Computing and Informatics, Vol. 33, 2014, 281–302

# COMPACT INDEXES BASED ON CORE CONTENT IN PERSONAL DATASPACE MANAGEMENT SYSTEM

Ning Wang, Hongfang Du, Baomin Xu

*School of Computer and Information Technology*
*Beijing Jiaotong University*
*No. 3 Shangyuancun, Haidian District, 100044 Beijing, China*
*e-mail:* {nwang, 09120533, bmxu}@bjtu.edu.cn

Guojun Dai

*Computer School, Hangzhou Dianzi University*
*Hangzhou, 310018, China*
*e-mail:* daigj@hdu.edu.cn

**Abstract.** A Personal DataSpace Management System is a platform to manage personal data with heterogeneous data types, in which keyword query is a primary query form for users who know little about the structure of the dataspace. Unlike exploratory queries in web search, a user in a personal dataspace usually has a specific search target and wants to find some known items in mind. To improve result quality in terms of query relevance in a personal dataspace, we propose the concept of compact index in this paper. We refer to the most important and representative semantics from documents as core content, and build compact index on it. We propose algorithm for selecting core content from a document based on semantic analysis, which can process English and Chinese documents uniformly. Furthermore, a software platform named Versatile is introduced for flexible personal data management, in which core content is extracted for building compact indexes and generating query-biased snippet efficiently and accurately. Finally, extensive experiments have been conducted to show the effectiveness and feasibility of compact indexes in personal dataspace management system.

**Keywords:** Keyword query, indexing, result quality, semantic analysis, personal dataspace management system

## 1 INTRODUCTION

Nowadays, with explosion of the amount of digital information, individual computer users have developed their own vast data on their desktops and varied electronic devices such as laptops, PDAs, cellphones. A Personal DataSpace Management System (PDSMS) is a platform to manage personal data with heterogeneous data types [1]. Unlike data integration systems with a unified integration schema, dataspaces do not have a single schema beforehand upon which users can pose queries. Initially, keyword queries can be used for users with little knowledge about structure information. As users know more about the dataspace, they should be able to pose more sophisticated queries.

For keyword search, search engines usually build indexes based on full text in order to return a comprehensive list of results,which may require much time and space for large datasets. The most crucial problem is to achieve result relevance. Sometimes users could not easily find relevant results from a long list of results spanning many pages, and they are reluctant to view the results listed beyond the first page [2]. In order to let users easily find relevant information, various rank algorithms [3, 4] have then been studied so that documents which are considered to match users' intention can be listed as early in the list as possible. Also, query-biased snippets are presented by web search engines to give the user a sneak preview of the document contents and to help him/her make selections [5]. However, query-biased snippets are generated dynamically online, which will impose high computational cost when large amounts of results need to be processed.

Several recent attempts have been made on query model [6] and a range of query facilities [7] in personal dataspace management systems, and extended inverted list has been developed in Semex [8] for improving query efficiency. However, none of the above work has addressed the problem how to improve result quality in terms of query relevance in PDSMS.

We observe that unlike exploratory queries in web search, a user in a personal dataspace usually has a specific search target and wants to find some known items, which are stored by themselves [7]. In this situation, the user expects to retrieve items (which are documents in most cases) whose topics are relevant to given query keywords. The topic of a document is represented by the core content of a document. Inspired by the partial index technique in relational database management system [9], we propose an alternative approach for improving result relevance in personal dataspaces. We build compact indexes based on core content which can catch the most important and representative semantics in documents.

Being concentrated on most important semantics in a document, a compact index is succinct and with low space cost. In order to build a compact index, we extract core sentences from documents based on semantics, and those core sentences can also be used for producing query-biased snippets. The major contributions are summarized as follows:

1. To our best knowledge, we are the first to use a compact index on core content in documents to address keyword search effectiveness problem in PDSMS. Unlike common inverted indexes based on full text, a compact index is built on core content which can catch most important and representative semantics in documents.
2. We present algorithms for selecting core content based on semantic analysis, which can catch the most important semantics from documents and can process English and Chinese documents uniformly.
3. We introduce Versatile, a personal dataspace management system, in which our compact inverted indexes can capture not only core content but also structure information so that queries combining keyword and structure can also be supported. The Versatile system can also provide query-biased snippets based on core content, which can be generated more quickly and accurately than the ones using the whole documents.
4. Our comprehensive experiments show that the performance of processing keyword queries using a compact index is better than that using a full-text index in a personal dataspace. Because compact indexes are built on a small amount of core sentences, not only query precision but also response time are improved, which is significant for very large personal dataspaces. By our coverage experiments, compact indexes are proved feasible because topic keywords which can capture the most important and representative semantics in a document can be covered by compact indexes built on core sentences.

This paper is structured as follows. Section 2 introduces related work. Section 3 describes the method to build compact indexes based on core sentences in a personal dataspace. Section 4 gives an overview of Versatile – a personal dataspace management system architecture. Section 5 presents experimental results. Finally, Section 6 concludes this paper.

## 2 RELATED WORK

There are three categories of work most related to ours, which are partial indexes in relational database management system, extended inverted lists in dataspace management system and automatic text summarization.

The partial index technique is proposed by Michael Stonebraker, which has now been implemented in some relational database management systems such as SQL Server, Postgres [9, 10]. A partial index is built over a subset of table rows, which is defined by a conditional expression and contains entries only for those table rows that satisfy the conditional expression. The aim of building partial indexes in RDBMS is obvious for reducing the space cost and improving query efficiency, which is different from our main aim of building compact indexes in PDSMS for improving keyword search quality.

Dataspaces are collections of heterogeneous and partially unstructured data. Indexes based on extended inverted lists, which incorporate attribute labels, relations between data items and hierarchies of schema elements, have been proposed to support queries that combine keywords and structure [8]. Extended inverted indexes can improve the efficiency of queries combining keywords and structure in PDSMS; result quality in terms of query relevance has not been addressed in this paper. In the meantime, the space cost for extended inverted indexes is high. With the increase of personal data, volume of inverted indexes will become very large.

