Searching the Web with Hand-Handled Devices

Dorin Sima, "Lucian Blaga" University, Bd. Victoriei No. 10, Sibiu, Romania, email: dorin.sima@ulbsibiu.ro
Rodica Baciu, "Lucian Blaga" University, Bd. Victoriei No. 10, Sibiu, Romania, email: rodica.baciu@ulbsibiu.ro

Abstract: In this paper we describe a Java based meta-search engine designed to be used on hand-handled devices. The main problem for these devices (the problem of information visualisation on small size screen) was solved using two techniques: documents clustering and text summarisation.

Keywords: Information Visualization, Hierarchical Clustering, LSA, Galois Lattices and Summarization

INRODUCTION

It is well known that the task of finding the right information on the Web isn't an easy one. The result depends on the performances of chosen search engines and the ability of the user to put a right query and to navigate on the returned results set. Very large data sets returned by the search engine are problematic to display even on desktop screens. For a hand-handled device this is more problematic: one main characteristic of the hand-handled devices is that they have small screens compared to the existing desktop environment displays.

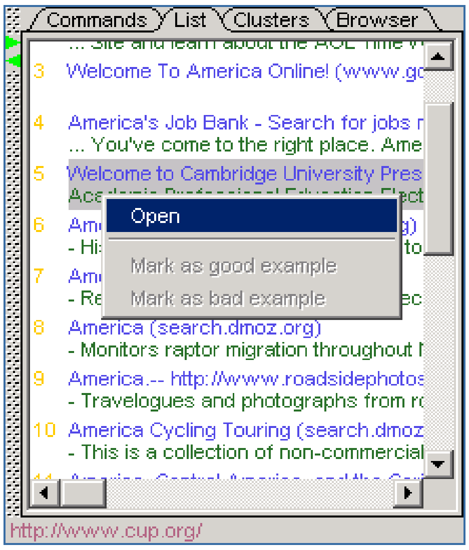
In this paper we describe a Java based meta-search engine designed to be used on hand-handled devices. The system has a Client/Server architecture: the server part (meta-search engine) runs on a desktop computer and the client, responsible with visualisation, runs on any Java enabled computer, including had-handled like iPAQ.

As a meta-search engine the system could be configured to work with different regular search engines. Each regular search engine (google, yahoo, etc.) "has a unique index of pages, and different relevance algorithms. Because of this, you often get very different results using the same query words on different engines. If you're not finding what you're looking for, stop banging away on your "favourite" and try another engine! " (Chris Sherman - What's the Best Search Engine?).

More, a search engine identifies "static" pages, rather than the "dynamic" information stored in databases. A meta-search engine could be configured to access a so-called "invisible Web".

The problem of visualisation on small devices was solved here using two techniques:

- the result set returned by meta-search engine is shown in different ways: as a list or clustered (hierarchical or conceptual-based on Galois Lattices)
- the user may see only the summarised information from target pages



... Site and learn about the AOL Time W
3 Welcome To America Online! (www.go
4 America's Job Bank - Search for jobs n
... You've come to the right place. Ame
5 Welcome to Cambridge University Pres
Aca
6 Ame g)
- Hi to
7 Ame
- Re ec
8 America (search.dmoz.org)
- Monitors raptor migration throughout N
9 America.-- http://www.roadsidephotos
- Travelogues and photographs from ro
10 America Cycling Touring (search.dmoz
- This is a collection of non-commercial
http://www.cup.org/

Fig. 1 The result set returned by search engines

BACKGROUND

Traditional search engines provide users with a way of locating interesting documents related to a query, using certain keywords to search for. Usually conventional retrieval systems return long lists of ranked documents that users are forced to scan through to find the relevant documents (fig. 1).

We must carry out a representation of "*documents*" that allows clustering. In this paper the term *document* denotes the text in the description part of each item returned by a search engine.

