Knowl Inf Syst (2012) 30:493–525
DOI 10.1007/s10115-011-0388-2

REGULAR PAPER

On exploiting static and dynamically mined metadata for exploratory web searching

Panagiotis Papadakos · Nikos Armenatzoglou · Stella Kopidaki · Yannis Tzitzikas

Received: 23 December 2009 / Revised: 17 January 2011 / Accepted: 22 February 2011 /
Published online: 22 March 2011
© Springer-Verlag London Limited 2011

Abstract Most Web Search Engines (WSEs) are appropriate for focalized search, i.e., they make the assumption that users can accurately describe their information need using a small sequence of terms. However, as several user studies have shown, a high percentage of search tasks are exploratory, and focalized search very commonly leads to inadequate interactions and poor results. This paper proposes exploiting *static* and *dynamically mined* metadata for enriching web searching with *exploration* services. Online *results clustering*, which is a mining task of dynamic nature since it is based on query-dependent snippets, is useful for providing users with overviews of the top results and thus allowing them to restrict their focus to the desired parts. On the other hand, the various *static metadata* that are available to a search engine (e.g., domain, language, date, and filetype) are commonly exploited only through the advanced (form-based) search facilities that some WSEs offer (and users rarely use). We propose an approach that combines *both* kinds of metadata by adopting the interaction paradigm of dynamic taxonomies and faceted exploration, which allows the users to restrict their focus gradually using both static and dynamically derived metadata. Special focus is given on the design and analysis of incremental algorithms for speeding up the exploration process. The experimental evaluation over a real WSE shows that this combination results to an effective, flexible, and efficient exploration experience. Finally, we report the results of a user study indicating that this direction is promising in terms of user preference, satisfaction, and effort.

Keywords Web searching · Interactive information retrieval · Results clustering · Faceted and dynamic taxonomies

P. Papadakos · N. Armenatzoglou · S. Kopidaki · Y. Tzitzikas (✉)
Institute of Computer Science (ICS),
Foundation for Research and Technology – Hellas (FORTH),
Vassilika Vouton, 700 13 Heraklion, Crete, Greece
e-mail: tzitzik@ics.forth.gr

P. Papadakos · N. Armenatzoglou · S. Kopidaki · Y. Tzitzikas
Computer Science Department, University of Crete,
L. Knossou, 714 09 Heraklion, Crete, Greece

1 Introduction

Web Search Engines (WSEs) typically return a ranked list of documents that are relevant to the submitted query. For each document, its title, URL, and *snippet* (fragment of the text that contains keywords of the query) are usually presented. It is observed that most users are impatient and look only at the first results [1]. Consequently, when either the documents with the intended (by the user) meaning of the query words are not in the first pages or there are a few dotted in various ranks (and probably different result pages), it is difficult for the user to find the information he really wants. The problem becomes harder if the user cannot guess additional words for restricting his query, or the additional words the user chooses are not the right ones for restricting the result set.

One solution is to try exploiting the various static metadata that are available to WSEs, like domain, dates, language, filetype, and predefined categories [2,43]. Such metadata are usually exploited through the advanced search facilities that some WSEs offer, but users very rarely use these services. A more flexible and user-friendly approach is to exploit such metadata in the context of the interaction paradigm of *faceted (or multidimensional) and dynamic taxonomies (FDT)* [38], a paradigm that is used more and more nowadays. In brief, the main advantages of this interaction paradigm are as follows: (a) it provides an overview of the returned answer (active dimensions, active values, and count information), (b) it releases the user from the effort of formulating metadata-based queries for locating resources or for restricting his focus, since he only has to click on the, so-called, *zoom points*, usually displayed at the left bar of the GUI (which also make evident how many hits he will get), (c) it does not disappoint the user since clicks always lead to resources (no empty results ever), (d) it is session-based (in contrast to the state-less query response) paradigm, thus allowing the user to reach his target gradually (through the so-called information thinning process). This model can host and exploit the various static metadata of a WSE.

Another solution to the problem is *results clustering* [49] that provides an overview of the search results. It aims at grouping the results into topics, called *clusters*, with predictive names (labels), aiding the user to locate quickly documents that otherwise he would not practically find especially if these documents are low ranked (and thus not in first result pages). The snippets, the cluster labels, and their structure is one instance of what we call *dynamically mined metadata*. We use this term to refer to metadata that are query-dependent, i.e., they cannot be extracted/mined a priori. Such metadata are usually mined only from the top-K part of a query answer because it would be unacceptably expensive (for real-time interaction) to apply these tasks on large quantities of data.

We propose exploiting both explicit (static) and mined (dynamically) metadata during web searching. The proposed process for exploratory web searching is sketched at Fig. 1. Since the first two steps of that process correspond to the *query-and-answer* interaction that current WSEs offer (which is stateless), what we propose essentially extends and completes the current interaction.

We focus on combining automatic results clustering with the interaction paradigm of FDT, by clustering *on demand* the top elements of the current answer (focus set). To speedup the interaction, we provide an *incremental* evaluation algorithm for reusing past dynamically mined metadata, and we report detailed performance results that demonstrate the feasibility and the efficiency of the approach. We have already applied this approach over a WSE, and Fig. 2 shows the GUI of that engine (to the best of our knowledge, there are no other WSEs that offer the same kind of information/interaction). Finally, we report the results of a user study that we have conducted for investigating user satisfaction, and degree of

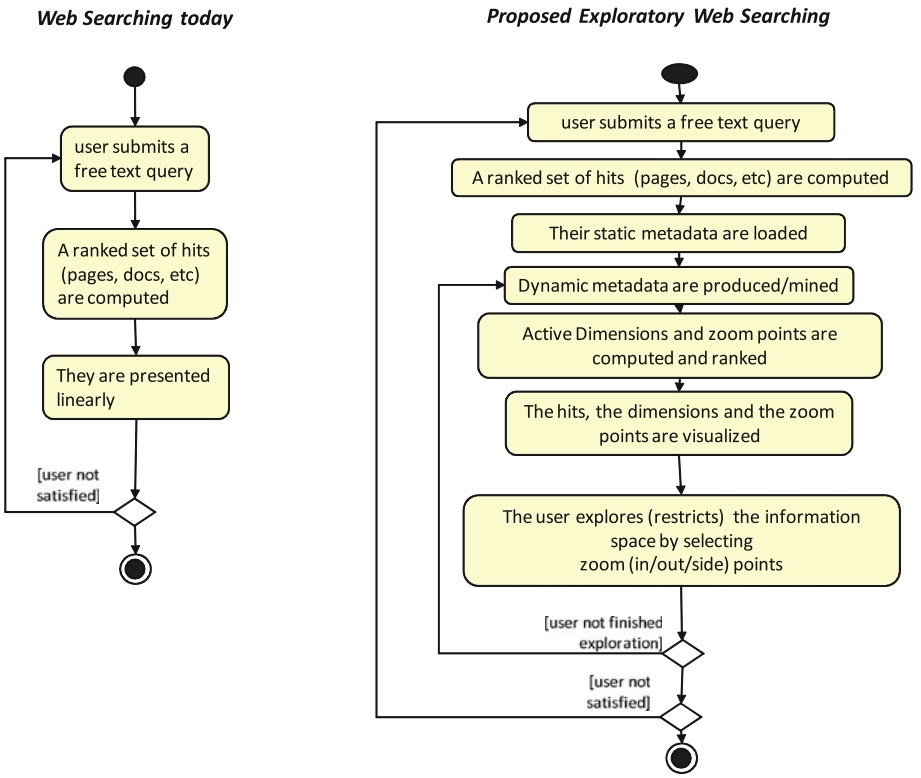

Fig. 1 Exploiting static and dynamically mined metadata for exploratory web searching

task accomplishment. This paper analyzes and extends the ideas and techniques that were originally presented in [35]. In comparison with that work, this paper:

- details the results clustering algorithm employed which is important for designing an incremental algorithm,
- elaborates on the costs of adopting pruned versus unpruned suffix trees and analyzes the performance gain of the incremental algorithm,
- reports detailed performance results, and
- reports the results of a user study that demonstrate the benefits of the proposed interaction in terms of user preference, satisfaction and degree of task accomplishment.

The rest of this paper is organized as follows. Section 2 discusses requirements, related work, and background information. Section 3 describes our approach for coupling results clustering with dynamic taxonomies. Section 4 describes implementation and reports performance results. Subsequently, Sect. 5 details the user study and discusses its results. Finally, Sect. 6 concludes the paper and identifies issues for further research.

2 Requirements and background

At first, we describe results clustering (Sect. 2.1), then exploratory search through FDT (Sect. 2.2), and finally other related works (Sect. 2.3).

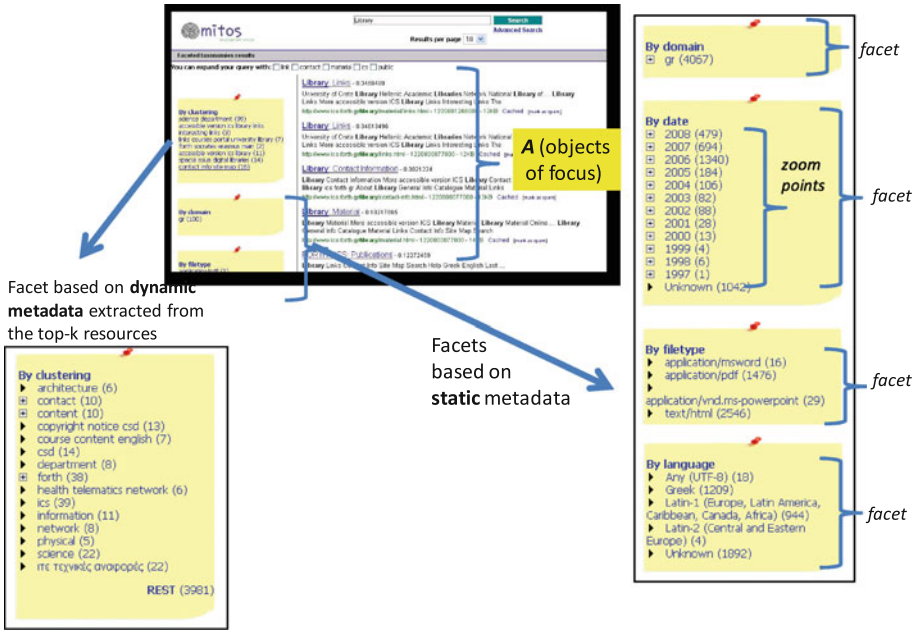

Fig. 2 `Mitos` GUI for exploratory web searching

2.1 Results clustering

Results clustering algorithms should satisfy several requirements. First of all, the generated clusters should be characterized from high intra-cluster similarity. Moreover, results clustering algorithms should be efficient and scalable since clustering is an online task and the size of the retrieved document set can vary. Usually, only the top-C documents are clustered in order to increase performance. In addition, the presentation of each cluster should be concise and accurate to allow users to detect what they need quickly. *Cluster labeling* is the task of deriving readable and meaningful (single-word or multiple-word) names for clusters, in order to help the user to recognize the clusters/topics he is interested in. Such labels must be predictive, descriptive, concise, and syntactically correct. Finally, it should be possible to provide high-quality clusters based on small document snippets rather than the whole documents.

In general, clustering can be applied either to the original documents, like in [7,14] or to their (query-dependent) snippets, as in [12,13,40,42,49,50]. Clustering meta-search engines (e.g., clusty.com) use the results of one or more WSEs, in order to increase coverage/relevance. Therefore, meta-search engines have direct access only to the snippets returned by the queried WSEs. Clustering the snippets rather than the whole documents makes clustering algorithms faster. Some clustering algorithms [8,12,19,46] use internal or external sources of knowledge like Web directories (e.g., DMoz[1]), Web dictionaries (e.g., WordNet), and thesauri, online encyclopedias and other online knowledge bases. These external sources are exploited to identify significant words/phrases, which represent the contents of the retrieved documents or can be enriched, in order to optimize the clustering and improve the quality of cluster labels.

[1] www.dmoz.org.

One very efficient and effective approach is the *Suffix Tree Clustering (STC)* [49] where search results (mainly snippets) can be clustered fast (in linear time), incrementally, and each cluster is labeled with one phrase. One important remark is that experiments have shown that the quality of clustering obtained by considering only the query-dependent snippets is very close to the quality obtained if the full contents of the documents are available [49]. Overall and for the problem at hand, we consider important the requirements of *relevance, browsable summaries, overlap, snippet-tolerance, speed*, and *incrementality* as described in [49]. Several variations of STC have emerged recently, e.g., [6,17,42].