The goal of text summarization is to create compressed summaries which contain the most important and representative information [11]. There are two types of text summarization called extractive summarization and abstractive summarization. While an abstractive summarization aims to produce a grammatical summary and requires advanced language generation technique, most of researches focus on the technique of extractive summarization which relies on extraction of sentences. In order to obtain an extractive summary, statistics and machine learning technique are usually used [12, 13]. In order to improve the quality of summaries, the method based on semantic analysis [14, 15] is proposed by using semantic dictionary like Wordnet [16] and Hownet [17].

We are the first to introduce the idea of partial index into personal dataspace management system. Just like a partial index that does not contain all rows in a table, a compact index is a kind of extended inverted list built on core content that are the most representative and informative part in a document. Unlike partial indexes built on tables in RDBMS, a compact index is built on a resource net [18] which is a uniform data model to describe various resources and associations between resources in a personal dataspace, and can answer keyword queries with structure specifications more accurately and efficiently. In the process of core content extraction, we use text summarization technique for reference, but our system is semantic-based and can process English and Chinese documents uniformly.

## 3 COMPACT INDEXES BASED ON CORE SENTENCES

### 3.1 A Framework for Implementing a Compact Index

A compact index is a kind of extended inverted list which can capture keyword and structure in a personal dataspace, but it is concise because we build it on a set of selected sentences in documents which are called core sentences.

Figure 1 gives the framework for implementing compact indexes. The key problem is how to extract core sentences. We follow four steps to finish the extraction work. Text preprocessing is the first step, which includes tokenizing, removing stop words, and stemming. After that, similarities between pairs of sentences are calculated based on similarities between words. Then, in the third step, sentences are clustered based on improved $K$-Medoids clustering algorithm. Last, centroids of

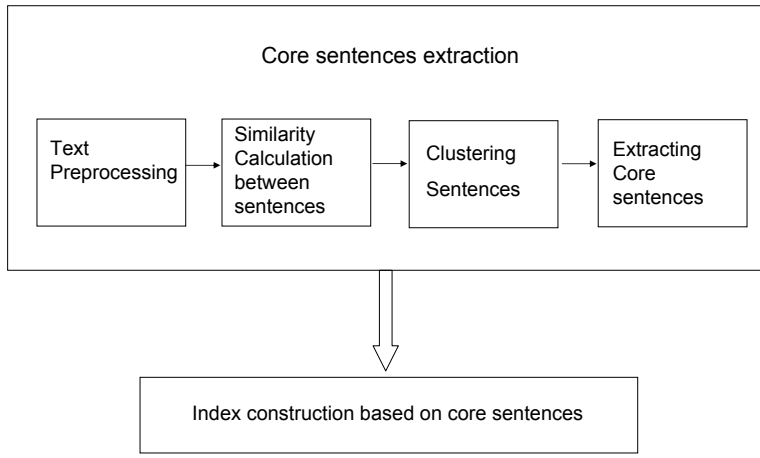



Figure 1. A framework for implementing compact indexes

clusters are selected as core sentences. Compact indexes are extended inverted lists built on selected core sentences.

### 3.2 Similarity Calculation between Sentences

In four steps for extracting core sentences, the technology of text preprocessing is common as in the area of natural language processing (NLP), so our work focuses on similarity calculation for sentences and clustering.

The selection of core sentences is based on centroids of clusters, and the basis of clustering is similarity calculation for sentences. There exist several methods for calculating sentences' similarity, of which the method based on semantic dictionary is used in Versatile because it takes semantics of words and expressions into account.

Sentence-sentence similarity can be calculated according to word-word similarity based on semantic dictionaries like Wordnet [16] and Hownet [17]. Hybrid method [19] is adopted to calculate word-word similarity based on Wordnet for English sentences in Versatile. For Chinese sentences, Hownet is used to calculate the similarity between a pair of Chinese words. The typical algorithm is proposed by Qun Liu and Sujian Li, based on the observation that the higher probability that words could be replaced in different contexts and the semantic structure of sentences could not be changed, the higher the similarity between two words [20].

Algorithm 1 gives sentence-sentence similarity calculation in Versatile. In order to simplify the algorithm, the most informative words are selected to take part in the calculation.The input parameters are two sentences $S_1$ and $S_2$, a language flag $F$. Algorithm 1 executes the following three steps and returns SIM as similarity value between $S_1$ and $S_2$.

First, it finishes tokenizing and identifying the different parts of speech by using the technology of NLP, then getting two nouns sets ($AttrN_1$, $AttrN_2$) and two verbs sets ($AttrV_1$, $AttrV_2$) from $S_1$ and $S_2$, respectively (Lines 2–5). In addition, Adjectives sets ($AttrA_1$, $AttrA_2$) are also obtained if input sentences are in Chinese (Lines 6–9).

Second, word-word similarity values are calculated for each pair of nouns from $S_1$ and $S_2$ and maintained in a two-dimensional array ArrN12 (Line 10). The most matched noun pairs with maximal similarity value are found, and similarity values of every matched noun pairs are put into an array ArrMaxN (Lines 11–16). We then calculate similarity value SimN for nouns according to similarity values in ArrMaxN (Line 17). In the same way, similarity value SimV for verbs can be calculated (Lines 18–25). For Chinese sentences, similarity value SimA for adjectives is also calculated (Lines 26–34).

Third, the sentence-sentence similarity value is obtained by calculating weighted average of SimN, SimV and SimA for a Chinese sentence (Line 35) and by calculating weighted average of SimN and SimV for an English sentence (Line 37).

### 3.3 Selecting Core Sentences

After getting sentence-sentence similarity values, we get core sentences by clustering. In various clustering algorithms, $k$-means algorithm is usually used to process large data set but can be sensitive to outliers (or noise). By using the most central object as a representative point of each cluster, $k$-medoids algorithm is generally more robust to outliers. One of the main factors to limit the use of $k$-medoids algorithm is its inefficiency compared with $k$-means. This drawback can be overcome with the aid of an efficient sampling scheme [21]. An improved $k$-medoids algorithm is proposed, in which initial centroids of clusters are decided based on the Min-Max clustering principle [22].

Core sentences extraction is centroid-based in general, and Algorithm 2 gives the details. The input parameters are: a set of initial sentences $ArrS$, number of initial sentences $N$, the proportion of core sentences $R$. The algorithm executes in three steps.