Documents representation

In the vector-space model (VSM) both the documents and requests are represented as m-dimensional vectors. In this case the attached weights are normalised sub-unitary positive numbers.

$D_j = (w_{1j}, w_{2j}, \ldots, w_{mj})$, - where wij represents the weight of i-*th* term in the j-th document.

Therefore we can rigorously define the similarity between two documents or between a document and a query as a scalar product or a cosines between two m-dimensional vectors. Using the following notations, we can define the cosine between documents:

$w_{ij} = tf_{ij} * idf_i = tf_{ij} * \log_2 (N/ df_i)$ where**:**

w_{ij} – is the weight of term *i* in document *j*

$tf_{ij} = f_{ij} / max\{f_{ij}\}$ local weight of the term *i* ;

f_{ij} –frequency of the term *i* in document *j*

df_i – number of documents containing term *i*

N – all documents

$$\mathrm{Cos}(D_i, D_j) = \frac{\sum_{i=1}^{m}(w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{m} w_{ij}^2 \cdot \sum_{i=1}^{m} w_{iq}^2}}$$

Hierarchical Clustering

Clustering starts with a set of singleton clusters, each containing a single document di D, i=1, ..., N, where D equals the entire set of documents and N equals the number of all documents. The two most similar clusters over the entire set D are merged to form a new cluster that covers both. This process is repeated for each of the remaining N-1 documents. The process stops when the cosine similarity between the closest clusters is less than some ***min*** value. In figure 2 this happens after step 3.

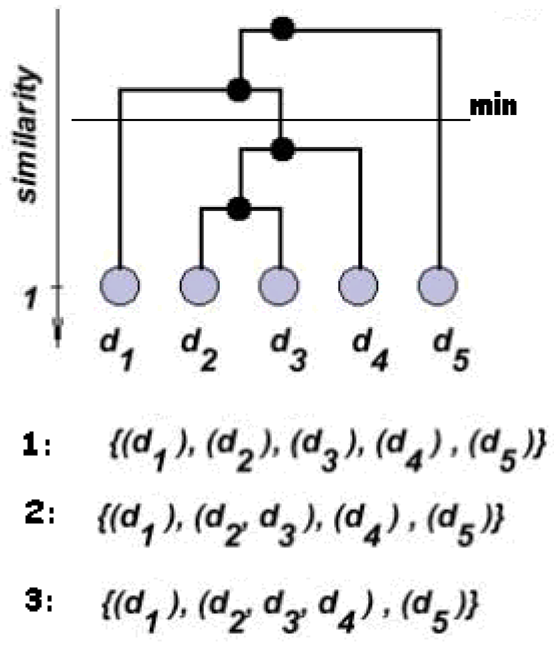

similarity
min
1
d1 d2 d3 d4 d5
1: {(d1), (d2), (d3), (d4), (d5)}
2: {(d1), (d2, d3), (d4), (d5)}
3: {(d1), (d2, d3, d4), (d5)}

Fig. 2 Hierarchical documents clustering

Latent Semantic Indexing

This approach of using terms as the descriptors of a document has certain drawbacks. Its major limitation is that it assumes that terms are independent. But some terms are likely to co-occur in documents about given topic because they all refer to aspects of that topic. To capture these term-term statistical relationships, the Latent Semantic Indexing (LSI) method is used. LSI is based on a matrix decomposition method called Singular Value Decomposition (SVD). Having a term-document matrix A, SVD computes three matrices U, S and V such that:

$$\mathbf{A = U*S*V^T},\ \text{where}$$

A is an **m x n** matrix that represents the **n** documents containing **m** words. Then rank of the matrix A is ***r***.

U is an **m x m** orthogonal matrix, having the left singular vectors of A as its columns

S is the diagonal matrix having the ***r*** nonzero singular values of A in order along its diagonal.

V is an **r x n** orthogonal matrix, having the right singular vectors of A as its columns.