2.2 Exploratory search and information thinning

Most WSEs are appropriate for focalized search, i.e., they make the assumption that users can accurately describe their information need using a small sequence of terms. However, as several user studies have shown, this is not the case. A high percentage of search tasks are exploratory [1], the user does not know accurately his information need, the user provides 2–5 words, and focalized search very commonly leads to inadequate interactions and poor results. Unfortunately, available UIs do not aid the user in query formulation and do not provide any exploration services. The returned answers are simple ranked lists of results, with no organization.

We believe that modern WSEs should guide users in exploring the information space. *Faceted (multidimensional) Dynamic Taxonomies (FDT)* [38] is a general purpose interaction model for exploratory searching. It can be applied over information bases consisting of objects that are described with respect to one or more aspects (facets). The description of an object with respect to one facet consists of assigning to the object one or more terms from the taxonomy that corresponds to that facet. The main advantages of FDT are the following: (a) it provides an overview of the returned answer (active dimensions, active values, and count information), (b) it releases the user from the effort of formulating metadata-based queries for locating resources or for restricting his focus, since he only has to click on the, so-called, *zoom points*, usually displayed at the left bar of the GUI (which also make evident how many hits he will get), (c) it does not disappoint the user since clicks always lead to resources (no empty results ever), (d) it is session-based (in contrast to the state-less query response) paradigm, thus allowing the user to reach his target gradually (through the so-called information thinning process). This model can host and exploit various static metadata, and it is currently used with success in e-commerce.

An example of the idea assuming only one facet is shown in Fig. 3. Figure 3a shows a taxonomy and 8 indexed objects (1–8). Figure 3b shows the dynamic taxonomy if we restrict our focus to the objects {4,5,6}. Figure 3c shows the browsing structure that could be provided at the GUI layer and Fig. 3d sketches user interaction.

The user explores or navigates the information space by setting and changing his *focus*. The notion of focus can be *intensional* or *extensional*. Specifically, any set of terms, i.e., any conjunction of terms (or any boolean expression of terms) is a possible *focus*. For example, the initial focus can be the top term of a facet. However, the user can also start from an arbitrary set of objects, and this is the common case in the context of a WSE and our primary scenario. In that case, we can say that the focus is defined extensionally. Specifically, if A is the result of a free text query q, then the interaction is based on the restriction of the faceted taxonomy on A (Fig. 3b shows the restriction of a taxonomy on the objects {4,5,6}). At any point during the interaction, we compute and provide the user the immediate zoom-in/out/side points along with count information (as shown in Fig. 3d). When the user selects one of these points, then the selected term is added to the focus, and so on. Note that the user

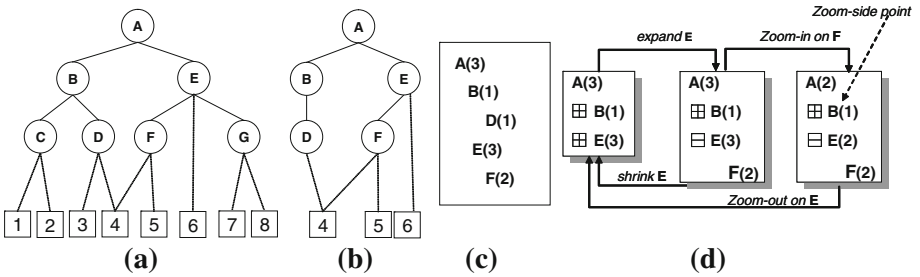

Fig. 3 Dynamic taxonomies

can exploit the faceted structure and at each step may decide to select a zoom point from a different facet (e.g., filetype, modification date, language, and web domain).

Examples of applications of faceted metadata search include e-commerce (e.g., ebay), library, and bibliographic portals (e.g., DBLP), museum portals e.g., [16] and Europeana,[2] mobile phone browsers e.g., [21], specialized search engines and portals e.g., [31], Semantic Web e.g., [15,32], general purpose WSEs e.g., Google Base, and other frameworks e.g., mSpace [39].

For reasons of self-containedness, the upper part of Table 1 defines formally and introduces notations for *terms*, *terminologies*, *taxonomies*, *faceted taxonomies*, *interpretations*, *descriptions* and *materialized faceted taxonomies* (the reader could skip this table and refer to it only when needed). The notion of restriction and zoom-in/out/side points are defined formally at the bottom part of that table.

2.3 Related works

Systems like those described in [4,10,15,27,32,47] support multiple facets, each associated with a taxonomy/ontology, which can be predefined. Moreover, the systems described in the study by [15,32,47] support ways to configure the taxonomies that are available during browsing based on the contents of the results. Specifically, [47] enriches the values of the object descriptions with more broad terms by exploiting WordNet, [15] supports rules for deciding which facets should be used according to the type of data objects based on RDF, and [32] supports reclassification of the objects to predefined types. However, none of these systems applies content-based results clustering, nor reconstructs the cluster tree taxonomy, while the user explores the answer set. Instead, they construct it once per each submitted query.

A somehow related interaction paradigm that involves clustering is Scatter/Gather [7,14]. This paradigm allows the user to select clusters. Subsequently, the documents of the selected clusters are clustered again, and the new clusters are presented and so on. This process can be repeated until individual documents are reached. However, for very big answer sets, the initial clusters apart from being very expensive to compute online will also be quite ambiguous and thus not very helpful for the user. Our approach alleviates this problem, since the user can restrict his focus through the available metadata, to a size that allows deriving more specific and informative cluster labels. There are a few works in the literature, e.g., [28], which compare automatic results clustering with guided exploration (through FDT). In this work, we propose combining these two approaches.

[2] http://www.europeana.eu.

Table 1 Faceted information sources and FDT interaction

Name	Notation	Definition
Materialized faceted taxonomies		
Terminology	T	A set of *terms* (can capture categorical/numeric values)
Subsumption	$\leq$	A partial order (reflexive, transitive and antisymmetric)
Taxonomy	$(T, \leq)$	T is a terminology, $\leq$ a subsumption relation over T
Broaders of t	$B^+(t)$	$\{ t' \mid t < t' \}$
Narrowers of t	$N^+(t)$	$\{ t' \mid t' < t \}$
Direct broaders of t	$B(t)$	$minimal_<\left(B^+(t)\right)$
Direct narr. of t	$N(t)$	$maximal_<\left(N^+(t)\right)$
Top element	$\top_i$	$\top_i = maximal_{\leq}(T_i)$
Faceted taxonomy	$\mathcal{F} = \{F_1, \ldots, F_k\}$	$F_i = (T_i, \leq_i)$, for $i = 1, \ldots, k$ and all T_i are disjoint
Object domain	Obj	Any denumerable set of objects
Interpretation of T	I	Any function $I : T \rightarrow 2^{Obj}$
Materialized faceted taxonomy	$(\mathcal{F}, I)$	$\mathcal{F}$ is a faceted taxonomy $\{F_1, \ldots, F_k\}$ and I is an interpretation of $T = \bigcup_{i=1,k} T_i$
Ordering of two interpretations	$I \sqsubseteq I'$	$I(t) \subseteq I'(t)$ for each $t \in T$
Model of $(T, \leq)$ induced by I	$\bar{I}$	$\bar{I}(t) = \cup \{ I(t') \mid t' \leq t \}$
Descr. of o w.r.t. I	$D_I(o)$	$D_I(o) = \{ t \in T \mid o \in I(t) \}$
Descr. of o w.r.t. $\bar{I}$	$D_{\bar{I}}(o) \equiv \bar{D}_I(o)$	$\{ t \in T \mid o \in \bar{I}(t) \} = \cup_{t \in D_I(o)} \left(\{t\} \cup B^+(t)\right)$
FDT-interaction: basic notions and notations		
Focus	ctx	Any subset of T such that $ctx = minimal(ctx)$
Projection on f. i	ctx_i	$ctx \cap T_i$
Kinds of zoom points w.r.t. a facet i while being at ctx		
Zoom points	$AZ_i(ctx)$	$\{ t \in T_i \mid \bar{I}(ctx) \cap \bar{I}(t) \neq \emptyset \}$
Zoom-in points	$Z_i^+(ctx)$	$AZ_i(ctx) \cap N^+(ctx_i)$
Immediate zoom-in points	$Z_i(ctx)$	$maximal(Z_i^+(ctx)) = AZ_i(ctx) \cap N(ctx_i)$
Zoom-side points	$ZR_i^+(ctx)$	$AZ_i(ctx) \setminus \left(ctx_i \cup N^+(ctx_i) \cup B^+(ctx_i)\right)$
Immediate zoom-side points	$ZR_i(ctx)$	$maximal\left(ZR^+(ctx)\right)$
Restriction over an object set $A \subseteq Obj$		
Reduced interpretation	I_A	$I_A(t) = I(t) \cap A$
Reduced terminology	T_A	$\{ t \in T \mid \bar{I}_A(t) \neq \emptyset \} = \{ t \in T \mid \bar{I}(t) \cap A \neq \emptyset \} = \cup_{o \in A} B^+(D_I(o))$

3 Coupling static and dynamically mined metadata for exploration

Here, we describe the coupling of static and dynamically mined metadata. In our discussion, we assume that we are in the context of a WSE called `Mitos` [34], which is a prototype WSE,[3] and we have at our disposal a main memory API (Application Programmatic Interface)

[3] Under development by the Department of Computer Science of the University of Crete and FORTH-ICS. URL: http://groogle.csd.uoc.gr:8080/mitos/.

Table 2 Table of symbols

Symbol	Meaning
q	Current query
$Ans(q)$	Answer of the query as returned by the WSE
C	Number of top elements of the answer that will be clustered (usually $C < 500$)
K	Number of top elements of the answer that will be loaded to `FleXplorer` (usually $K < 10^4$)
A_f	Top-K elements of $Ans(q)$
A_c	Top-C elements of $Ans(q)$
c_i	Cluster c_i
$Ext(c_i)$	Documents that belong to cluster c_i

called `FleXplorer`[4] for FDT-based interaction. `FleXplorer` allows managing (creating, deleting, modifying) terms, taxonomies, facets, and object descriptions. It supports both finite and infinite terminologies (e.g., numerically valued attributes) as well as explicitly and intensionally defined taxonomies. The former can be classification schemes and thesauri, the latter can be hierarchically organized intervals (based on the *inclusion* relation). Regarding user interaction, `FleXplorer` provides methods for setting the focus and getting the applicable zoom points, i.e., it provides the functionality described at the bottom part of Table 1.

Fast FDT-based interaction is feasible for hundreds of thousands of objects (as it will be described in Sect. 4.1.2). However, the application of results clustering on thousands of snippets would have the following shortcomings: (a) *Inefficiency*, since real-time results clustering is feasible only for hundreds of snippets, and (b) *Low cluster label quality*, since the resulting labels would be too general. To this end, we propose a *dynamic (on demand) integration* approach. The idea is to *apply the result clustering algorithm only on the top-C (usually $C < 500$) snippets of the current focus and to redo this whenever the focus changes.* This approach not only can be performed fast, but it is expected to return more informative cluster labels. Let q be the user query and $Ans(q)$ be the answer of this query. We shall use A_f to denote top-K (usually $K < 10^4$) objects of $Ans(q)$. We use the notation A_f, and not A_K, because the current answer could have less than K elements (i.e., $|Ans(q)| < K$). This means that $|A_f| = min(K, |Ans(q)|)$. We shall use A_c to denote top-C objects of $Ans(q)$. Clearly, $A_c \subseteq A_f \subseteq Ans(q)$. Table 2 lists all symbols that are used in the sequel.

The steps of the proposed process are the following:

Algorithm 1 Coupling Static and Dynamically mined Metadata

(1) The snippets of the elements of A_c are generated.
(2) Clustering is applied on the snippets of the elements of A_c, generating a cluster label tree clt.
(3) The elements in the set A_f (with their metadata), as well as clt, are loaded to `FleXplorer`. As the facet that corresponds to automatic clustering includes only the elements of A_c, we create an additional artificial cluster label, named "REST" where we place all objects in $A_f \setminus A_c$ (i.e., it will contain at most $K - C$ objects).
(4) `FleXplorer` computes and delivers to the GUI, the (immediate) zoom points.

[4] http://www.ics.forth.gr/~tzitzik/flexplorer.

The user can start browsing by selecting the desired zoom point(s), refining in this way the answer set. When a zoom point is selected, the new focus is computed and steps (1–4) are performed again over the new A_f (and A_c). A more efficient approach is possible. Specifically, Sect. 3.2 will introduce an incremental approach after first explaining, at Sect. 3.1, how results clustering is performed.