First, the number of clusters K is decided according to the input proportion of core sentences (Line 2), and initial centroids of clusters are selected based on the Min-Max clustering principle (Line 3).

Second, after getting centroids, we partition the remaining sentences into $K$ clusters based on the maximal similarity principle. That is, a sentence is put into a cluster when the centroid of the cluster is the most similar with this sentence (Lines 6–19). In each cluster, we replace the original centroid with other sentences in turn, and calculate delta which is an increased summation of similarity value in the cluster every time. We select the sentence as a new centroid in a cluster if it can make delta be maximum (Lines 21–33). If any centroid is changed, this step will be executed repeatedly from the beginning.

**Algorithm 1** Sentence-sentence Similarity Calculation Algorithm

**procedure** GETSIMSS($S_1, S_2, F, SIM$)
  $ArrN_1 \leftarrow getNouns(S_1, F)$
  $ArrN_2 \leftarrow getNouns(S_2, F)$
  $ArrV_1 \leftarrow getVerbs(S_1, F)$
  $ArrV_2 \leftarrow getVerbs(S_2, F)$
  **if** $F =$ 'Chinese' **then**
    $ArrA_1 \leftarrow getAdjs(S_1)$
    $ArrA_2 \leftarrow getAdjs(S_2)$
  **end if**
  $ArrN12 \leftarrow getSimWW(ArrN_1, ArrN_2, F)$
  $k \leftarrow 0$
  **while** $exists(ArrN12)$ **do**
    $ArrMaxN \leftarrow getMax(ArrN12, i, j)$
    $k \leftarrow k$+1
    $setArr(ArrN12, i, j)$
  **end while**
  $SimN \leftarrow sumMax(ArrMaxN)/K$
  $ArrV12 \leftarrow getSimWW(ArrV_1, ArrV_2, F)$
  $k \leftarrow 0$
  **while** $exists(ArrV12)$ **do**
    $ArrMaxV \leftarrow getMax(ArrV12, i, j)$
    $k \leftarrow k + 1$
    $setArr(ArrV12, i, j)$
  **end while**
  $SimV \leftarrow sumMax(ArrMaxV)/k$
  **if** $F =$ 'Chinese' **then**
    $ArrA12 \leftarrow getSimWW(ArrA_1, ArrA_2, F)$
    $k \leftarrow 0$
    **while** $exists(ArrA12)$ **do**
      $ArrMaxA \leftarrow getMax(ArrA12, i, j)$
      $k \leftarrow k + 1$
      $setArr(ArrA12, i, j)$
    **end while**
    $SimA \leftarrow sumMax(ArrMaxA)/k$
    $SIM \leftarrow \alpha_1 \times SimN + \alpha_2 \times SimV + \alpha_3 \times SimA$
  **else**
    $SIM \leftarrow \beta_1 \times SimN + \beta_2 \times SimV$
  **end if**
  **return** $SIM$
**end procedure**

**Algorithm 2** Core Sentences Extraction Algorithm

```
1: procedure getCoreSentence(ArrS, N, R, CS)
2:     K ← ⌈N × R⌉
3:     Medoids ← getInitCent(ArrS, K)
4:     F ← True
5:     while F do
6:         for all S ∈ Arrs do
7:             if S ∉ Medoids then
8:                 maxSim ← 0
9:                 N ← 0
10:                for i ← 0, K do
11:                    simSS ← getSimSS(Medoids[i], S)
12:                    if simSS > maxSim then
13:                        N ← i
14:                        maxSim ← simSS
15:                    end if
16:                end for
17:                Cluster[N] ← S
18:            end if
19:        end for
20:        F ← False
21:        for i ← 0, K do
22:            sumSim ← getSum(Medoids[i], Cluster[i])
23:            delta ← 0
24:            for all S ∈ Cluster[i] do
25:                repMed(Cluster[i], Medoids[i], S);
26:                newsumSim ← getSum(S, Cluster[i])
27:                if (newsumSim − sumSim) > delta then
28:                    delta ← newsumSim − sumSim
29:                    Medoids[i] ← S
30:                    F ← True
31:                end if
32:            end for
33:        end for
34:    end while
35:    CS ← Medoids
36:    return CS
37: end procedure
```

Third, after all centroids are not changed any more, they can be returned as core sentences (Lines 35–36).

## 4 VERSATILE PERSONAL DATASPACE MANAGEMENT SYSTEM ARCHITECTURE

Personal Information contains not only data created and managed by different applications, but also web data that a user often accesses. To offer users a flexible platform for personal information management, we are building the Versatile System, which provides a logical view of one's personal information and a flexible query model based on the logical view.

As shown in Figure 2, the architecture of Versatile contains 3 layers: data sources layer, Versatile PDSMS layer, and user interface layer. Our work focuses on the Versatile PDSMS layer, which is composed of 4 components (Information extraction & association construction manager, Index builder, Schema extraction & summarization manager, Versatile query language processor) and 5 repositories (Resource net repository, Core content repository, Compact inverted index, Resource summary repository, Metadata catalog).

Information Extraction & Association Construction Manager (IEACM) works for extracting information from various data sources such as local file system, DBMS, email system, and constructing semantic associations predefined by Versatile or defined by users while the system is running. The output of IEACM is a resource net [18], which is a graph with nodes representing resources in a personal dataspace and edges describing associations between resources. The resource net is put into resource net repository.

Based on the resource net, Index Builder (IB) extracts core content from resources and builds compact inverted indexes. Like extended inverted lists implemented in Semex [8], our compact inverted indexes can also capture structure information so that queries combining keyword and structure can be supported. Unlike an ordinary inverted index built on full text, a compact inverted index is based on core sentences which are the most informative and representative based on semantics.

Schema Extraction & Summarization Manager (SESM) works for extracting schema from a resource net and then generating its resource summary. A resource summary can provide a succinct overview of the entire personal dataspace and help structured query construction and optimization. While generating the resource summary, SESM also generates some mappings between schema and schema summary and puts them into metadata catalog.

Personal dataspace users often want to relocate expected items, but in most cases, their requirements could not be satisfied by keyword queries. Versatile provides a query language called VQL which can combine keyword and structure, and Versatile Query Language Processor analyses VQL and generates an execution plan based on 5 repositories.