In LSI, the rank-*k* approximation of the original matrix A is computed by using k-largest singular values. This corresponds to a projection of original *m*-dimensional vectors (documents) in a *k*-dimensional space, where k is much smaller than *m*. As result minor differences in words usage will be ignored (Berry 1994).

Formal Concept Analysis

An alternative to HCA is based on *Formal Concept Analysis:* a way to find, structures, and display relationships between concepts, which consist of attributes and objects. Concept analysis is a mathematical technique that starts with a set of objects, each of which has a set of attributes. A “concept” is simply the subset containing all objects that have a particular subset of attributes. It turns out that a lattice (i.e., a tree-like structure in which each node can have more than one parent) can be formed of concepts, each of which contains its children in the lattice. In other words, the top node of the tree represents all of the objects, and each branch down the tree reduces the number of objects by adding one or more attributes to the concept definition.

More formally (Miller):

A formal context (FC) is a triple (G, M, I) which consists of a set G of objects, a set M of attributes and a binary (incidence) relation $I \subseteq G\times M$ between objects and attributes. In our case the objects will be the *documents* (the description parts of each item in the results set returned by search engines) and the “keywords” extracted from that documents are attributes.

A concept (A, B) is defined as a pair of objects $A \subseteq G$ and attributes $B \subseteq M$, which fulfil certain conditions. A is called extent and B is called intent of the concept. To define the necessary and sufficient conditions for a formal context we present two derivation operators. Given $A \subseteq G$ we define

$A' := \{m \in M | \forall g \in A: (g, m) \in I\}$

and dually for $B \subseteq M$

$B' := \{g \in G | \forall m \in B: (g, m) \in I\}$.

A' contains all attributes that are common to all objects in A. And B' is the set of all objects that carry all the attributes of B.

With that, the pair (A, B) is a formal concept if

A' = B and A = B'.

This property says that all objects of the concept carry all its attributes and that there is no other object in G carrying all attributes of the concept. Looking at the definition of a formal concept one can easily see that for all $A \subseteq G$ the pair (A'', A') is a formal concept. The dual holds for all B ⊆M, i.e. (B', B") is always a formal concept, too. Yet, the sets of concepts achieved in this way are equal and contain exactly the concepts existing in the given context.

For formal concepts a subconcept/superconcept relationship ≤ can then be defined as follows:

(A1, B1) ≤(A2, B2) ⇔A1 ⊆A2 (⇔B2 ⊆B1)

This relationship shows the dualism that exits between attributes and objects of concepts. A concept C1= (A1, B1) is a subconcept of concept C2=(A2, B2) if the set of its objects is a subset of the objects of C2. Or an equivalent expression is if the set of its attributes is a superset of the attributes of C2. Actually, the set of all formal concepts of a context forms a so-called concept lattice. The infimum of this lattice is formed by (∅, M) and its supremum is formed by (G, ∅) if the context is given by (G, M, I).

For example, in table 1, is shown a formal context that corresponds to the documents d1, d2, d3 and d4 containing the keywords as is shown in table 1.

The corresponding Galois lattice, in figure 3 describes the relation between concepts. This structure is useful to easy navigate to a particular concept guided by the keywords.