3.1 Results clustering: NM-STC

Regarding results clustering, we adopt NM-STC, a novel extension of STC [49] that we have devised. This algorithm is described in detail at [23]. Since the cluster labels returned by that algorithm are frequently occurring subphrases of phrases that occur in the snippets and titles of the returned pages, they are by construction syntactically correct. As in STC, NM-STC begins by constructing the suffix tree of the titles and snippets and then it scores each node of that tree. However, NM-STC uses a scoring formula that favors the occurrences in titles, does not merge base clusters, and returns an hierarchically organized set of cluster labels. For example, assume a is the snippet "a e. c a e." where each latin character denotes a word. The sentence suffixes of a is the set $P(a)$={"a e", "e", "c a e"} from which a suffix tree is constructed as shown in Fig. 4. [5] The snippets of all top-C elements are inserted to the suffix tree in this way (an example with three snippets is shown in Fig. 8A1). Subsequently, NM-STC scores all labels of the suffix tree. For reasons of self-containedness, we describe scoring in brief. Each element a of A has a title (denoted by $a.t$) and a snippet (denoted by $a.s$). NM-STC uses the following scoring formula:

$$
\begin{aligned}
score(p) &= Fr_t(p)\,|A|\,N + Fr_s(p)N + \text{PIDF}(p) \text{ where} \\
Fr_t(p) &= |\{a \in A \mid p \in a.t\}| \\
Fr_s(p) &= |\{a \in A \mid p \in a.s\}|
\end{aligned}
$$

PIDF stands for phrase IDF and N is the total number of indexed documents ($N = |Obj|$). If p is a single word, then PIDF(p) is the IDF of p $\left(\text{i.e., IDF}(w) = \frac{N}{|\{d \in Obj \mid w \in d\}|}\right)$. If p is a phrase consisting of the words $\{w_1, \ldots, w_m\}$, then PIDF is the average IDF of its words, i.e.,

$$\text{PIDF}(p) = \frac{1}{m} \sum_{i=1}^{m} \text{IDF}(w_i)$$

or alternatively $\text{PIDF}(p) = \max_{w \in p}(\text{IDF}(w))$. In our implementation and experiments, we used the average IDF. The IDF can be computed based on the entire collection if we are in the context of a single WSE. In our case, the index of `Mitos` stores only the stems of the words so IDF(w) is estimated over the stemmed words. If we are in the context of a MWSE (Meta WSE), then IDF could be based on external sources or on the current answer.[6] Synopsizing, the scoring function of NM-STC yields an order that resembles the lexicographic ordering where the number of occurrences in titles is the most significant criterion, followed by occurrences in snippets, followed by PIDF. Since PIDF has the lowest priority, it is used just for breaking some ties. From our experiments, the number of broken ties is low, so it does not affect significantly the results.

[5] For speeding up scoring we adopt a suffix tree where each node indexes all objects of it's narrower nodes.

[6] $\text{IDF}(w) = \frac{|A|}{|\{d \in A \mid w \in d\}|}$.

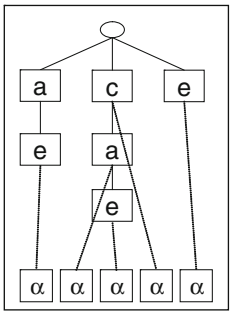

Fig. 4 Suffix tree example

If $NC_{\max}$ denotes the desired number of clusters, the algorithm returns the smaller set of the top-scored phrases whose maximal elements equals $NC_{\max}$. NM-STC, like STC, results in overlapping clusters.

In brief, the advantages of NM-STC are as follows: (a) the user never gets unexpected results, as opposed to the existing STC-based algorithms that adopt overlap-based cluster merging, (b) it is more configurable w.r.t. desired cluster label sizes (STC favors specific lengths), (c) it derives hierarchically organized labels, and (d) it favors occurrences in titles. The experimental evaluation showed that this algorithm is more preferred by users[7], and it is around two times faster that the plain STC (see Sect. 4.1.2). Figure 5 shows some indicative examples of hierarchically organized labels as produced by the algorithm NM-STC when applied upon the results returned from Google at 02/07/2009. Each frame of that figure corresponds to the query shown at the top of each hierarchy. For example, the first frame shows the cluster labels computed for the answer of the query *elephants*.[8]

Although we do not report here all the details of NM-STC, since the interested reader can refer to [23], it is worth noting that the hierarchy of cluster labels by NM-STC can be considered as a subsumption relation (i.e., a taxonomy) since it satisfies the following constraint: $c < c' \implies Ext(c) \subseteq Ext(c')$, i.e., if c is a cluster label that is direct or indirect child of c', then the extension of c (i.e., the set of pages classified under the cluster label c) is subset of the extension of c'. This property allows the incorporation of clustering as an additional facet of the FDT interaction.

3.2 Incremental algorithm for exploration

Here, we present an incremental approach for exploiting past computations and results. Specifically, the algorithm aims at reusing the extracted snippets and their suffixes since this is the more expensive task of online results clustering. From that perspective, the suffix tree of the snippets sf can be considered as a *cache*. The gained speedup is beneficial mainly for a standalone WSE that has to extract the query-dependent snippets from the cached textual contents of the indexed pages. In a Meta WSE, this cost is less since the underlying WSE payed the cost for extracting the snippets, and the Meta WSE just has to parse the results pages (and compute and insert their suffixes).

Let A_f be the objects of the current focus. If the user selects a zoom point, then he moves to a different focus. With A'_f, we denote the top-K elements of the new focus, and with A'_c, the top-C elements of the new focus. The steps of the incremental algorithm are given in Algorithm 2.

[7] http://groogle.csd.uoc.gr:8080/mitos/files/clusteringEvaluation/userStudy.html.

[8] The interested user could use the meta-search engine `StellaSearch` available at: http://groogle.csd.uoc.gr:8080/stellaSearch.

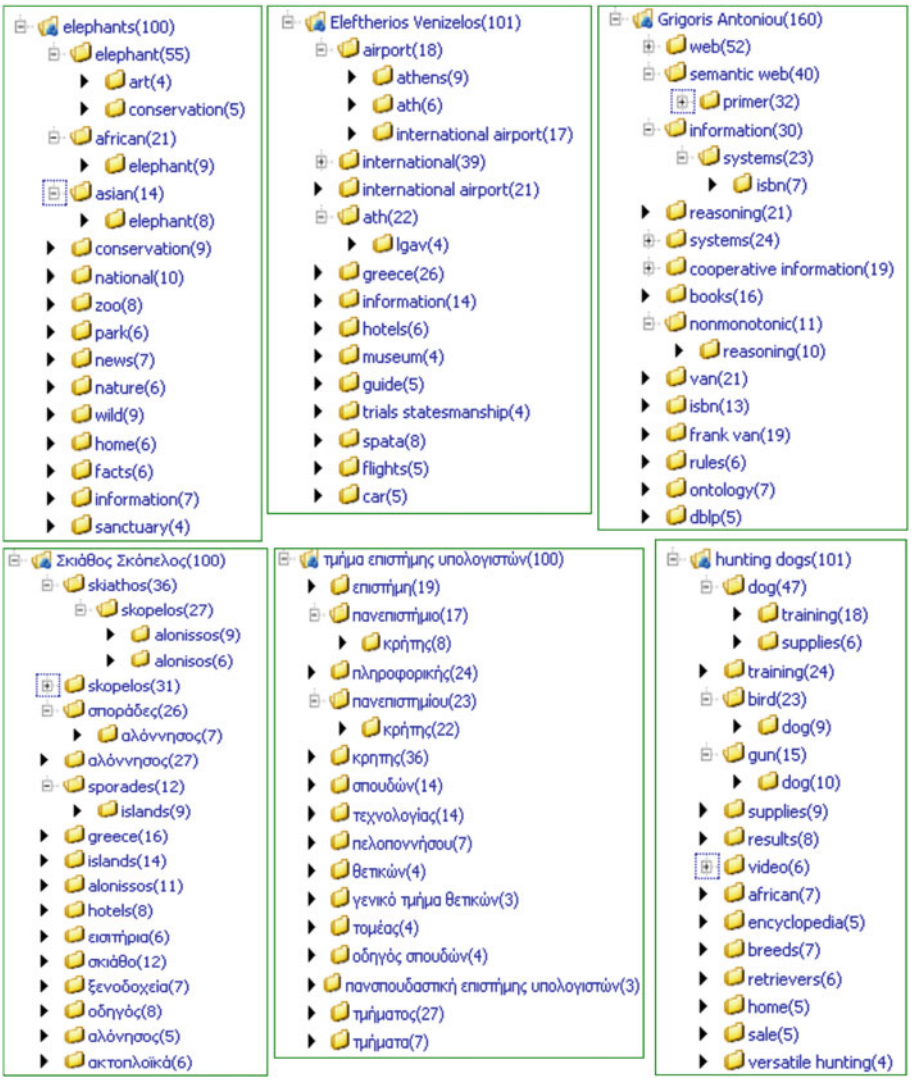

Fig. 5 Examples of applying NM-STC over the snippets returned by Google

Algorithm 2 *Incremental* Coupling of Static and Dyn.-mined Metadata

(1) We set $A_{c,new} = A'_c \setminus A_c$ and $A_{c,old} = A_c \setminus A'_c$, i.e., $A_{c,new}$ is the set of the new objects that have to be clustered, and $A_{c,old}$ is the set of objects that should no longer affect clustering.

(2) The snippets of the objects in $A_{c,new}$ are generated (those of $A_{c,old}$ are available from the previous step). Recall that snippet generation is expensive.

(3) NM-STC is applied *incrementally* to $A_{c,new}$.

(4) The new cluster label tree clt' is loaded to `FleXplorer`.

(5) `FleXplorer` computes and delivers to the GUI the (immediate) zoom points for the focus with contents A'_f.

Step (3) attempts an incremental application of NM-STC. Incremental means that the previous suffix tree sf is preserved. Specifically, we extend sf with the suffixes of the elements in the titles/snippets of the elements in $A_{c,new}$, exploiting the incremental nature of STC. With sf', we denote the extended suffix tree. To derive the top-scored labels, we have to score again all nodes of the suffix tree. However, we should not take into account objects that belong to $A_{c,old}$. Specifically, scoring should be based on the extension of the labels that contain elements of A'_c only. In brief, the suffix tree is incrementally updated while the user browses the information space by selecting zoom points and has to be constructed from scratch whenever the user submits a new query. We should also mention that the incremental NM-STC omits PIDF from its scoring formula (so $score(p) = Fr_t(p)|A| + Fr_s(p)$), in order to avoid keeping in main memory the IDFs of the words occurring in the elements of A_c.

3.2.1 On the size of snippet suffixes

Let $A_c = \{a_1, \ldots, a_c\}$ be the current focus. Each element a_i of A_c has a title (denoted by $a_i.t$) and a snippet (denoted by $a_i.s$). Each snippet can contain up to 2 sentences separated with the ellipsis punctuation symbol ("…") and each sentence can contain up to 11 words. We will denote these sentences with $a_i.s_1$ and $a_i.s_2$. If d is a text, then we shall use $P(d)$ to denote all *phrases* of d that are *sentence suffixes*, i.e., start from a word beginning and stop at the end of a sentence of d. For example, P("this is a test") = {"this is a test", "is a test", "a test", "test"}, while P("this is. A test") = {"this is", "is", "A test", "test"}. If d consists of k words then there are k suffixes (one starting from each word occurrence of d). The number of word occurrences in all these suffixes is at most equal to $k + (k-1) + \ldots 1 = \frac{(1+k)k}{2}$. We get this value if d contains only one sentence. If it contains more, i.e., if d contains one or more "." symbols, then the number of word occurrences is less.

We shall use $W(A_c)$ to denote the set of distinct words that appear in A_c and $P(A_c)$ to denote all phrases of the elements of A_c, i.e., $P(A_c) = \bigcup_{a \in A_c}(P(a.t) \cup P(a.s))$. The *number of sentence suffixes* of the elements of A_c is $|P(A_c)|$. We now define the $size$ of $P(A_c)$ as the number of word occurrences in the sentence suffixes in $P(A_c)$:

$$size(P(A_c)) = \sum_{a \in A_c} \sum_{p \in \{a.t,\ a.s_1,\ a.s_2\}} \frac{(1+|p|)|p|}{2}$$

For example, if the set of title/snippets of the elements of A_c is {{a b c},{a b. e}}, then $P(A_c)$={"a b c", "b c", "c", "a b", "b", "e"}, $|P(A_c)| = 6$, and $size(P(A_c)) = 10$.