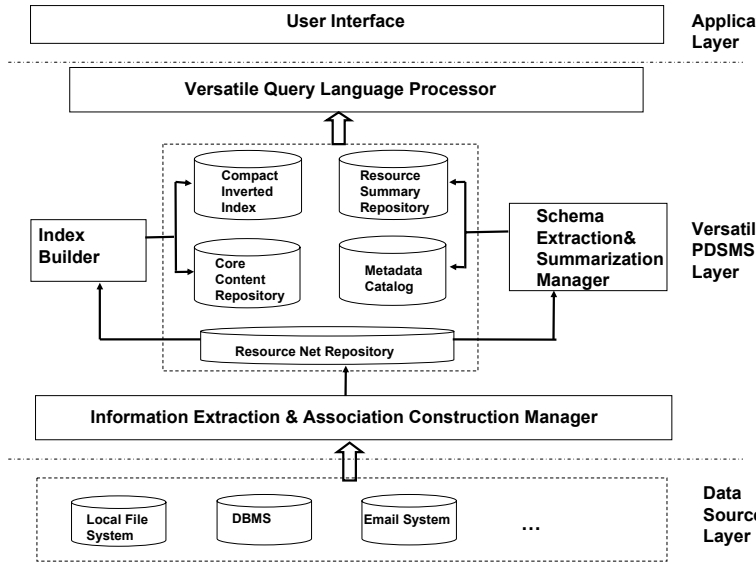



Figure 2. Versatile architecture

Compared with existing personal dataspace management system prototypes [6, 8, 7], Versatile has four innovative characteristics.

1. Compact inverted indexes are implemented based on core content. Using compact inverted indexes in personal dataspaces, keyword queries can be answered more accurately and quickly while index space cost and maintenance cost are reduced.

2. Our Versatile system can provide query-biased snippets upon core content which has been extracted during construction of compacted indexes. Compared with snippets upon whole documents, our query-biased snippets can be generated more quickly and accurately, because core content is concise in volume and concentrates on the most important semantics in a document.

3. Schema extraction and summarization based on user interests is implemented in Versatile PDSMS. Unlike XML schema extraction and summarization [23], Versatile considers uncertainty and user interests when extracting schema and building a resource summary.

4. Versatile provides a query language called VQL, which combines structured query with keyword query. Unlike iMeMex Query Language (iQL) based on large resource view graph [24], VQL can be written based on a concise resource summary. Not only actual schema element but also abstract schema element can be presented in VQL path expressions.

## 5 EVALUATING COMPACT INDEXES

The main aim of building a compact index is to improve the quality of keyword queries, especially the precision of return results in personal dataspaces. Unlike web search, a user in a personal dataspace usually wants to find some known items stored by him/herself. In this situation, the purpose of the user is to find a few documents whose core content contains keywords s/he issues, in other words, s/he only expects to get documents whose topics are relevant to given keywords.

The evaluation experiments are divided into two parts. One part of experiments aims at evaluating the effectiveness and efficiency of a compact index in personal dataspace system, by comparing its performance for queries with an inverted index based on full text. We have tested two metrics to compare the performance: the quality of the search results measured by precision, recall and F-measure, the response time for query. Another part of experiments aims at evaluating the feasibility of a compact index. By the experiments, we can see that topic keywords which can capture the most important and representative semantics in a document can be covered by a compact index built on a small fraction of core sentences.

We implemented the indexing module using the Lucene indexing tool [25]. We implemented our algorithms in Java, and conducted all the experiments on a 2.2 GHz Intel machine running Windows 7, with 2 CPUs, 2 GB memory and 320 GB hard disk.

### 5.1 Comparison with Full-text Indexes

In order to evaluate the effectiveness and efficiency of a compact index in personal dataspace system, we compare its performance for keyword queries with an inverted index based on full text.

#### 5.1.1 Experimental Data

Experiments for search effectiveness and efficiency were conducted on dataset DS0 which is composed of English data set and Chinese data set. English data set contains 400 documents which cover 15 topics from Reuters-21578 [26], and Chinese data set contains 200 documents which cover 10 topics from the Internet. Each document contains about 40 sentences.

#### 5.1.2 Precision, Recall & F-measure

To evaluate keyword query quality, we use precision, recall, and F-measure, defined in Equations (1), (2) and (3):

$$Precision = \frac{|Rel \cap Ret|}{|Ret|} \tag{1}$$

$$Recall = \frac{|Rel \cap Ret|}{|Rel|} \tag{2}$$

$$F = \frac{(1+\alpha) \times Precision \times Recall}{\alpha \times Precision + Recall} \quad (3)$$

Precision measures the percentage of the output documents that are desired, and recall measures the percentage of the desired documents that are output. F-measure is the weighted harmonic mean of precision and recall, in which precision and recall are evenly weighted if $\alpha = 1$. Additional two commonly used F-measures are $\alpha = 0.5$, assigning a larger weight to precision, and $\alpha = 2$, assigning a larger weight to recall.

In Equations (1) and (2), Rel is the set of relevant documents which are desired, and Ret is the set of output documents. A key problem is how to judge if a document is related to the query. In order to remove the difference between different standards from various users, top-$n$ keyword query results based on a full-text index constructed by Lucene indexing tool are set as ground truth of Rel. The top-$n$ query results, which are the most relevant in all outputs, are decided by Lucene ranking mechanism, and $n$ is set according to the number of documents in data set.

### 5.1.3 Experiments and Results

In the following experiments, a full-text index and a compact index are constructed for DS0 using Lucene indexing tool. For building a compact index, we try to extract about 10 core sentences in each document. As for how many core sentences are suitable for building a compact index, we will get suggestions from experiments in 5.2. We set the parameter of the number of returned documents as 1 000 so that every search result meeting search condition can be returned, and set $n$ as one third of the number of all returned documents for a query using full-text indexes. Under the above assumptions, the values of recall for queries using a full-text index will be unified as 1, and the values of precision for queries using a full-text index will be about 0.33.

