Exploratory Web Searching with Dynamic Taxonomies and Results Clustering

Panagiotis Papadakos, Stella Kopidaki,
Nikos Armenatzoglou, and Yannis Tzitzikas

Institute of Computer Science, FORTH-ICS, Greece
Computer Science Department, University of Crete, Greece
{papadako,skopidak,armenan,tzitzik}@ics.forth.gr

Abstract. This paper proposes exploiting both *explicit* and *mined* metadata for enriching Web searching with *exploration* services. On-line results clustering is useful for providing users with overviews of the results and thus allowing them to restrict their focus to the desired parts. On the other hand, the various metadata that are available to a WSE (Web Search Engine), e.g. domain/language/date/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. This combination results to an effective, flexible and efficient exploration experience.

1 Introduction

Web Search Engines (WSEs) typically return a ranked list of documents that are relevant to the query submitted by the user. 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 to these problems is *results clustering* [25] which provides a quick 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 wouldn't practically find especially if these documents are low ranked (and thus not in first result pages). Another solution is to exploit the various metadata that are available to WSEs (like domain, dates, language, filetype, etc). 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 promising approach is to exploit such metadata in the context of the interaction paradigm of *faceted and dynamic taxonomies* [18,21], a paradigm

M. Agosti et al. (Eds.): ECDL 2009, LNCS 5714, pp. 106–118, 2009.

that is used more and more nowadays. Its main benefit is that it shows only those terms of the taxonomy that lead to non-empty answer sets, and the user can gradually restrict his focus using several criteria by clicking. In addition this paradigm allows users to switch easily between searching and browsing.

There are works [1,14] in the literature that compare automatic results clustering with guided exploration (through dynamic faceted taxonomies). In this work we propose combining these two approaches. In a nutshell, 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 with the interaction paradigm of dynamic taxonomies, by clustering on-demand the top elements of the user focus, (c) providing incremental evaluation algorithms, and (d) reporting experimental results that prove the feasibility and the effectiveness of the approach.

To the best of our knowledge, there are no other WSEs that offer the same kind of information/interaction. A somehow related interaction paradigm that involves clustering is Scatter/Gather [4,8]. This paradigm allows the users to select clusters, subsequently the documents of the selected clusters are clustered again, 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 on-line, 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.

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

2 Requirements and Background

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 [4,8]), or to their (query-dependent) snippets (as in [25,20,6,26,7,22]). 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 [6,5,23] 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, that represent the contents of the retrieved documents or can be enriched, in order to optimize the clustering and improve the quality of cluster labels.

One very efficient and effective approach is the *Suffix Tree Clustering (STC)* [25] where search results (mainly snippets) can be clustered fast (in linear time), incrementally, and each cluster is labeled with a phrase. 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 [25]. Several variations of STC have emerged recently (e.g. [3,11,22]).

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. *Dynamic taxonomies* [18] (faceted or not) is a general knowledge management model based on a multidimensional classification of heterogeneous data objects and is used to explore and browse complex information bases in a guided, yet unconstrained way through a visual interface. Features of faceted metadata search include (a) display of current results in multiple categorization schemes (facets) (e.g. based on metadata terms, such as size or date), (b) display categories leading to non-empty results, and (c) display of the count of the indexed objects of each category (i.e. the number of results the user will get by selecting this category). An example of the idea assuming only one facet, is shown in Figure 1. Figure 1(a) shows a taxonomy and 8 indexed objects (1-8). Figure 1(b) shows the dynamic taxonomy if we restrict our focus to the objects {4,5,6}. Figure 1(c) shows the browsing structure that could be provided at the GUI layer and Figure 1(d) 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 empty, or the top term

[1] www.dmoz.org

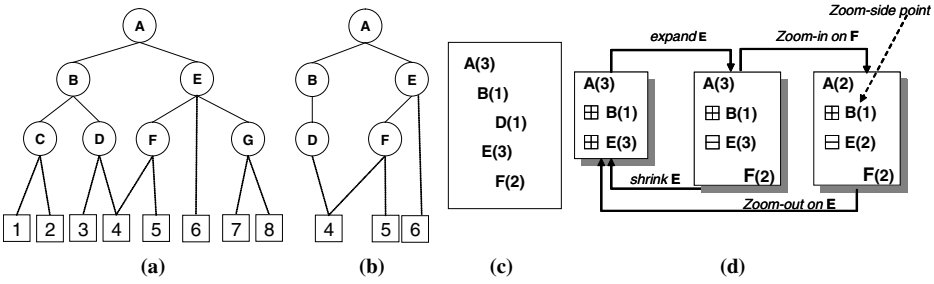

Fig. 1. Dynamic Taxonomies

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 (Figure 1(b) shows the restriction of a taxonomy on the objects {4,5,6}). At any point during the interaction, we compute and provide to the user the immediate zoom-in/out/side points along with count information (as shown in Figure 1(d)). When the user selects one of these points then the selected term is added to the focus, and so on. Note that the user can exploit the faceted structure and at each step may decide to select a zoom point from different facet (e.g. filetype, modification date, language, web domain, etc).