In our testbed, the average title size is 6 words, and very frequently each part of the snippet ($a.s1$, $a.s2$) has a dot in the middle meaning that each part of the snippet consists of 2 sentences, with 5 and 6 words, respectively. In addition, in the scenario which is described at Sect. 4.1.3, we observed that for almost half of the results, the returned snippet consisted of one part ($a.s1$). Using the above assumptions, we can make the following estimation regarding the size of $P(A_C)$:

$$size(P(A_c)) \sim |A_c| * \left(\left(\frac{7*6}{2} + \frac{6*5}{2}\right) * 1.5 + \frac{7*6}{2}\right) = |A_c| * 75$$

3.2.2 Pruned versus unpruned suffix trees

One design choice at this point concerns the kind of the preserved suffix tree that can be either the *initial (unpruned) suffix tree* or the *pruned suffix tree*. Below, we describe and clarify the pros and cons of each choice.

In an unpruned tree, each node corresponds to a single word. A pruned tree is more compact in the sense that it contains less nodes. Specifically, if a node n (with label $n.lab$) contains only one child node n' (with label $n'.lab$) and both nodes contain the same objects, i.e., $Ext(n) = Ext(n')$, then they are collapsed to one single node, say n'', that has as label the space separated concatenation of the labels of the constituent nodes, i.e., $n.lab_n'.lab$. Scoring is done over the pruned suffix tree. However, to add/delete objects to/from a pruned suffix tree, sometimes requires "splitting" nodes (in additions) and pruning extra nodes (in deletions). On the other hand, if the unpruned suffix tree is preserved, then additions and deletions are performed right away and pruning takes place at the end. Below, we elaborate on their sizes and the insertion/modification costs.

If T is a suffix tree, we will denote by $|T|$ the number of nodes of T, by T_u the unpruned suffix tree of $P(A_c)$ and by T_p the pruned suffix tree of $P(A_c)$. It is not hard to see that it always holds $|T_p| < |T_u|$. Let us now count the min and max number of nodes that T_u and T_p can have.

Regarding an unpruned suffix tree T_u, it is clear that $|T_u| \leq size(P(A_c))$, so $size(P(A_c))$ is the worst case (maximum number of nodes) and occurs when all phrases are different. An example is given at Fig. 6 where we have $size(P(A_c)) = |T_u| = 12$. The least number of nodes is 1 occurring when all phrases are the same and comprise the same word. If all phrases comprise one word and there are $W(A_c)$ different words, then the least number of nodes is $W(A_c)$. So, we can write $W(A_c) \leq |T_u| \leq size(P(A_c))$.

Now, we will elaborate on pruned suffix trees. The worst case (maximum number of nodes) occurs when all nodes have different extensions. In that case, pruning cannot be applied, and therefore, the worst case is the same with that of unpruned trees. In the best case (minimum number of nodes), we have $|T_p| = 1$ (as in the unpruned case). If, however, all snippets have length 11, then the best case occurs when all snippets are the same, and in that case, the minimum number of nodes is 11 (as Fig. 7 shows for the case where $|p| = 3$).

Below follows a discussion regarding the *insertion cost*. If s is a suffix, we will denote by $|s|$ the number of words in s. If T is unpruned, then the insertion (addition) of the suffix s to t takes O($|s|$) time as each node of the suffix tree contains a HashMap for its children nodes (the key of the HashMap is a word and its value is the pointer to the corresponding node). If T is pruned, then the insertion of s in T needs more time since in the worst case all

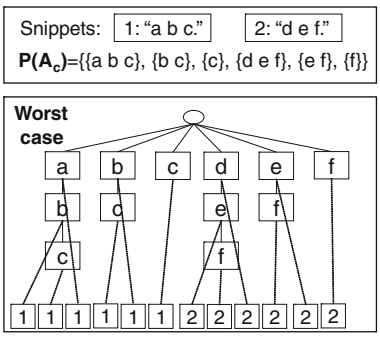

Fig. 6 Unpruned suffix trees: max number of nodes

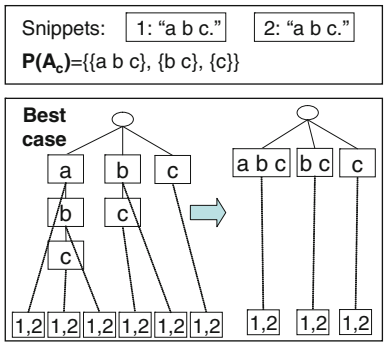

Fig. 7 Pruned suffix trees: min number of nodes

the nodes of the tree have to be checked. In particular, to insert s, for each node n of T, we have to check whether s is prefix of n's label. Furthermore, if a prefix of s is also prefix of a node's label, then we have to split that node. It follows that in an unpruned tree, the cost for inserting all suffixes of a phrase s is analogous to the number of words of these suffixes, so this cost is analogous to $\frac{(1+|s|)|s|}{2}$. In a pruned tree, this cost is analogous to $\frac{(1+|s|)|s|}{2}|T_p|$.

Although scoring is done over the pruned tree, an unpruned tree is beneficial for the incremental algorithm due to the insertions, and for this reason, our implementation follows this policy. Independently of the kind of the preserved suffix tree, below we discuss two possible approaches for *updating* the suffix tree:

- *Scan*-approach. We scan the nodes of the suffix tree sf' and delete from their extensions all elements that belong to $A_{c,old}$. Figure 8 illustrates an example where we have three snippets (1, 2, and 3). The words of the snippets are depicted by latin characters ('a'–'g'). Suppose that $A_c = \{1, 2\}$ and that $A'_c = \{1, 3\}$. Figure 8A1 shows the initial suffix tree of A_c, which is kept in main memory. Figure 8A2 shows the suffix tree after inserting snippet 3 (which belongs to A'_c), while Fig. 8A3 shows the suffix tree after deleting snippet 2 (which belongs to A_c but not to A'_c), and this is actually the suffix tree of A'_c. Finally, Fig. 8A4 shows the pruned suffix tree over which scoring is reapplied.
- *Object-to-ClusterLabel Index*-approach. An alternative approach is to have an additional data structure that for each object o in A_c it keeps pointers to the nodes of the suffix tree to whose extension o belongs. In that case, we do not have to scan the entire suffix tree since we can directly go to the nodes whose extension has to be reduced. The extra memory space for this policy is roughly equal to the size of the suffix tree. However, the suffix tree construction process will be slower as we have to maintain the additional data structure too.

3.2.3 Incremental versus non-incremental algorithm

Below, we compare the performance of the incremental algorithm and the non-incremental algorithm, as a function of the overlap between A_c and A'_c. Suppose that $|A_c| = \left|A'_c\right| = C$, which is a reasonable assumption, since C is small in comparison with the size of the answers. Clearly $0 \leq \left|A_c \cap A'_c\right| \leq C$ and let $L = \left|A_c \cap A'_c\right|$.

The advantage of the incremental algorithm is that it does not have to extract L snippets (those corresponding to $A_c \cap A'_c$), and it does not have to insert them into the suffix tree sf. With t_{extr}, we denote the time required to extract a snippet and with t_{insert} the time required to insert the snippet (i.e., all sentence suffixes of that snippet) to the suffix tree. The cost to insert

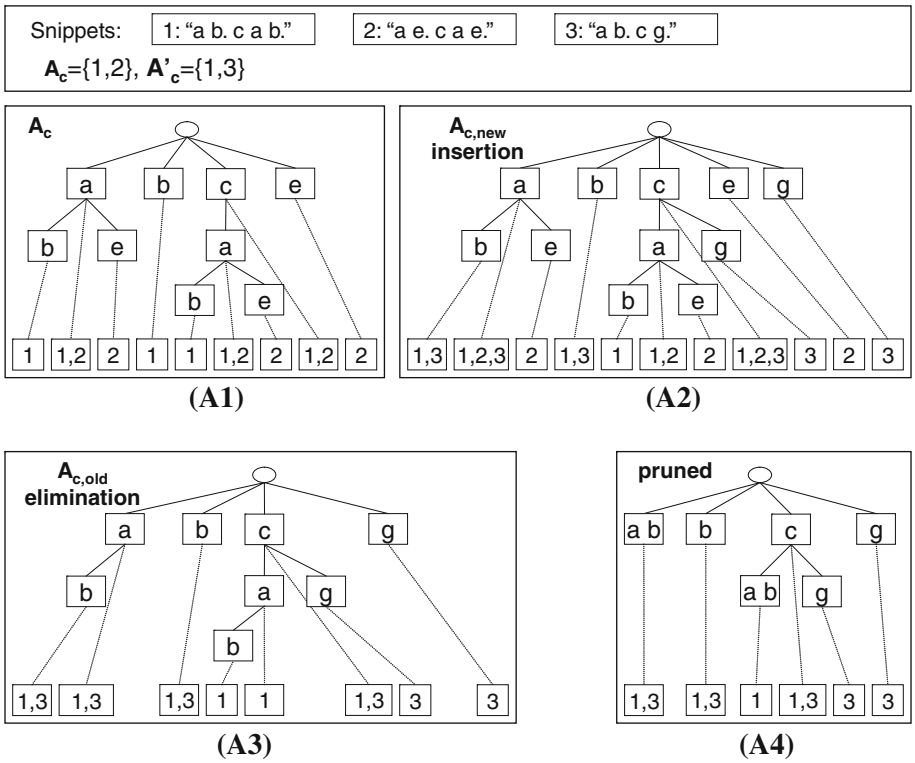

Fig. 8 Incremental evaluation

the suffixes that correspond to a phrase s is analogous to $\frac{(1+|s|)|s|}{2}$ in an unpruned suffix tree. Therefore, we consider that $t_{\text{insert}} = \frac{(1+|s|)|s|}{2} * F$ where F is a constant. It follows from the above that the incremental algorithm has $L(t_{\text{extr}} + t_{\text{insert}})$ less cost than the non-incremental. In our setting, using the assumptions discussed in Sect. 3.2.1, we can express this gain as $L(t_{\text{extr}} + 75 * F)$.

On the other hand, the overhead of the incremental algorithm (in comparison with the non-incremental) is that it has to scan the suffix tree sf and to mark the old elements (those in $A_{c,old} = A_c \setminus A'_c$). The cost of a scan is analogous to the size of the suffix tree $STsize$. Since the size of $A_{c,old}$ is $|A_{c,old}| = C - L$, the cost is analogous to $STsize * (C - L)$. We can express this cost by $STsize * (C - L) * G$ where G is a constant. As we discussed in the previous section, the size of the suffix tree, $STsize$, over a set of elements A is less than $|A| * 75$. In our case, A stands for the union of A_c and A'_c (i.e., $A = A_c \cup A'_c$), so $C = |A_c| \leq |A| \leq |A_c| + |A'_c| = 2C$. The lower (resp. upper) bound is reached when these two sets are the same (resp. disjoint). This means that $C \leq |A| \leq 2C$. More precisely, $|A| = 2C - L$. In our setting, it follows that $STsize = (2C - L) * 75$.

From the above, it follows that the incremental algorithm is faster than the non-incremental if the computational gain is greater than the computational loss, i.e., if the following inequality holds:

$$(t_{\text{extr}} + t_{\text{insert}})L > STsize(C - L)G$$
$$\Leftrightarrow (t_{\text{extr}} + 75F)L > 75G(2C - L)(C - L)$$

If we express L as a fraction of C, i.e., $L = x * C$, then the inequality becomes:

$$\begin{aligned}(t_{\text{extr}} + 75F)Cx &> 75G(2C - xC)(C - xC)\\ \Leftrightarrow (t_{\text{extr}} + 75F)x &> 75CG(2 - x)(1 - x)\\ &= 75CG(2 - x) - 75CG(2 - x)x\end{aligned}$$

If we set $H = 75G$, then the inequality becomes:

$$\begin{aligned}(t_{\text{extr}} + 75F)x &> HC(2 - x) - HCx(2 - x)\\ &= 2HC - 3HCx + HCx^2\end{aligned}$$

and we finally get:

$$-HCx^2 + (t_{\text{extr}} + 75F + 3HC)x - 2HC > 0 \tag{1}$$

Furthermore, since $L \leq C$ it should hold:

$$x \leq 1 \tag{2}$$

Synopsizing, we have a performance gain if the x (where $x = \frac{|A_c \cap A'_c|}{C}$), satisfies the inequalities (1) and (2).