Before experiments, documents in DS0 are analyzed manually and 10 most representative keywords are extracted from each document as topic keywords which can reflect the most important semantics of the document. After that, we select 5 topics from English and Chinese data sets, and we select randomly 2 or 3 topic keywords from each topic to compose 10 query cases. We select topic keywords to compose query cases in our experiments just for getting enough results.

10 keyword queries are evaluated, and their precision,recall and response time under two index approaches on DS0 are calculated and shown in Figures 3 a), 3 b) and 3 d). Furthermore, we compute the F-measure of each approach according to the average precision and recall across all the test queries with parameter $\alpha = 0.5$, 1 and 2, as presented in Figure 3 c).

From Figure 3, we have following observations:

1. The precision of queries using a compact index is obviously higher than that using a full-text index, because only semantically more important sentences are selected and used to build a compact index.

2. For F-measures with 3 different parameter values assigning different weight to precision and recall, our approach always outperforms the approach using a full-text index, although the recall of some query cases under a compact index is a little lower than that under a full-text index. Because we use top-$n$ keyword query results based on a full-text index constructed by Lucene indexing tool as ground truth of Rel, it is nothing surprising that the value of recall is biased towards full-text indexes. In fact, we can improve query recall by the way of extracting more core sentences for building compact indexes. In 5.2, some suggestions will be given about the setting for suitable number of core sentences.
3. Queries based on a compact index are more efficient than those based on a full-text index, because the volume of a compact index built on core sentences is greatly reduced.

As we can see, when using compact indexes in PDSMS, the comprehensive performance of queries is higher than using full-text indexes. Because compact indexes are built on a small amount of core sentences, not only query precision but also response time are improved, which is significant for very large personal dataspaces.

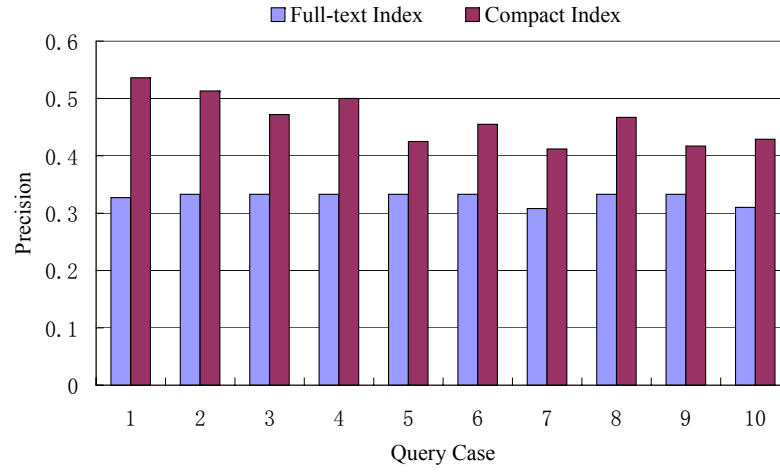
All that was missing was recall. In PDSMS, a user issues keyword queries and just wants to get a small amount of objects which are most relevant to topic keywords. For a document, topic keywords are limited and their coverage rate in a compact index can be improved through extracting more core sentences. Of course, we can also develop new algorithms for extracting core sentences with high quality, which is our future work.

### 5.2 Coverage of Topic Keywords

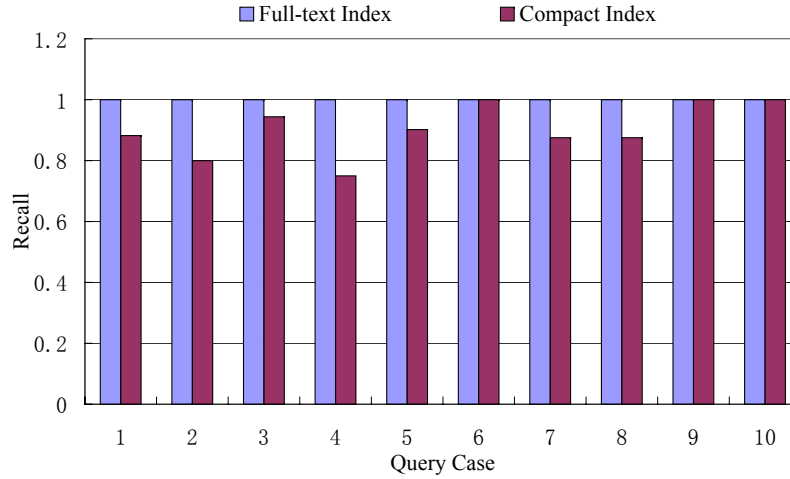
In order to improve the recall of keyword queries based on a compact index, we can select more core sentences. However, the question we are interested in is how many core sentences can cover topic keywords, and it is also about the feasibility of compact indexes. To keep its advantage, a compact index has to be built on a small amount of core sentences. In this section, we will validate this conclusion by experiments.

#### 5.2.1 Experimental Data

Experimental data set DS1 is composed of 210 documents collected from a personal desktop, which contain 105 English documents and 105 Chinese documents. To make experimental results representative, documents are selected according to 5 types. There are two types of short documents, which are about 1–2 pages or 4–6 pages; medium-length documents are about 12 pages; long documents are about 60 pages; super-long documents are about 200 pages.




a)

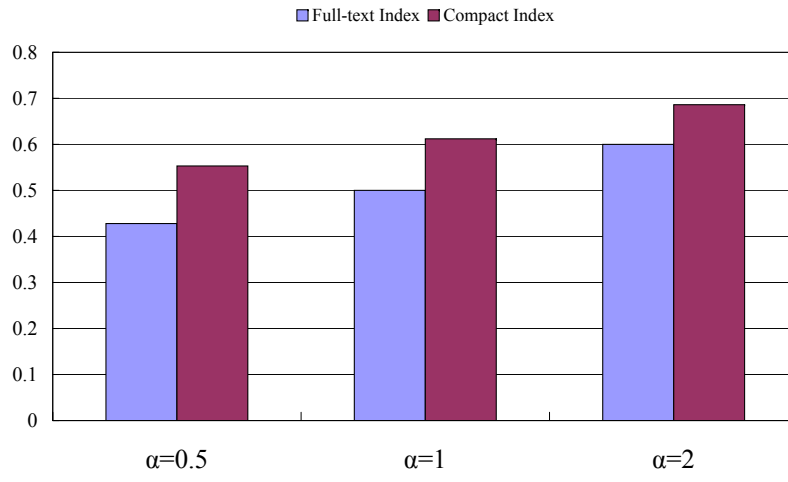



b)