	Java	OOP	C++	OOD	café	island
d1	X	X				
d2		X	X	X		
d3	X				X	
d4	X					X

Table 1. Formal context example

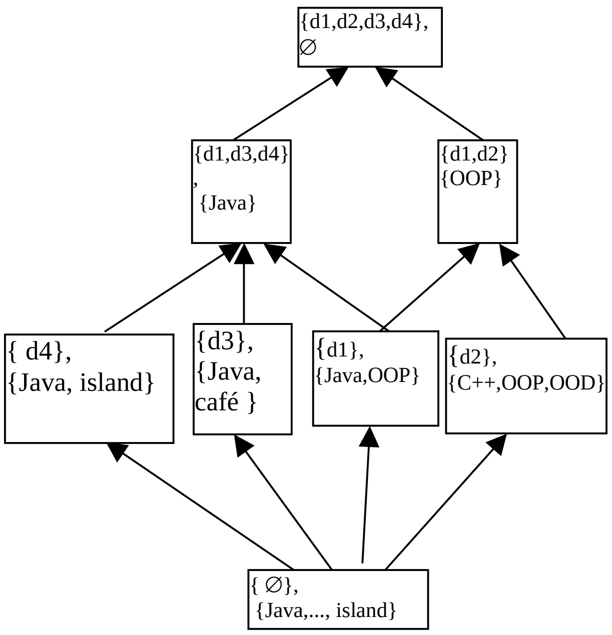


Fig. 3 Galois lattice for FC in table 1

Text summarization

Text summarization is the process of identifying salient concepts in text narrative, conceptualizing the relationships that exist among them and generating concise representations of the input text that preserve the gist of its content. We will use the term text summarization, as in literature, but in fact this is an approximation called some time *extraction* that is more feasible today. To create an extract, a system needs simply to identify the most important/topical/central topic(s) of the text, and return them to the reader. Although the summary is not necessarily coherent, the reader can form an opinion of the content of the original. This process is useful for hand handled devices because is hard to display all information contained in html pages.

The summarization algorithm contains five main steps:

1. identify the sentences in text
2. weighting of each sentence
3. sort the list of sentences according to the weights
4. select the main sentences
5. the remaining sentences are sorted in the order of apparition in the main text

Fore more details about summarization can be founds in (Buyukkokten 2011;Stoffer)

SYSTEM DESCRIPTION

Our system, called SmartSearch, is a client/server application that should offer information from the user interest domain. The main components of the system as shown in figure 4, are:

• SmartSearch Server
• SmartSearch iPaq Client
• SmartSearch Engine Descriptor

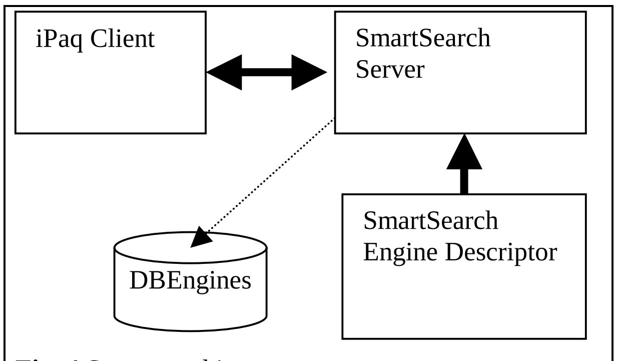
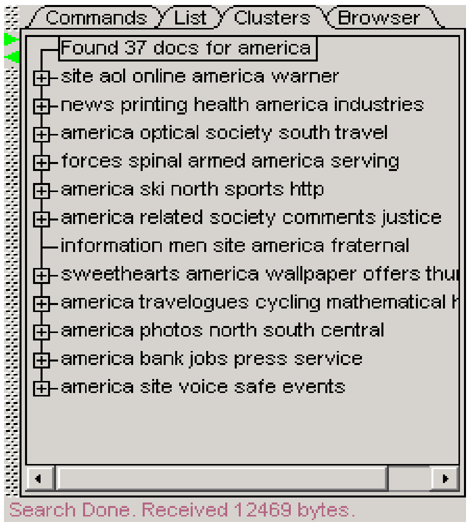


Fig. 4 System architecture

The server can respond to the user requests. These requests are made by the Client application (login, search, etc) or by the Description Tools applications. When the server receives a search request from the client, the query is identified and that query string is send to each search engine selected by the user in his profile. The server gets the results pages, which will then be analyzed and the results will be sent to the client in three formats:

- as a list of all results, sorted by some criteria
- as a set of clusters. The clustering is done using hierarchical clustering on a set of vectors computed by SVD (Lerman 1999)
- as the Galois lattice built from the results



Found 37 docs for america
site aol online america warner
news printing health america industries
america optical society south travel
forces spinal armed america serving
america ski north sports http
america related society comments justice
information men site america fraternal
sweethearts america wallpaper offers thu
america travelogues cycling mathematical h
america photos north south central
america bank jobs press service
america site voice safe events

Search Done. Received 12469 bytes.