We have carried out some experiments in our setting in order to compute average values for t_{extr}, G and F. To compute t_{extr}, we measured the average time of snippet extraction for a set of queries evaluated over the corpus of `Mitos`. The set of queries contained 2.4 words in average. Some results are shown in Table 3. It follows that $t_{\text{extr}} = 0.00793$ by considering the time for extracting the snippets of 100 docs and dividing by 100. We used scenarios like the one described at Sect. 4.1.3 to compute F and G. The average values obtained were $F = 3.60007798 \times 10^{-6}$ and $G = 1.76621753 \times 10^{-6}$, for $C = 100$. Using the above values, $H = 0.034966$ and inequality (1) becomes:

$$-0.01324 * x^2 + 0.04792 * x - 0.02648 > 0 \tag{3}$$

Inequality (3) crosses the x-axis at $x_1 = 0.680554$ and $x_2 = 2.938781$. Since, this is a "negative" quadratic inequality and by also considering inequality (2), we end up to:

$$0.68 \leq x \leq 1 \tag{4}$$

That means we should have at least $\simeq 68.0\%$ overlap to get a speedup in our system. In cases where t_{extr} times are bigger, we need smaller overlaps (e.g., for the same settings if $t_{\text{extr}} = 0.01$ then the overlap should be at least $\simeq 63.9\%$, and if $t_{\text{extr}} = 0.016$ then the overlap should be at least $\simeq 54.3\%$). We have to note that above analysis can be exploited also for deciding at run-time the more beneficial policy based on the x value of the action (zoom operation) at hand. More experiments for the overall performance gain in a real system are given at the subsequent section.

Table 3 Top-C snippet generation and clustering times (in seconds)

Measured task	100	200	300	400	500
Time to generate snippets	0.793	1.375	1.849	2.268	2.852
Time to apply STC	0.138	0.375	0.833	1.494	2.303
Time to apply NM-STC	0.117	0.189	0.311	0.449	0.648

4 Implementation and experimental evaluation

`FleXplorer` is used by `Mitos` for offering general purpose browsing and exploration services. Currently, on the basis of the top-K answer of each submitted query, the following five facets are created and offered to users:

- `By clustering`: the hierarchy of cluster labels derived by NM-STC
- `By domain`: the hierarchy of web domains (e.g., $csd.uoc.gr < uoc.gr < gr$)
- `By filetype`: the list of filetypes, e.g., *pdf, html, doc* (no hierarchy is created in this case)
- `By language`: the languages where the language of a document is based on the encoding of the web page
- `By date`: the (modification) dates of the pages structured hierarchically (year → month → day)

The metadata that correspond to *domains, filetypes, languages* and *dates* are already stored in `Mitos`, and `Mitos` loads them to `FleXplorer`. Figure 2 shows a screenshot of `Mitos` WSE. As the screen of a computer or a PDA may be too small to show all the facets (like the previous screenshot) or the user is not interested in a specific facet or prefers a specific ranking of the facets, we have developed a mechanism that allows the user to drag and drop the facets (so that to place them in the desired order) and open or close them. Figure 9 shows the modified version of the GUI of Fig. 2 where the user has opened only the facets *By Clustering* and *By Date* and prefers to show first the facet *By Date*. User's preferences are stored to the cache of the web browser as cookies to persist over time.

When the user interacts with the clustering facet (i.e., when he clicks on a cluster label), we do not apply the re-clustering process. This behavior is more intuitive, since it preserves the clustering hierarchy (while the user interacts with the clustering facet) and does not frustrate him with unexpected results. In case the user is not satisfied by the available cluster labels for the top-C objects of the answer, he can enforce the execution of the clustering algorithm for the next top-C by pressing the *REST* zoom-in point as it has already been mentioned (which keeps pointers to maximum $K - C$ objects).

4.1 Experimental results

4.1.1 Clustering performance

It is worth noting that the most time-consuming subtask is not the clustering algorithm but the extraction of the snippets from the cached copies of textual contents of the pages that should feed the clustering algorithm. To measure the performance of the clustering algorithm and the snippet generation, we selected 16 queries and we counted the average times to generate and cluster the top-{100, 200, 300, 400, 500} snippets. All measurements were taken using a Pentium IV 3.4 GHz, with 2 GB RAM, running Linux Debian.

Table 3 shows snippet generation times and the clustering algorithms performance (measured in seconds). Notice that snippet generation is a slow operation and is the bottleneck in order to provide fast on demand clustering, for a big top-C number ($C > 500$). We should mention though that our testbed includes a rather big number of large-sized files (i.e., pdf, ppt), which hurt snippet generation times. Moreover, notice that NM-STC is faster than STC (at least two times faster than STC for $C \geq 200$). This is because NM-STC does not have to intersect and merge base clusters.

Fig. 9 Modified faceted exploration UI according to user's preferences

4.1.2 Dynamic taxonomies performance

Loading times of `FleXplorer` have been thoroughly measured in [41], and related work also includes [4,37,48]. In brief, the computation of zoom points with count information is more expensive than without. In 1 sec, we can compute the zoom-in points of 240,000 results with count information, while without count information, we can compute the zoom-in points of 540,000 results.

4.1.3 Performance of the incremental versus the non-incremental algorithm

In this experiment, we measured the overall cost for cluster generation (snippet generation and clustering algorithm execution times) and dynamic taxonomies (to compute the zoom points and to load the new clustering labels to the corresponding facet). Moreover, we compare the non-incremental with the incremental algorithm, which preserves the initial suffix tree and the elimination of old objects is done using the Scan approach. The scenario includes: (a) the execution of the query *crete* which returns 4,067 results, (b) the expansion of the *gr* zoom point of the *By domain* facet and the selection of the *uoc.gr (1277)* zoom-in point from the hierarchy revealed from the expansion, and finally, (c) the selection of the *text/html (807)* zoom-in point of the *By filetype* facet. Let c_a, c_b and c_c be snippets of the $top - C$ elements in the steps (a, b) and (c), respectively. Figure 10 shows the facet terms after steps (a, b), and (c), as they are displayed in the left bar of the WSE GUI. We set $K = 10^4$ (i.e., the whole answer set is loaded) and repeated the above steps for the following values of C: 100, 200,…,500.

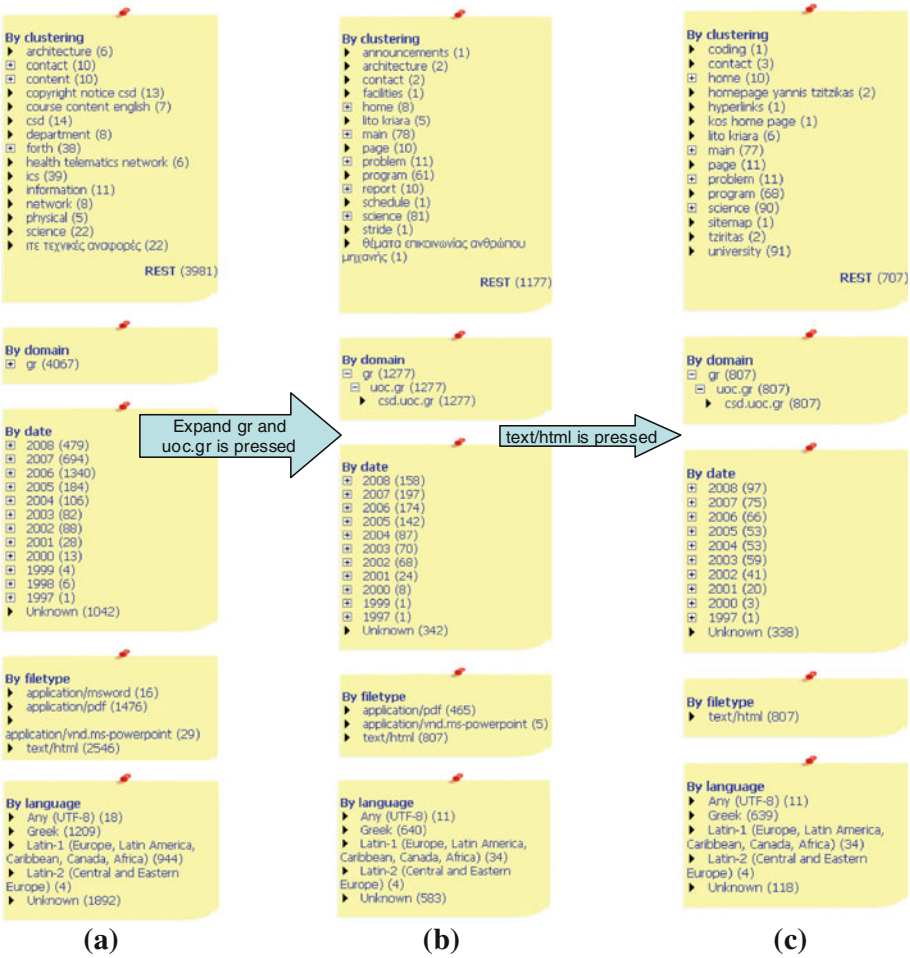

Fig. 10 Steps **a–c** of running scenario

We do not measure the cost of the query evaluation time. In all experiments, the displayed zoom points are accompanied by count information.

Table 4 shows the intersection of A_c and A'_c for steps (a, b) and (c) and the execution times that correspond to the integration of `FleXplorer` and results clustering using the *non-incremental* and the *incremental approach* of NM-STC, for the top-C elements. Specifically, FDT times include the time to compute the zoom points (for each step), the time to load to `FleXplorer` the metadata of the top-K hits (step a), and the time to remove the old and load the new clustering facet (for steps b and c). It is evident that for top-100 and top-200 values, the results are presented to the user pretty fast (especially for steps b and c), making the proposed on demand clustering method suitable as an online task. Moreover, we can see that there is a linear correlation between the time cost and the value C. Finally, the time required for loading the clusters for the top-500 documents, is less than 4.5 sec making even big top-C a feasible configuration.

Table 4 Top-C integration timings for non-incremental and incremental algorithms (in seconds)

	Step (a)				Step (b)				Step (c)			
	Total	FDT	Clust.	Snip.	Total	FDT	Clust.	Snip.	Total	FDT	Clust.	Snip.
top-100	$\lvert c_a\rvert = 100$				$\lvert c_a \cap c_b\rvert = 85$, overlap = 85%				$\lvert c_b \cap c_c\rvert = 22$, overlap = 22%			
Non-incr.	1.72	0.62	0.06	1.04	**1.07**	0.23	0.03	0.82	0.36	0.21	0.01	0.14
Incr.	1.75	0.62	0.07	1.06	**0.53**	0.24	0.04	0.25	0.42	0.20	0.09	0.12
top-200	$\lvert c_a\rvert = 200$				$\lvert c_a \cap c_b\rvert = 174$, overlap = 87%				$\lvert c_b \cap c_c\rvert = 24$, overlap = 12%			
Non-incr.	2.88	0.49	0.08	2.31	**1.95**	0.13	0.07	1.76	0.73	0.24	0.13	0.36
Incr.	2.93	0.50	0.11	2.32	**0.58**	0.13	0.08	0.37	0.66	0.20	0.18	0.28
top-300	$\lvert c_a\rvert = 300$				$\lvert c_a \cap c_b\rvert = 232$, overlap = 77.3%				$\lvert c_b \cap c_c\rvert = 78$, overlap = 26%			
Non-incr.	3.51	0.42	0.14	2.95	**2.60**	0.23	0.16	2.21	0.82	0.21	0.17	0.44
Incr.	3.69	0.5	0.22	2.97	**0.7**	0.12	0.38	0.2	0.85	0.21	0.31	0.33
top-400	$\lvert c_a\rvert = 400$				$\lvert c_a \cap c_b\rvert = 257$, overlap = 64.25%				$\lvert c_b \cap c_c\rvert = 170$, overlap = 42.5%			
Non-incr.	3.87	0.43	0.23	3.21	**3.12**	0.25	0.31	2.56	**2.16**	0.12	0.24	1.8
Incr.	4.04	0.5	0.34	3.2	**1.09**	0.24	0.57	0.28	**0.84**	0.11	0.45	0.28
top-500	$\lvert c_a\rvert = 500$				$\lvert c_a \cap c_b\rvert = 269$, overlap = 53.8%				$\lvert c_b \cap c_c\rvert = 268$, overlap = 53.6%			
Non-incr.	4.39	0.46	0.45	3.48	**3.14**	0.26	0.32	2.56	**2.71**	0.24	0.32	2.15
Incr.	4.56	0.45	0.63	3.48	**1.32**	0.14	0.8	0.38	**1.08**	0.23	0.56	0.29

Table 5 Text sizes to be processed for snippet generation (in MB)

	Step a	Step b		Step c	
		Non-incr.	Incr.	Non-incr.	Incr.
top-100	51.8	57.7	6.5	0.7	0.6
top-200	110.8	124	14.0	1.5	1.4
top-300	141.8	147	8.4	3.6	3.0
top-400	151.7	149.1	3.8	6.1	4.8
top-500	156.5	150.8	4.0	8.5	5.7

On the other hand, Table 5 reports the sizes of the documents that have to be processed to create the snippets for both the incremental and the non-incremental version of the clustering algorithm. We can see that there are huge differences, especially in step (b) (due to the large amount of overlapping pdf documents, which are big sized).

