Information Reuse and Integration (IRI)*, Las Vegas, USA.

[30] C. Wei, P. Hu, and H. H. Chen (2002), "Design and Evaluation of A Knowledge Management System," *IEEE Software*, 19(3), pp.56-59.

[31] K.-S. Yang (2004), "Implementation of Task-based Knowledge Support System", NCTU, Master Thesis.

[32] M.H. Zack (1999). Managing codified knowledge. *Sloan Management Review*, 40(4), 45-58.

Appendix A1

(comparing stage knowledge support with non-stage knowledge support method of overall match)

Stage knowledge support results (Group 1)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.900	0.385	0.207	0.142	0.117	0.700	0.270	0.140	0.095	0.078	0.400	0.171	0.092	0.063	0.052
Top-20	0.900	0.574	0.371	0.275	0.234	0.600	0.382	0.248	0.183	0.156	0.400	0.255	0.165	0.122	0.104
Top-30	0.867	0.660	0.486	0.385	0.338	0.433	0.330	0.243	0.192	0.169	0.433	0.330	0.243	0.192	0.169
Average	**0.889**	**0.540**	**0.355**	**0.267**	**0.230**	**0.578**	**0.327**	**0.210**	**0.157**	**0.134**	**0.411**	**0.252**	**0.167**	**0.126**	**0.108**

Stage knowledge support results (Group 2)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.300	0.082	0.039	0.026	0.021	0.300	0.082	0.039	0.026	0.021	0.700	0.194	0.093	0.061	0.050
Top-20	0.150	0.067	0.037	0.025	0.021	0.300	0.137	0.075	0.052	0.043	0.600	0.271	0.149	0.103	0.085
Top-30	0.167	0.095	0.058	0.042	0.035	0.233	0.135	0.082	0.059	0.050	0.567	0.325	0.198	0.142	0.120
Average	**0.206**	**0.082**	**0.045**	**0.031**	**0.026**	**0.278**	**0.118**	**0.066**	**0.046**	**0.038**	**0.622**	**0.264**	**0.147**	**0.102**	**0.085**

Stage knowledge support results (Group 3)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.500	0.186	0.096	0.065	0.053	0.500	0.186	0.096	0.065	0.053	0.700	0.260	0.134	0.090	0.074
Top-20	0.500	0.287	0.175	0.126	0.106	0.350	0.200	0.122	0.088	0.074	0.550	0.316	0.193	0.139	0.117
Top-30	0.533	0.373	0.258	0.197	0.170	0.267	0.187	0.129	0.098	0.085	0.533	0.373	0.258	0.197	0.170
Average	**0.511**	**0.282**	**0.176**	**0.129**	**0.110**	**0.372**	**0.191**	**0.116**	**0.084**	**0.071**	**0.594**	**0.317**	**0.195**	**0.142**	**0.120**

Appendix A2-1

(comparing stage knowledge support with non-stage knowledge support method of stage match)

Stage knowledge support results (Group 1)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.500	0.294	0.182	0.131	0.111	0.500	0.316	0.204	0.150	0.128	0.000	0.000	0.000	0.000	0.000
Top-20	0.450	0.360	0.277	0.225	0.200	0.350	0.294	0.237	0.198	0.179	0.000	0.000	0.000	0.000	0.000
Top-30	0.433	0.394	0.347	0.310	0.289	0.267	0.252	0.232	0.215	0.205	0.100	0.103	0.107	0.112	0.115
Average	**0.461**	**0.349**	**0.268**	**0.222**	**0.200**	**0.372**	**0.287**	**0.224**	**0.188**	**0.171**	**0.033**	**0.034**	**0.036**	**0.037**	**0.038**

Stage knowledge support results (Group 2)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.300	0.115	0.059	0.040	0.033	0.200	0.088	0.048	0.033	0.027	0.400	0.222	0.133	0.095	0.080
Top-20	0.150	0.088	0.054	0.039	0.033	0.200	0.131	0.086	0.064	0.055	0.300	0.231	0.171	0.136	0.120
Top-30	0.133	0.095	0.066	0.051	0.044	0.167	0.129	0.097	0.077	0.068	0.233	0.206	0.175	0.152	0.140
Average	**0.194**	**0.099**	**0.060**	**0.043**	**0.037**	**0.189**	**0.116**	**0.077**	**0.058**	**0.050**	**0.311**	**0.220**	**0.160**	**0.128**	**0.113**

Stage knowledge support results (Group 3)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.400	0.201	0.114	0.080	0.067	0.500	0.255	0.147	0.103	0.086	0.400	0.202	0.145	0.082	0.068
Top-20	0.400	0.285	0.200	0.153	0.133	0.300	0.217	0.154	0.119	0.103	0.350	0.252	0.228	0.137	0.119
Top-30	0.433	0.361	0.289	0.241	0.217	0.233	0.197	0.159	0.134	0.121	0.367	0.307	0.247	0.206	0.186
Average	**0.411**	**0.282**	**0.201**	**0.158**	**0.139**	**0.344**	**0.223**	**0.153**	**0.119**	**0.103**	**0.372**	**0.254**	**0.207**	**0.142**	**0.124**

Appendix A2-2

(comparing stage knowledge support with non-stage knowledge support method of stage match)

Non-Stage knowledge support results (Group 1)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.600	0.352	0.218	0.158	0.133	0.200	0.126	0.081	0.060	0.051	0.100	0.075	0.055	0.043	0.038
Top-20	0.550	0.440	0.338	0.275	0.244	0.250	0.210	0.169	0.142	0.128	0.150	0.141	0.130	0.121	0.115
Top-30	0.433	0.394	0.347	0.310	0.289	0.267	0.252	0.232	0.215	0.205	0.167	0.171	0.179	0.186	0.192
Average	**0.528**	**0.395**	**0.301**	**0.247**	**0.222**	**0.239**	**0.196**	**0.161**	**0.139**	**0.128**	**0.139**	**0.129**	**0.121**	**0.117**	**0.115**

Non-Stage knowledge support results (Group 2)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.300	0.115	0.059	0.040	0.033	0.200	0.088	0.048	0.033	0.027	0.200	0.111	0.067	0.048	0.040
Top-20	0.150	0.088	0.054	0.039	0.033	0.150	0.098	0.064	0.048	0.041	0.200	0.154	0.114	0.091	0.080
Top-30	0.167	0.120	0.084	0.065	0.056	0.200	0.155	0.116	0.093	0.082	0.167	0.147	0.125	0.109	0.100
Average	**0.206**	**0.107**	**0.066**	**0.048**	**0.041**	**0.183**	**0.114**	**0.076**	**0.058**	**0.050**	**0.189**	**0.137**	**0.102**	**0.082**	**0.073**

Non-Stage knowledge support results (Group 3)

	Stage 1					**Stage 2**					**Stage 3**				
	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.	Pre.	β=0.5	β=1	β=2	Re.
Top-10	0.200	0.099	0.057	0.040	0.033	0.300	0.154	0.088	0.062	0.052	0.200	0.099	0.058	0.040	0.033
Top-20	0.300	0.214	0.150	0.115	0.100	0.300	0.217	0.154	0.119	0.103	0.300	0.216	0.152	0.118	0.102
Top-30	0.400	0.400	0.400	0.400	0.400	0.333	0.280	0.227	0.190	0.172	0.367	0.307	0.247	0.206	0.186
Average	**0.300**	**0.238**	**0.202**	**0.185**	**0.178**	**0.311**	**0.217**	**0.156**	**0.124**	**0.109**	**0.289**	**0.208**	**0.152**	**0.121**	**0.107**