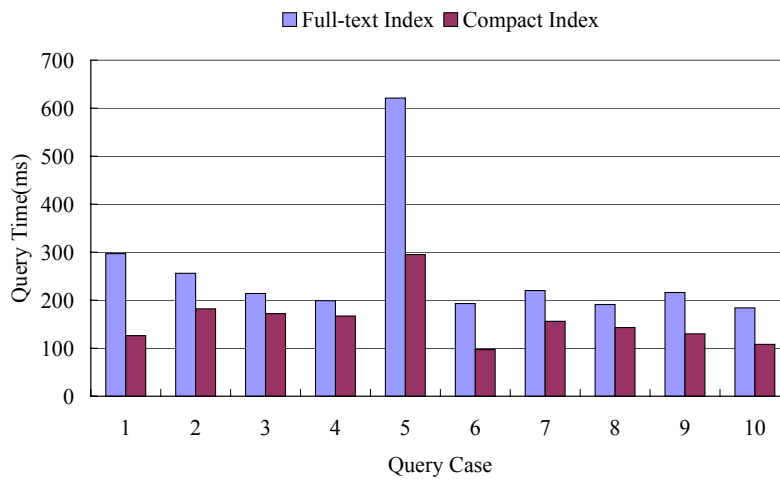
Our experiments totally process 10 types of documents, and the statistics thereof is shown in Table 1. For the same type of documents, we select those with the number of sentences as close as possible, so # of Sentences in Table 1 is an average sentence number for the same type of documents.

### 5.2.2 Selection of Topic Keywords

Before experiments, the first thing we have to make clear is how to select topic keywords. Topic keywords for a document are topic oriented and can reflect the most




c)




d)

Figure 3. Precision, Recall, F-Measure and Response Time under two index approaches: a) precision measurement, b) recall measurement, c) F-measurement with different $\alpha$ values, d) query time

important content of this document. Generally speaking, the content of a document can be observed upon three layers, just like a journal paper, which are keywords, abstract and full text. Full text can perfectly describe the whole content, but quite most words are not vital to identify the subject of the document. Keywords and abstract are two other layers of a document which can describe the most important content of the document succinctly. For a journal paper, the number of keywords is about 5, and does not exceed 10. Further observations show that 30 keywords

| | # of Sentences (per document) | # of documents |
|---|---|---|
| Short document1 (English) | 40 | 25 |
| Short document2 (English) | 100 | 25 |
| Medium-length document (English) | 200 | 25 |
| Long document (English) | 300 | 25 |
| Super-long document (English) | 1 500 | 5 |
| Short document1 (Chinese) | 40 | 25 |
| Short document2 (Chinese) | 100 | 25 |
| Medium-length document (Chinese) | 200 | 25 |
| Long document (Chinese) | 300 | 25 |
| Super-long document (Chinese) | 1 500 | 5 |

Table 1. Dataset DS1 statistics

are enough to cover whole content of the abstract of a common paper. Topic keywords of a document are classified into 3 types based on the number of keywords. In the following paper, the set of 5 topic keywords is called primary topic keywords; the set of 10 topic keywords is called secondary topic keywords; the set of 30 topic keywords is called extensive topic keywords. In fact, extensive topic keywords contain secondary topic keywords, and secondary topic keywords contain primary topic keywords which are the most pivotal for a document.

We enlisted three users to generate "gold standard" topic keywords for DS1. Based on understanding the main meaning of each document, users wrote down 3 types of topic keywords independently. In fact, because short documents have less sentences, it is unnecessary to provide extensive topic keywords. Finally, for each document, a user consensus topic keyword list for a particular type is generated by combining all users' topic keywords and retaining only keywords selected by most users (in this case, at least two users).

### 5.2.3 Experimental Results

For each type of English or Chinese documents, we extracted different number of core sentences from each document, and measured the coverage rate of 3 types of topic keywords in core sentences. The primary coverage rate is defined as the percentage of 5 primary topic keywords occurring in core sentences; the secondary coverage rate is defined as the percentage of 10 secondary topic keywords occurring in core sentences; the extensive coverage rate is defined as the percentage of 30 extensive topic keywords occurring in core sentences. For short documents, we have not measured their extensive coverage rates.

Tables 2 and 3 show the average primary coverage rates and secondary coverage rates respectively for 8 types of documents under different numbers of core sentences. For medium-length documents and long documents, Table 4 gives their average extent coverage rates. Finally, Table 5 gives 3 types of average coverage rates for super-long documents.

| # Core Sentences | Short Doc1 | | Short Doc2 | | M-length Doc | | Long Doc | |
|---|---|---|---|---|---|---|---|---|
| | En | Ch | En | Ch | En | Ch | En | Ch |
| 4 | 0.752 | 0.824 | 0.687 | 0.824 | 0.576 | 0.733 | 0.555 | 0.752 |
| 6 | 0.784 | 0.904 | 0.722 | 0.816 | 0.616 | 0.883 | 0.600 | 0.768 |
| 8 | 0.872 | 0.896 | 0.730 | 0.888 | 0.656 | 0.875 | 0.691 | 0.808 |
| 10 | 0.904 | 0.976 | 0.757 | 0.944 | 0.744 | 0.883 | 0.700 | 0.896 |
| 20 | 0.968 | 0.992 | 0.870 | 0.968 | 0.856 | 0.952 | 0.818 | 0.944 |
| 30 | 1 | 1 | 0.930 | 0.984 | 0.920 | 0.983 | 0.873 | 0.968 |
| 50 | 1 | 1 | 0.983 | 0.992 | 0.984 | 1 | 0.973 | 0.976 |