Comparing the incremental and the non-incremental algorithm, we observe a significant speedup whenever the overlap is more than around 40%, for our scenario. Notice that big overlaps tend to reduce the overhead of the incremental clustering algorithm, due to the smaller number of updates to the preserved suffix tree, and also tend to reduce the cost of snippet generation. The difference between 40% and 68% estimated in Sect. 3.2.3, is probably due to variances of the values of t_{extr}, F, G, C. At step (a), the suffix tree construction is the same for both algorithms as the suffix tree sf has to be constructed from scratch (small variations are due to random differences in system load). For step (b), the speedup of the incremental algorithm is significant due to the big overlaps and large differences in text sizes as reported in Table 5 (almost a 4X speedup in the case of top-300). Notice that speedups are

not linearly correlated with overlaps, due to different sizes in non-overlapping documents. Notice that although the overlap for top-200 is the bigger one, the speedup is not the highest, due to the bigger size of the processed documents. Another observation is that the cost of the incremental algorithm is especially high in the case of top-500, despite the fact that the overlap is around 53%. In that case, 235 new snippets have to be inserted to the original suffix tree. For step (c), we do not observe big speedups due to the small overlaps and small differences in text sizes as reported in Table 5. The only exceptions are the cases of top-400 and top-500, where we have big overlaps and measurable differences in text sizes.

In general, big overlaps are expected for information needs that require inspecting several results hits. Such information needs are usually called *exhaustive* [24] or *recall oriented*. For instance, patent search, bibliographic search (e.g., for writing a survey), hotel search, and car search, fall into this category.

Finally, we have to note that the incremental algorithm can speedup not only NM-STC but any suffix tree-based clustering algorithm, e.g., [6,17,49].

5 Evaluation of usability

At first, we should clarify that the traditional metrics for evaluating retrieval effectiveness (like *precision* and *recall*) are not appropriate for evaluating session-based interactive information retrieval (IIR). The evaluation of systems offering session-based IIR is difficult, because of the complexity of data relationships, diversity of displayed data, interactive nature of exploratory search, along with the perceptual and cognitive abilities offered. They rely heavily on users' ability to identify and act on exploration opportunities [44]. Regarding web search interfaces, [20] discusses various evaluation methods and measures, while [45] proposes an evaluation method for exploratory search features. [26] describes how to create exploratory tasks for faceted search interfaces and [29] discusses the interaction of searchers with a web-based faceted library catalog. *Usage-based* evaluation methods for interactive session-based information retrieval are discussed in [3,18], while [36] stresses the role of *user-centered* evaluations and [22] provides a survey of such approaches.

In our case, we conducted a *user* evaluation in order to *compare* the usefulness and usability of three interfaces:

(A) one offering *FDT* but without results clustering,
(B) one offering *results clustering* only, and
(C) one offering *both* FDT and results clustering.

We wanted to compare the above interfaces with respect to *user preference*, *user satisfaction*, and *task accomplishment*. Furthermore, we wanted to examine how users used these interfaces and for this reason we also conducted a *log analysis* (usage-based evaluation). Figure 11 depicts the steps of our evaluation process, which is detailed below.

5.1 Tasks and participants

We specified four information needs (or tasks) that are shown in Table 6. For the first three ($T1$–$T3$), we specified three variations of each, one for each interface (the subscripts indicate the interface that should be used in each variation). We decided to follow this policy because if we had the same task for each interface, then the order by which participants used the interfaces could bias the results (the users would remember the title of the pages that have the sought information). However, for the fourth task ($T4$), we did not provide a distinct

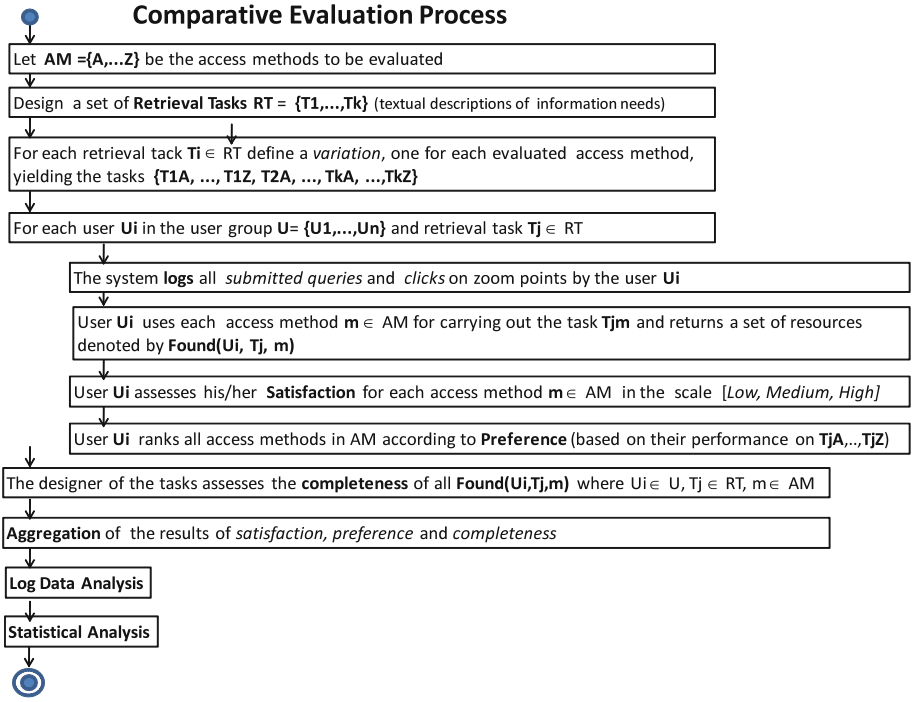

Fig. 11 Comparative evaluation process

variation for each interface but the participants had to return the number of results they found using each interface in a specific interval of time.

All tasks were refined using the task refinement steps described in [25]:

- The task descriptions included words that were values of a facet (or a facet contained a semantically close term).
- By using the words of the task description, the user should not be able to answer the task using the first 10 results of the answer (else the task would be too easy, not requiring exploratory search).
- The facets should be useful without having to click the "show more" link of a facet.

Thirteen users participated in the evaluation with ages ranging from 20 to 30, 61.5% men and 38.5% women. We can distinguish two groups: the *advanced* group consisting of 3 users and the *regular* one consisting of 10 users. The *advanced* users had prior experience in using clustering and multidimensional browsing services, while the *regular* ones had not. For this reason, and before starting the evaluation, we gave to each regular user a brief tutorial on using these services through examples over the `Mitos` WSE. The tutorial was personal (for each user individually) and lasted around 5 min. We admit that the number of participants is not high; however, it can be considered adequate for our purposes, since according to [11], 10 evaluators are enough for getting more than 82% of the usability problems of a user interface (at least in their experiments). On the other hand, we preferred to create more user tasks for evaluation, since according to [30], a bigger number of tasks can improve usability test performance.

Table 6 Evaluation tasks

Id	Information need/task description
$T1$	
$(T1_A)$	Find at least two papers of the head of the Information Systems Laboratory of FORTH-ICS that were published on 2007 and concern Semantic Web.
$(T1_B)$	Find at least two papers of the chairman of the Computer Science Department of University of Crete that were published on 2006 and concern e-learning.
$(T1_C)$	Find at least two papers of the Director of the ICS-FORTH that were published on 1997 and concern user interfaces.
$T2$	
$(T2_A)$	Find the personal web pages of at least two local collaborators of Dimitris Plexousakis.
$(T2_B)$	Find the personal web pages of at least two local collaborators of Grigoris Antoniou.
$(T2_C)$	Find the personal web pages of at least two local collaborators of Vasilis Christophides.
$T3$	
$(T3_A)$	Find two presentations about Wireless Networks in .ppt that are published at the FORTH domain (forth.gr).
$(T3_B)$	Find two presentations about Web Services in .ppt that are published at the FORTH domain (forth.gr).
$(T3_C)$	Find two presentations about Computer Vision in .ppt that are published at the FORTH domain (forth.gr).
$T4$	
$(T4)$	Find in 2 min all (or the most) persons of CSD who include into their home pages information about their music preferences.

For each information need ($T1$–$T4$), each user had to fill a form like the one shown at Table 7 where apart from the sought information (i.e., the requested URLs), the user had to rank the three interfaces according to preference by assigning each interface a distinct number from {1,2,3} (1 for the most preferred), and to express the degree of his satisfaction from each interface by using a value from the set {high, medium, low}. The latter question is not comparative in the sense that a user could assign "high" or "low" to all interfaces.

5.2 Evaluation results

Table 8 shows the results of the evaluation for all users per task, while Table 9 shows the aggregated results. For instance, from Table 8, we can see that all users had $high$ satisfaction from *FDT* in Task 3, while from Table 9, we can observe that 50% of the *advanced* users had *medium* satisfaction from *Clustering*. Below, we analyze each aspect of the evaluation.

5.2.1 Completeness

The last column of Table 9 shows the average percentage of the correct URLs that users found in task $T4$ in each user interface, out of the total number of correct URLs in our testbed (6 correct URLs). We observe that with (C) (combination of the *Clustering* with the *FDT*) *advanced* users achieved the highest degree of completeness i.e., 77.7%. *Regular* users achieved the highest degree of completeness using (A) i.e., 46.6% while (C) is quite close, i.e., 45%.

Table 7 User evaluation form

Information Need:	Response:	
Interface	User Satisfaction (Low/Medium/High)	Preference (1/2/3)
(A) FDT		
Information Need:	Response:	
Interface	User Satisfaction (Low/Medium/High)	Preference (1/2/3)
(B) Clustering		
Information Need:	Response:	
Interface	User Satisfaction (Low/Medium/High)	Preference (1/2/3)
(C) Both		

5.2.2 Log data analysis

During the evaluation by each user, we logged and counted: (a) the number of submitted queries and (b) the number of clicked zoom points (by facet).

Table 10 shows the average number of submitted queries and clicked zoom points by the users of the two groups for all tasks. For instance, a *regular* user submitted in average 13.2 queries in *Clustering* interface during the evaluation. At first, we observe that both groups submitted the least number of queries with interface (C). This is an evidence that the combination of interfaces makes information seeking less laborious. The difference is significant for the advanced users: they made more than 50% less queries with *Faceted Taxonomies* and with (C), than with *Clustering*. We can also see that a regular user pressed in average only 2.7 zoom points of the facet *By Clustering*, while an advanced user pressed 6 (we will attempt to explain this difference later on). Notice that 6 clicks on clustering zoom points are more than the sum of the clicks on the points of the rest facets.

5.2.3 User preference

To identify the most preferred interface (for regular and advanced users), we aggregated the preference rankings for each task using *Plurality Ranking* (i.e., we count the first positions), and *Borda ranking* [9] (i.e., we sum the positions of each interface in all rankings). We should note that preferences can give us more clear and safe results from satisfaction levels, because each user had to rank the different interfaces with a *distinct* preference number (from 1 to 3), while he could assign a "high" or "low" satisfaction level to all interfaces.

Table 11 shows the aggregated rankings, per user group, user task and interface. In a *Plurality* column, the higher a value is, the better (i.e., the more first positions it got), while in a *Borda* column the less a value is, the better. The rows marked with *All* show the sum of the values of all tasks. With bold, we have marked the best values for each task. Figure 12 shows the same results graphically.