Fig. 5 Hierarchical clustering visualisation

When the server receives a request for a particular page from the client, the server gets the specified page and returns to the client that page or the summary of the page, depending of the request type.

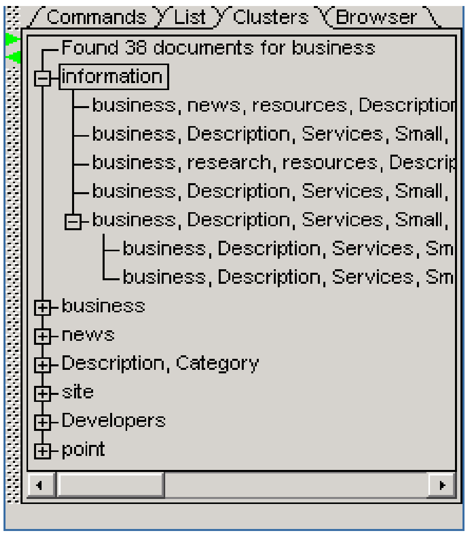
Because the most work is done on the server module, the module "iPAQ Client" becames very small. The main responsibility of this module is to implement the user interface for small devices. The user can select to see the results set of search as list, as is shown in figure 1. In this mode, a list of all found results will be displayed. By double clicking an item in the list a menu will appear; the user can select "Open" in order to view that page in the Browser window.

The *Hierarchical* clustering is displayed in a tree view. Each visible node represents a cluster. The cluster name (what is written) is composed by all the keywords contained in that cluster. Clicking on a "+" sign expands that node and allows you to see all the document titles contained by that cluster (fig. 5).

The Galois clustering is displayed in a tree view (fig. 6). This is the tree representation of the Galois lattice. Each visible node represents a concept: all documents that contain the keyword(s) labelling this node. As we said above, each document is the text of one result returned by search engines. Clicking on a "+" sign expands that node and allows you to see the subconcepts of this one, i.e. a subset of documents that all contains additional words.

CONCLUSIONS

In this paper we described some techniques used in Smart Search system to facilitate the task of finding the right information in the list of the results returned by search engines. If the number of results is small, the best option is to show them as a list. If this list is very large, clustering the results helps the user to find easier the desired information. Depending on the user's purpose, the hierarchical clustering could be more useful as the



Found 38 documents for business
information
business, news, resources, Descriptio
business, Description, Services, Small,
business, research, resources, Descri
business, Description, Services, Small,
business, Description, Services, Small,
business, Description, Services, Sm
business, Description, Services, Sm
business
news
Description, Category
site
Developers
point

Fig. 6 Tree visualisation of Galois lattice

Galois clustering and vice-versa. According to the user's wish, the searching results could be displayed as a list, as hierarchical clusters or as Galois lattice. Together with summarisation, the clustering has the advantage of displaying large amounts of data on a small area, thus being useful for the hand-handled devices.

REFERENCES

Berry M. and Dumais S., 1994 - Using Linear Algebra for Intelligent Information Retrieval, CS-94-270, Tenessee Univ., berry@cs.utk.edu,

Buyukkokten, Orkut, 2001 - Accordion Summarization for End-Game Browsing on PDA and Cellular Phones, Proc. Of the Human Factors in Computing Systhems, CHI-01

Lerman Kristina , 1999 - Document Clustering in Reduced Dimension Vector Space - USC Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292

Miiller, T – Concept Lattices for Web Searching – http://www.cc.gatech.edu/~tomiller/concepts.pdf

Stoffer, S; Holley K – Text Summarization Presentation Http://www.cs.unm.edu/~storm/TSPresent