Table 2. Primary coverage rate

| # Core Sentences | Short Doc1 | | Short Doc2 | | M-length Doc | | Long Doc | |
|---|---|---|---|---|---|---|---|---|
| | En | Ch | En | Ch | En | Ch | En | Ch |
| 4 | 0.664 | 0.632 | 0.613 | 0.676 | 0.444 | 0.596 | 0.500 | 0.676 |
| 6 | 0.656 | 0.736 | 0.609 | 0.656 | 0.504 | 0.738 | 0.550 | 0.668 |
| 8 | 0.772 | 0.760 | 0.630 | 0.752 | 0.564 | 0.792 | 0.618 | 0.704 |
| 10 | 0.828 | 0.836 | 0.691 | 0.856 | 0.640 | 0.796 | 0.609 | 0.804 |
| 20 | 0.948 | 0.932 | 0.791 | 0.920 | 0.784 | 0.888 | 0.759 | 0.844 |
| 30 | 0.992 | 0.996 | 0.878 | 0.968 | 0.868 | 0.913 | 0.868 | 0.904 |
| 50 | 1 | 1 | 0.961 | 0.988 | 0.976 | 0.965 | 0.955 | 0.952 |

Table 3. Secondary coverage rate

| # Core Sentences | M-length Doc | | Long Doc | |
|---|---|---|---|---|
| | En | Ch | En | Ch |
| 6 | 0.368 | 0.472 | 0.414 | 0.541 |
| 8 | 0.401 | 0.535 | 0.494 | 0.572 |
| 10 | 0.513 | 0.557 | 0.515 | 0.637 |
| 20 | 0.664 | 0.661 | 0.662 | 0.713 |
| 30 | 0.744 | 0.768 | 0.732 | 0.801 |
| 50 | 0.917 | 0.867 | 0.870 | 0.865 |
| 80 | 0.927 | 0.931 | 0.911 | 0.923 |

Table 4. Extensive coverage rate

| # Core Sentences | Primary Topic Key | | Secondary Topic Key | | Extensive Topic Key | |
|---|---|---|---|---|---|---|
| | En | Ch | En | Ch | En | Ch |
| 30 | 0.92 | 0.94 | 0.90 | 0.91 | 0.713 | 0.763 |
| 50 | 0.96 | 0.96 | 0.94 | 0.95 | 0.853 | 0.877 |
| 80 | 0.96 | 0.98 | 0.98 | 0.98 | 0.900 | 0.913 |
| 100 | 1 | 1 | 0.98 | 0.99 | 0.933 | 0.943 |
| 150 | 1 | 1 | 1 | 1 | 0.980 | 0.987 |

Table 5. Coverage rates for super-long documents

From Tables 2 through 5, we have the following observations:

1. Except for very few instances, the more core sentences have been extracted, the higher the coverage rates are. Exceptions occur when not much difference exists in the number of core sentences between two extractions.

2. In order to approach one coverage rate, less proportion of core sentences are needed for longer documents. We can use secondary coverage rates for English documents as an example. For short documents with about 40 sentences, when 25 % sentences (10 sentences) are extracted from a document as core sentences, average coverage rate is 0.828. However, for medium-length documents with about 200 sentences, when 15 % sentences (30 sentences) are extracted as core sentences, average coverage rate can reach 0.868. For the same average coverage rate 0.868, only 10 % sentences (30 sentences) need to be extracted from long documents with about 300 sentences. Finally, in super-long documents with about 1500 sentences, when 2 % sentences (30 sentences) are extracted, average coverage rate can reach 0.9. It is obvious that our algorithm for building compact indexes is scalable to the length of documents.

3. Under the condition of extracting the same number of core sentences, the coverage rates for Chinese documents are usually a little higher than those for English documents. The reasons are manifold, but can be reduced to two. First, three enlisted users are all Chinese, so it is taken for granted that "gold standard" topic keywords for Chinese documents are more accurate than those for English. Second, our algorithm of extracting core sentences based on semantics performs a little better for Chinese documents than that for English ones. It is our future research plan to improve the effectiveness of compact indexes by exploring new algorithms for extracting core sentences.

4. Absolutely most topic keywords can be covered by extracting a few core sentences. As we can observe, if 50 core sentences are extracted, average primary coverage rates for all types of documents are over 96 %, average secondary coverage rates for them are over 94 %, and average extensive coverage rates for them are over 85 %. In most cases, 10 secondary topic keywords are enough to cover the subject of a document. In other words, a compact index building upon a few core sentences (50 sentences for example) can meet most of the needs from users.

In fact, our method is mainly targeted for result accuracy but not query coverage. In this point, it is like Hilltop algorithm [27] which uses expert documents to assess relevancy. When there are not enough expert sites, Hilltop returns no results. For applications where query coverage is important, two steps could be taken in query processing. First we can use compact index to find high quality results efficiently, then we can resort full-text index for more results to improve recall.

## 6 CONCLUSION AND FUTURE WORK

In order to improve the search precision and efficiency of keyword query over personal dataspaces, we propose to construct compact indexes based on core content of documents. By extracting core sentences based on semantics from documents, a compact inverted index will concentrate on the most important semantics in a document, so it is succinct, with low space cost. Furthermore, we present algorithm for selecting core sentences based on semantic analysis, which can process English and Chinese documents uniformly. In particular, core sentences can also be used for producing query-biased snippets efficiently and accurately, which can give users a sneak preview of the document contents and help them to make selections. The results of a set of experiments show that the search precision and efficiency for keyword queries can be improved by using compact indexes in PDSMS. Last, the feasibility of compact indexes is validated by our coverage experiments, which reveal that a compact index building upon a few core sentences can meet most of the needs from users.

In the future, we plan to improve our algorithm of extracting core sentences to make it more efficient and effective, and we will study how to make use of query log to adjust the compact index so that the content that a user is more interested in will be indexed in order to further improve search quality. We hope the application scope of our method can be extended to vertical search engines in the near future.

### Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 61370060, Jiangsu Provincial Natural Science Foundation of China under Grant No. BK2011454 and the Open Foundation from Computer Application Technology in the Most Important Subjects of Zhejiang under Grant No. 10Y0001.

## REFERENCES

[1] Franklin, M.—Halevy, A.—Maier, D.: From Databases to Dataspaces: A New Abstraction for Information Management. SIGMOD Record, Vol. 34, 2005, No. 4, pp. 27–33.