Table 8 User satisfaction and preference percentages per interface and task

		Interface	User satisfaction			Preference		
			Low	Medium	High	1	2	3
Advanced	Task1	(A) FDT	**66.6**	33.3	–	33.3	–	**66.6**
		(B) Clustering	**33.3**	**33.3**	**33.3**	**33.3**	**33.3**	**33.3**
		(C) Both	**33.3**	**33.3**	**33.3**	33.3	**66.6**	–
	Task2	(A) FDT	–	**100**	–	–	33.3	**66.6**
		(B) Clustering	–	**66.6**	33.3	**33.3**	**33.3**	**33.3**
		(C) Both	–	**66.6**	**33.3**	**66.6**	33.3	–
	Task3	(A) FDT	–	–	**100**	33.3	**66.6**	–
		(B) Clustering	**66.6**	33.3	–	–	–	**100**
		(C) Both	–	–	**100**	**66.6**	33.3	–
	Task4	(A) FDT	33.3	–	**66.6**	–	**66.6**	33.3
		(B) Clustering	33.3	**66.6**	–	33.3	–	**66.6**
		(C) Both	33.3	–	**66.6**	**66.6**	33.3	–
Regular	Task1	(A) FDT	10	**60**	30	20	**40**	**40**
		(B) Clustering	20	**70**	10	20	**40**	**40**
		(C) Both	10	30	**60**	**60**	20	20
	Task2	(A) FDT	–	**50**	**50**	**40**	20	**40**
		(B) Clustering	20	**50**	30	40	–	**60**
		(C) Both	–	**60**	40	20	**80**	–
	Task3	(A) FDT	–	–	**100**	**70**	20	10
		(B) Clustering	**70**	30	–	–	10	**90**
		(C) Both	30	–	**70**	30	**70**	–
	Task4	(A) FDT	10	**50**	40	**40**	30	30
		(B) Clustering	40	**50**	10	20	10	**70**
		(C) Both	10	**60**	30	40	**60**	–

Table 9 User satisfaction, preference and completeness per interface (aggregated percentages)

	Interface	User satisfaction (%)			Preference (%)			Completeness of task (n4)
		Low	Medium	High	1	2	3	
Advanced users	(A) FDT	25	33.3	**41.6**	16.6	**41.6**	**41.6**	72.2
	(B) Clustering	33.3	**50**	16.6	25	16.6	**58.3**	55.5
	(C) Both	16.6	25	**58.3**	**58.3**	41.6	0	**77.7**
Regular users	(A) FDT	5	40	**55**	**42.5**	25	30	**46.6**
	(B) Clustering	37.5	**50**	12.5	20	15	**65**	38.3
	(C) Both	12.5	37.5	**50**	35	**57.5**	5	45

We can see that for advanced users, the combined interface (C) is the most preferred one, with respect to both rank aggregation methods. Regarding regular users, using the sum of the Plurality Rankings, the FDT (A) ties with the combined interface (C), while using Borda, interface (C) wins over (A) by a small margin. The clustering interface (B) gets the worst

Table 10 Number of user queries and clicks (as recorded in the log)

	Queries			Zoom points clicks								
				By clustering	By domain		By filetype		By date		By encoding	
	(A)	(B)	Both	Both	(A)	Both	(A)	Both	(A)	Both	(A)	Both
Avg. advanced	5.6	12.6	**5.33**	**6**	1.6	1.3	0.3	0.3	1	1	0	0
Avg. regular	11.9	13.2	**10.9**	**2.7**	1.8	0.8	1	1.1	2	1.7	0.1	0.6

Table 11 Aggregated ranking for preference per interface and task

		Interface	Preference	
			Borda	Plurality
Advanced	Task1	(A) FDT	7	**1**
		(B) Clustering	6	**1**
		(C) Both	**5**	**1**
	Task2	(A) FDT	8	–
		(B) Clustering	6	1
		(C) Both	**4**	**2**
	Task3	(A) FDT	5	1
		(B) Clustering	9	–
		(C) Both	**4**	**2**
	Task4	(A) FDT	5	–
		(B) Clustering	7	1
		(C) Both	**4**	**2**
	All	(A) FDT	25	2
		(B) Clustering	28	3
		(C) Both	**17**	**7**
Regular	Task1	(A) FDT	22	2
		(B) Clustering	22	2
		(C) Both	**16**	**6**
	Task2	(A) FDT	20	**4**
		(B) Clustering	22	**4**
		(C) Both	**18**	2
	Task3	(A) FDT	**14**	**7**
		(B) Clustering	29	–
		(C) Both	17	3
	Task4	(A) FDT	19	2
		(B) Clustering	25	2
		(C) Both	**16**	**4**
	All	(A) FDT	75	**15**
		(B) Clustering	98	8
		(C) Both	**67**	**15**

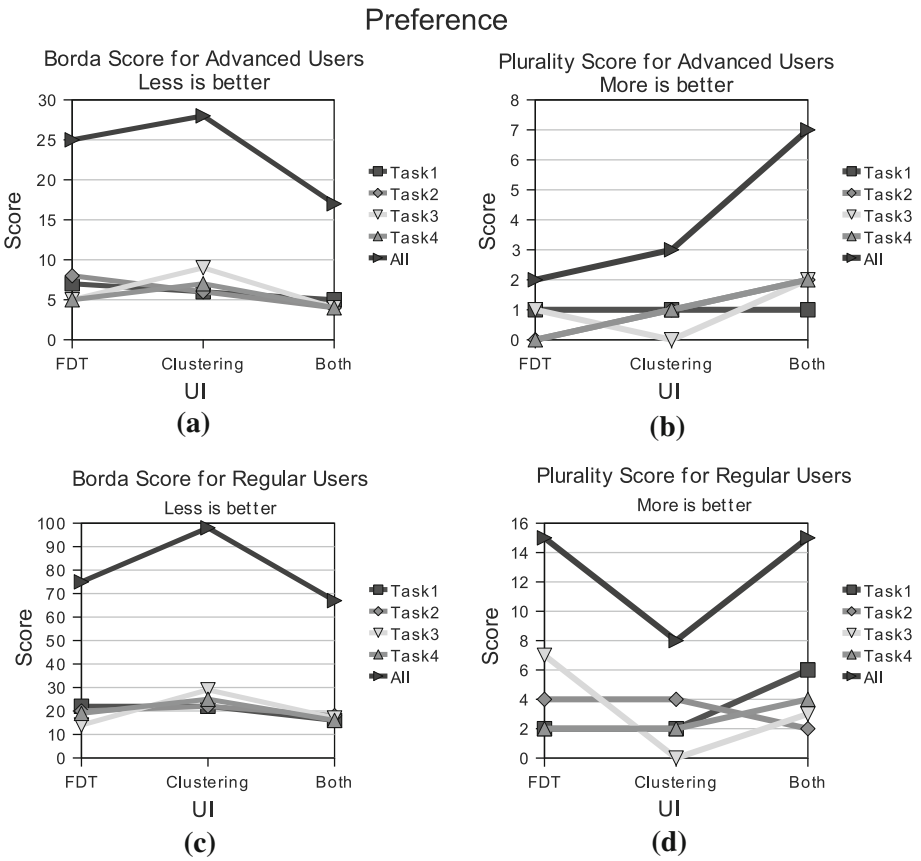

Fig. 12 **a** and **b**: Borda and Plurality preference derived scores for advanced users. **c** and **d**: Borda and Plurality preference derived scores for regular users

rankings in all cases, and from Table 9, we can see that it was the least preferred for 58.3% of the *advanced* users and for 65% of the *regular*. This is probably because users are either not familiar with clustering services (as there are only a few meta search engines—and not well known—that offer this feature) or because they do not yet trust such services (as they do not use them regularly).

It would be interesting to compute the level of agreement / disagreement of the raters; however, the available methods for this kind of analysis, e.g., the *Weighted Kappa for Multiple Raters* [5], do not have the constraint that each rater has to rank the subjects (i.e., to order them linearly); instead, one rater can give the same rating to more than one subjects. Consequently, such metrics would not provide statistically safe results.

5.2.4 User satisfaction

As shown in Table 9, advanced users were highly satisfied from (C), while regular users were most satisfied by (A), however, the difference in satisfaction between (A) and (C) is small: 55% for the first one and 50% for the second one.

If we look at the table with the detailed results (Table 8), we observe that the advanced users were more satisfied from *Clustering* than from *FDT* for the first two tasks, while in the last two tasks the opposite happened. One explanation why advanced users were satisfied by *FDT* in the last two tasks is because the facets *By filetype* and *By domain* are very helpful for these two tasks. Instead, they were not satisfied by *FDT* in the first task probably because the facet *By date* uses the modification time (and not the publication date which is required by $T1$), and thus, it is not very helpful. Regular users were not satisfied from clustering (B) in none of the tasks.

5.2.5 Other conclusions and synopsis

Figure 13 shows a statistical analysis over the results of our evaluation for user *satisfaction* and *preference*, for both regular and advanced users. In this analysis, we compute upper and lower limits with 95% confidence. For regular users, we can make the following observations:

– Only 5% of the regular users with $a \pm 9.2$ error were not satisfied by the FDT interface (A)

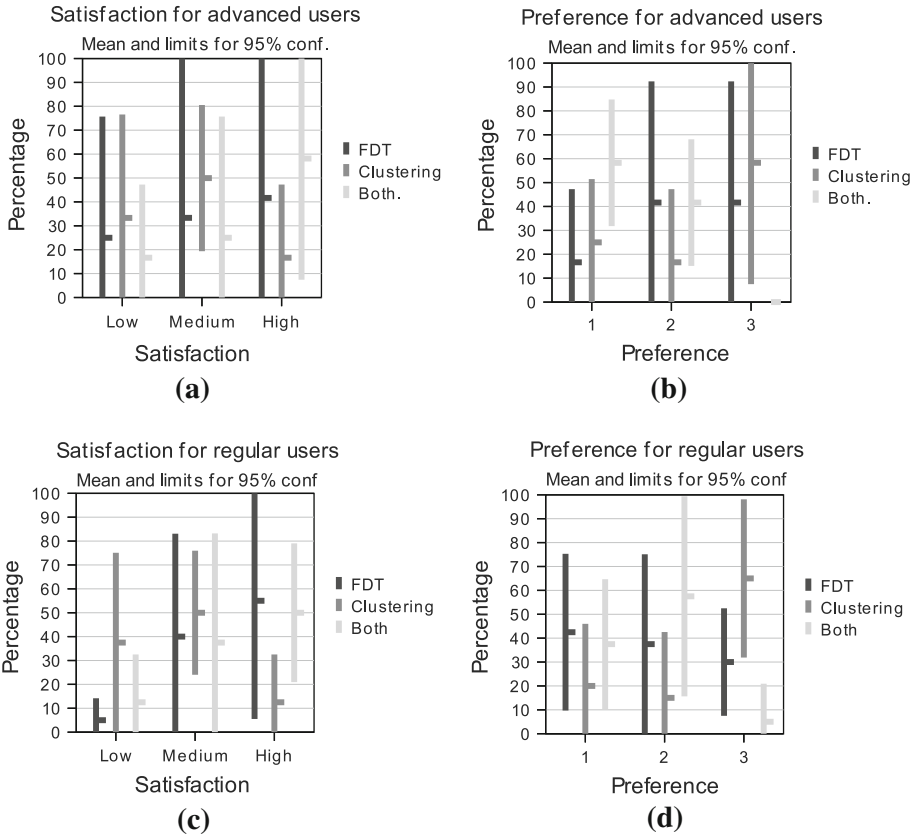

Fig. 13 **a** and **b**: Mean and limits for satisfaction and preference for advanced users. **c** and **d**: Mean and limits for satisfaction and preference for regular users

- Only 5% of the regular users with $a \pm 15.91$ error have low preference for the combined interface (C)
- Only 12.5% of the regular users, with $a \pm 20$ error, find the clustering interface highly satisfactory
- Only 12.5% of the regular users, with $a \pm 20$ error, have low satisfaction for the combined interface (C)

For the advanced users, we cannot come up with a clear conclusion due to the big errors (due to the small number of tasks).

Overall, we can conclude that the combination of *FDT* with *Clustering* is expected to help mainly users who are familiar with the functionality of each interface (especially with clustering), and for such users, this interface will probably be the most preferred. Users who are not familiar with these access methods are more satisfied with *FDT* (probably because they fully understand it immediately), than with the combination of both or with *Clustering* alone. This is quite expected since users who have not used real-time clustering before probably cannot understand it immediately (they may wonder whether the clusters contain overlapping or disjoint sets of pages) and/or do not trust it (since they have no experience on using such services). However, we have to remark that the tutorial was very brief, and it is possible that a more detailed and comprehensive tutorial (e.g., a hands on training session of 30 min) could turn the results of the regular users to converge to those of the advanced ones.

Finally, we should mention that both groups submitted the least number of queries when using (C) interface, which is evidence that the combination of interfaces makes information seeking less laborious.

6 Conclusion

The contribution of our work lies in: (a) proposing and motivating the need for exploiting both explicit and mined metadata during Web searching, (b) showing how automatic results clustering can be combined effectively and efficiently with the interaction paradigm of dynamic taxonomies by applying top-C clustering on demand, (c) providing incremental evaluation approaches for reusing the results of the more computationally expensive tasks (i.e., of snippet extraction), (d) reporting experimental results that demonstrate the feasibility of the approach, and (e) reporting the results of a comparative user study that gives interesting results regarding the effectiveness of the approach and its acceptance by users.