[2] White, R. W.—Ruthven, I.—Jose, J. M.: Finding Relevant Documents Using Top Ranking Sentences: An Evaluation of Two Alternative Schemes. In: Proceedings of SIGIR '02 Conference, 2002, pp. 57–64.

[3] Xiang, B.—Jiang, D.—Pei, J.—Sun, X.—Chen, E.—Li, H.: Context-Aware Ranking in Web Search. In: Proceedings of SIGIR '10, 2010, pp. 451–458.

[4] Dong, A.—Chang, Y.—Zheng, Z.—Mishne, G.—Bai, J.—Zhang, R. et al: Towards Recency Ranking in Web Search. In: Proceedings of WSDM '10, 2010, pp. 11–20.

[5] Ceccarelli, D.—Lucchese, C.—Orlando, S.—Perego, R.—Silvestri, F.: Caching Query-Biased Snippets for Efficient Retrieval. In: Proceedings of EDBT 2011, 2011, pp. 93–104.

[6] Dittrich, J.-P.—Vaz Salles, M. A.—Blunschi, L.: iMeMex: From Search to Information Integration and Back. IEEE Data Engineering Bulletin, Vol. 32, 2009, No. 2, pp. 28–35.

[7] Li, Y.—Meng, X.—Kou, Y.: An Efficient Method for Constructing Personal Dataspace. In: Proceedings of the 6th Web Information Systems and Applications Conference, 2009, pp. 3–8.

[8] Dong, X.—Halevy, A.: Indexing Dataspaces. In: Proceedings of SIGMOD '07 Conference, 2007, pp. 43–54.

[9] Seshadri, P.—Swami, A.: Generalized Partial Indexes. In: Proceedings of ICDE '95, 1995, pp. 420–427.

[10] Stonebraker, M.: The Case for Partial Indexes. ACM SIGMOD Record, Vol. 18, 1989, No. 4, pp. 4–11.

[11] Das, D.—Martins, A.: A Survey on Automatic Text Summarization. Technical Report, Language Technologies Institute, Carnegie Mellon University, 2007.

[12] Conroy, J. M.—O'Leary, D. P.: Text Summarization via Hidden Markov Models. In: Proceedings of SIGIR '01, 2001, pp. 406–407.

[13] Radev, D. R.—Hovy, E.—McKeown, K.: Introduction to the Special Issue on Summarization. Computational Linguistics, Vol. 28, 2002, No. 4, pp. 399–408.

[14] Silber, H. G.—McCoy, K. F.: Efficiently Computed Lexical Chains as an Intermediated Representation for Automatic Text Summarization. Computational Linguistics, Vol. 28, 2002, No. 4, pp. 487–496.

[15] Chen, Y.—Liu, B.—Wang, X.: Automatic Text Summarization Based on Textual Cohesion. Journal of Electronics (China), Vol. 24, 2002, No. 3, pp. 338–346.

[16] Miller, G. A.: Wordnet: A Lexical Database for English. Communications of the ACM, Vol. 38, 1995, No. 11, pp. 39–41.

[17] Hownet web site. Availaible on: `http://keenage.com/`.

[18] Ning, W.—De, X.—Baomin, X.: Collaborative Integration and Management of Community Information in the Cloud. In: Proceedings of International Conference on E-Business and E-Government, Guangzhou, 2010, pp. 1406–1409.

[19] Jiang, J. J.—Conrath, D. W.: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In: Proceedings of International Conference on Research in Computational Linguistics, 1997, pp. 19–33.

[20] Liu, Q.—Li, S.: Word Similarity Computing Based on How-Net. Computational Linguistics and Chinese Language Processing, Vol. 7, 2002, No. 2, pp. 59–76.

[21] Chu, Sh.-Ch.—Roddick, J. F.—Pan, J.-Sh.: Novel Multi-Centroid, Multi-Run Sampling Schemes for $k$-Medoids-Based Algorithms. KES Journal, Vol. 8, 2004, No. 1, PP. 45–56.

[22] Ding, Ch. H. Q.—He, X.—Zha, H.—Gu, M.—Simon, H. D.: A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering. In: Proceedings of 2011 IEEE International Conference on Data Mining, 2011, pp. 107–114.

[23] Cong Yu, H. V.: Jagadish: Schema Summarization. In: Proceedings of VLDB 2006, 2006, pp. 319–330.

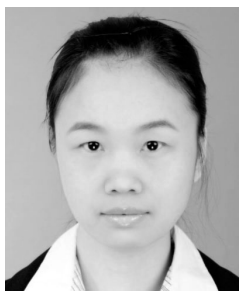
[24] Dittrich, J.-P.—Vaz Salles, M. A.: iDM: A Unified and Versatile Data Model for Personal Dataspace Management. In: Proceedings of the VLDB06 Conference, 2006, pp. 367–378.

[25] Lucene web site. Availaible on: `http://jakarta.apache.org/lucene/docs/index.html`.

[26] Reuters web site. Availaible on: `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`.

[27] Hilltop web site. Availaible on: `ftp://ftp.cs.toronto.edu/pub/reports/csri/405/hilltop.html`.

**Ning Wang** received her Ph. D. degree in computer science in 1998 from Southeast University in Nanjing, China. She is currently serving as an Associate Professor in School of Computer and Information Technology, Beijing Jiaotong University, China. Her research interests include web data integration, web search, XML keyword search and personal information management.



**Hongfang Du** received her Master degree in computer science from Beijing Jiaotong University in 2011 and currently works in Software Center of Bank of China. Her research interests include web data integration, data mining and personal information management.

**Baomin Xu** received his Ph. D. degree in computer science in 2000 from the Institute of Computing Technology at the Chinese Academy of Sciences, China. He is currently serving as an Associate Professor in School of Computer and Information Technology, Beijing Jiaotong University, China. His research areas include data mining, distributed computing including cloud computing and P2P.

**Guojun Dai** received his Ph. D. degree from the College of Electrical Engineering, Zhejiang University in 1998. He is currently a Professor and the Deputy Dean of the College of Computer Science, Hangzhou Dianzi University, Hangzhou, China. He is the author or co-author of more than 20 papers and books in recent years, and holds more than 10 patents. His research interests include biomedical signal processing, computer vision, embedded systems design, and wireless sensor networks.