In the future, we plan to continue our work on speeding up the multidimensional browsing along the lines of [4,37,48], as well on further speeding up the incremental algorithms presented. Finally, we plan to incorporate additional facets, e.g., we could add facets corresponding to the outcome of document classifiers [33].

References

1. (Apr 2006) Special issue on supporting exploratory search. Commun ACM 49(4)
2. Anagnostopoulos A, Broder A, Punera K (2008) Effective and efficient classification on a search-engine model. Knowl Inf Syst 16(2):129–154
3. Azzopardi L (2009) Usage based effectiveness measures: monitoring application performance in information retrieval. In: Proceedings the 18th ACM conference on information and knowledge management (CIKM'09). ACM, New York, NY, USA, pp 631–640

4. Ben-Yitzhak O, Golbandi N, Har'El N, Lempel R, Neumann A, Ofek-Koifman S, Sheinwald D, Shekita E, Sznajder B, Yogev S (Feb 2008) Beyond basic faceted search. In: Proceedings of the international conference on web search and web data mining (WSDM'08). Palo Alto, California, USA, pp 33–44
5. Berry KJ, Johnston JE, Mielke PW (2008) Weighted kappa for multiple raters. Percept Motor Skills 107:837–848
6. Crabtree D, Gao X, Andreae P (Sep 2005) Improving web clustering by cluster selection. In: Proceedings of the IEEE/WIC/ACM international conference on web intelligence (WI'05). Compiegne, France, pp 172–178
7. Cutting D, Karger D, Pedersen J, Tukey J (June 1992) Scatter/Gather: a cluster-based approach to browsing large document collections. In: Proceedings of the 15th annual international ACM conference on research and development in information retrieval (SIGIR'92). Copenhagen, Denmark, pp 318–329
8. Dakka W, Ipeirotis P (Apr 2008) Automatic extraction of useful facet hierarchies from text databases. In: Proceedings of the 24th international conference on data engineering (ICDE'08). Cancún, México, pp 466–475
9. de Borda JC (1781) Memoire sur les Elections au Scrutin. Histoire de l'Academie Royale des Sciences, Paris
10. Doms A, Schroeder M (2005) GoPubMed: exploring PubMed with the gene ontology. Nucleic Acids Res 33(Web Server Issue):W783
11. Faulkner L (2003) Beyond the five-user assumption: benefits of increased sample sizes in usability testing. Behav Res Methods Instrum Comput 35(3):379–383
12. Ferragina P, Gulli A (May 2005) A personalized search engine based on web-snippet hierarchical clustering. In: Proceedings of the 14th international conference on world wide web (WWW'05), vol 5. Chiba, Japan, pp 801–810
13. Gelgi F, Davulcu H, Vadrevu S (June 2007) Term ranking for clustering web search results. In: 10th international workshop on the web and databases (WebDB'07), Beijing, China
14. Hearst M, Pedersen J (Aug 1996) Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In: Proceedings of the 19th annual international ACM conference on research and development in information retrieval (SIGIR'96), Zurich, Switzerland, pp 76–84
15. Hildebrand M, van Ossenbruggen J, Hardman L (Nov 2006) /facet: a browser for heterogeneous semantic web repositories. In: Proceedings of international semantic web conference (ISWC'06), Athens, GA, USA, pp 272–285
16. Hyvönen E, Mäkelä E, Salminen M, Valo A, Viljanen K, Saarela S, Junnila M, Kettula S (2005) MuseumFinland—Finnish museums on the semantic web. J Web Semant 3(2):25
17. Janruang J, Kreesuradej W (Nov 2006) A new web search result clustering based on true common phrase label discovery. In: Proceedings of the international conference on computational intelligence for modeling control and automation and international conference on intelligent agents web technologies and international commerce (CIMCA/IAWTIC'06), Washington, DC, USA, p 242. ISBN 0-7695-2731-0
18. Järvelin K, Price SL, Delcambre LML, Nielsen ML (2008) Discounted cumulated gain based evaluation of multiple-query IR sessions. In: European conference on information retrieval (ECIR), pp 4–15
19. Jing L, Ng M, Huang J (2010) Knowledge-based vector space model for text clustering. Knowl Inf Syst 25(1):1–21
20. Käki M, Aula A (2008) Controlling the complexity in comparing search user interfaces via user studies. Inf Process Manag 44(1):82–91
21. Karlson AK, Robertson GG, Robbins DC, Czerwinski MP, Smith GR (Apr 2006) FaThumb: a facet-based interface for mobile search. In: Proceedings of the conference on human factors in computing systems (CHI'06), Montréal, Québec, Canada, pp 711–720
22. Kelly D (2009) Methods for evaluating interactive information retrieval systems with users. Found Trends Inf Retr 3(1–2):1–224
23. Kopidaki S, Papadakos P, Tzitzikas Y (Oct 2009) STC+ and NM-STC: two novel online results clustering methods for web searching. In: Proceedings of the 10th international conference on web information systems engineering (WISE'09)
24. Krikelas J (1983) Information-seeking behavior: patterns and concepts. Drexel Libr Q 2:5–20
25. Kules B, Capra R (2008a) Creating exploratory tasks for a faceted search interface. In: Proceedings of the 2nd workshop on human-computer interaction (HCIR'08)
26. Kules B, Capra R (2008b) Creating exploratory tasks for a faceted search interface. In: workshop on computer interaction and information retrieval, HCIR 2008, pp 18–21
27. Kules B, Kustanowitz J, Shneiderman B (June 2006) Categorizing web search results into meaningful and stable categories using fast-feature techniques. In: Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries, (JCDL'06), Chapel Hill, NC, USA, pp 210–219. ISBN 1-59593-354-9

28. Kules B, Wilson M, Schraefel M, Shneiderman B (2008) From keyword search to exploration: how result visualization aids discovery on the web. Human-Computer Interaction Lab Technical Report HCIL-2008-06, University of Maryland
29. Kules B, Capra R, Banta M, Sierra T (2009) What do exploratory searchers look at in a faceted search interface? In: JCDL, pp 313–322
30. Lindgaard G, Chattratichart J (2007) Usability testing: what have we overlooked? In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI'07). ACM, New York, NY, USA. pp 1415–1424
31. Mäkelä E, Viljanen K, Lindgren P, Laukkanen M, Hyvönen E (2005) Semantic yellow page service discovery: the veturi portal. Poster paper at international semantic web conference (ISWC'05), Galway, Ireland
32. Mäkelä E, Hyvönen E, Saarela S (Nov 2006) Ontogator—a semantic view-based search engine service for web applications. In: Proceedings of international semantic web conference (ISWC'06), Athens, GA, USA, pp 847–860
33. Malik H, Fradkin D, Moerchen F (2010) Single pass text classification by direct feature weighting. Knowl Inf Syst, pp 1–20. ISSN 0219-1377
34. Papadakos P, Theoharis Y, Marketakis Y, Armenatzoglou N, Tzitzikas Y (Aug 2008) Mitos: design and evaluation of a dbms-based web search engine. In: Proceedings of the 12th Pan-Hellenic conference on informatics (PCI'08), Greece
35. Papadakos P, Kopidaki S, Armenatzoglou N, Tzitzikas Y (Sept 2009) Exploratory web searching with dynamic taxonomies and results clustering. In: Proceedings of the 13th European conference on digital libraries (ECDL'09)
36. Petrelli D (2008) On the role of user-centered evaluation in the advancement of interactive information retrieval. Inf Process Manage 44(1):22–38. ISSN 0306–4573
37. Sacco G (2006) Some research results in dynamic taxonomy and faceted search systems. In: SIGIR'2006 workshop on faceted search
38. Sacco GM, Tzitzikas Y (eds)(2009) Dynamic taxonomies and faceted search: theory, practise and experience. Springer. ISBN 978-3-642-02358-3
39. Schraefel MC, Karam M, Zhao S (Aug 2003) mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia. In: Proceedings of workshop on adaptive hypermedia and adaptive web based systems, Nottingham, UK, pp 217–235
40. Stefanowski J, Weiss D (May 2003) Carrot2 and language properties in web search results clustering. In: Proceedings of the international atlantic web intelligence Conference (AWIC'03). Springer, Madrid, Spain
41. Tzitzikas Y, Armenatzoglou N, Papadakos P (2008) FleXplorer: a framework for providing faceted and dynamic taxonomy-based information exploration. In: 19th international workshop on database and expert systems applications (FIND'08 at DEXA'08), Torino, Italy, pp 392–396
42. Wang J, Mo Y, Huang B, Wen J, He L (2008) Web search results clustering based on a novel suffix tree structure. In: Proceedings of 5th international conference on autonomic and trusted computing (ATC'08), vol 5060, Oslo, Norway, pp 540–554
43. Wang P, Hu J, Zeng H, Chen Z (2009) Using wikipedia knowledge to improve text classification. Knowl Inf Syst 19(3):265–281
44. White RW, Drucker SM, Marchionini G, Hearst MA, Schraefel MC (2007) Exploratory search and hci: designing and evaluating interfaces to support exploratory search interaction. In: Rosson MB, Gilmore DJ (eds) CHI extended abstracts. ACM, New York pp 2877–2880
45. Wilson ML, Schraefel MC (Apr 2007) Bridging the Gap: using IR models for evaluating exploratory search interfaces. In: Workshop on exploratory search and HCI (SIGCHI'2007). ACM
46. Xing D, Xue G, Yang Q, Yu Y (Feb 2008) Deep classifier: automatically categorizing search results into large-scale hierarchies. In: Proceedings of the international conference on web search and web data mining (WSDM'08), Palo Alto, California, USA, pp 139–148
47. Yee K, Swearingen K, Li K, Hearst M (Apr 2003a) Faceted metadata for image search and browsing. In: Proceedings of the conference on human factors in computing systems (CHI'03), Ft. Lauderdale, Florida, USA, pp 401–408
48. Yee K, Swearingen K, Li K, Hearst M (2003b) Faceted metadata for image search and browsing. In: Proceedings of the SIGCHI conference on human factors in computing systems, pp 401–408
49. Zamir O, Etzioni O (Aug 1998) Web document clustering: a feasibility demonstration. In: Proceedings of the 21th annual international ACM conference on research and development in information retrieval (SIGIR'98), Melbourne, Australia, pp 46–54

50. Zeng H, He Q, Chen Z, Ma W, Ma J (July 2004) Learning to cluster web search results. In: Proceedings of the 27th annual international conference on research and development in information retrieval (SIGIR'04), Sheffield, UK, pp 210–217

Author Biographies

Panagiotis Papadakos conducted his undergraduate and graduate studies (MSc) in the Computer Science Department, University of Crete (Greece). From February 2008 to December 2010, he was a research assistant at the Information Systems Laboratory of the Institute of Computer Science at the Foundation for Research and Technology, Hellas (ICS-FORTH). Currently, he is PhD candidate in the same department, and his research focuses on exploratory methods for interactive Web Searching.

Nikos Armenatzoglou conducted his undergraduate and graduate studies (MSc) in the Computer Science Department, University of Crete (Greece). From November 2007 to March 2010, he was also a research assistant at the Information Systems Laboratory of the Institute of Computer Science at the Foundation for Research and Technology, Hellas (ICS-FORTH). Currently, he is PhD candidate at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests apart from information retrieval include spatial and spatio-temporal databases.

Stella Kopidaki conducted her undergraduate and graduate studies (MSc) in the Computer Science Department, University of Crete (Greece). From November 2008 to June 2009 she was also a research assistant at the Information Systems Laboratory of the Institute of Computer Science at the Foundation for Research and Technology, Hellas (ICS-FORTH). Currently, she works at the industry as a software engineer.

Yannis Tzitzikas is currently Assistant Professor of Information Systems in the Computer Science Department, University of Crete (Greece) and Associate Researcher of the Information Systems Laboratory at FORTH-ICS (Greece). Before joining UofCrete and FORTH-ICS, he was postdoctoral fellow at the University of Namur (Belgium) and ERCIM postdoctoral fellow at ISTI-CNR (Pisa, Italy) and at VTT Technical Research Centre of Finland. He conducted his undergraduate and graduate studies (MSc, PhD) in the Computer Science Department at the University of Crete. His research interests fall in the intersection of the following areas: Information Systems, Information Indexing and Retrieval, Conceptual Modeling and Knowledge Representation and Reasoning. The results of his research have been published in more than 60 papers in refereed international conferences and journals, and he has received two best paper awards (at CIA'2003 and ISWC'07). Co-editor of the book "Dynamic Taxonomies and Faceted Search" by Springer (2009).