Examples of applications of faceted metadata-search include: e-commerce (e.g. ebay), library and bibliographic portals (e.g. DBLP), museum portals (e.g. [10] and Europeana[2]), mobile phone browsers (e.g. [12]), specialized search engines and portals (e.g. [16]), Semantic Web (e.g. [9,15]), general purpose WSEs (e.g. Google Base), and other frameworks (e.g. mSpace[19]).

Related Work. Systems like [24,9,13,15,2] support multiple facets, each associated with a taxonomy which can be predefined. Moreover, the systems described in [24,9,15] support ways to configure the taxonomies that are available during browsing based on the contents of the results. Specifically, [24] enriches the values of the object descriptions with more broad terms by exploiting WordNet, [9] supports rules for deciding which facets should be used according to the type of data objects based on RDF, and [15] supports reclassification of the objects to predefined types. However, none of these systems apply content-based results clustering, re-constructing the cluster tree taxonomy while the user explores the answer set. Instead they construct it once per each submitted query.

3 On-Demand Integration

Dynamic taxonomies can load and handle thousands of objects very fast (as it will be described in Section 4.1). However, the application of results clustering on

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

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* $C = 100$*) snippets of the current focus.* 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 let $Ans(q)$ be the answer of this query. We shall use A_f to denote top-K (usually $K < 10000$) objects of $Ans(q)$ and A_c to denote top-C objects of $Ans(q)$. Clearly, $A_c \subseteq A_f \subseteq Ans(q)$. The steps of the process are the following:

(1) The snippets of the elements of A_c are generated.
(2) Clustering is applied on the elements of A_c, generating a cluster label tree *clt*.
(3) The set of A_f (with their metadata), as well as *clt*, are loaded to `FleXplorer`, a module for creating and managing the faceted dynamic taxonomy. 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 $K - C$ objects).
(4) `FleXplorer` computes and delivers to the GUI the (immediate) zoom points.

The user can start browsing by selecting the desired zoom point(s). When he selects a zoom point or submits a new query, steps (1)-(4) are performed again.

3.1 Results Clustering: HSTC

We adopt an extension of STC [25] that we have devised called HSTC. As in STC, this algorithm begins by constructing the suffix tree of the titles and snippets and then it scores each node of that tree. HSTC uses a scoring formula that favors the occurrences in titles, does not merge base clusters and returns an hierarchically organized set of cluster labels. In brief, the advantages of HSTC are: (a) the user never gets unexpected results, as opposed to the existing STC-based algorithms which 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[3] and it is around two times faster that the plain STC (see Section 4.1). We do not report here the details, because any result clustering algorithm could be adopted. However, it is worth noticing that the hierarchy of cluster labels by HSTC, can be considered as a subsumption relation since it satisfies $c < c' \implies Ext(c) \subseteq Ext(c')$, where $Ext(c_i)$ denotes the documents that belong to cluster c_i. This property allows exploiting the interaction paradigm of *dynamic taxonomies*. HSTC, like STC, results in overlapping clusters.

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

3.2 Dynamic Taxonomies: `FleXplorer`

`FleXplorer` is a main memory API (Application Programmatic Interface) that 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), etc. Regarding user interaction, the framework provides methods for setting the focus and getting the applicable zoom points.

3.3 Incremental Evaluation Algorithm

Here we present an incremental approach for exploiting past computations and results. Let A_f be the objects of the current focus. If the user selects a zoom point he moves to a different focus. Let A'_f denote the top-K elements of the new focus, and A'_c the top-C of the new focus. The steps of the algorithm follow.

(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) HSTC 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.

Let's now focus on Step (3), i.e. on the incremental application of HSTC. 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. Let sf' 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. The preserved suffix tree can be either the initial suffix tree or the pruned suffix tree. Each node of the initial tree corresponds to a single word, while the pruned tree is more compact in the sense that if a node contains only one child node and both nodes contain the same objects, they are collapsed to one single node that has as label the concatenation of the labels of the constituent nodes. Scoring is done over the pruned suffix tree. However to add and delete objects to/from a pruned suffix tree sometimes requires "splitting" nodes (due to the additions) and pruning extra nodes (due to the deletions). On the other hand, if the unpruned suffix tree is preserved, then additions and deletions are performed right away

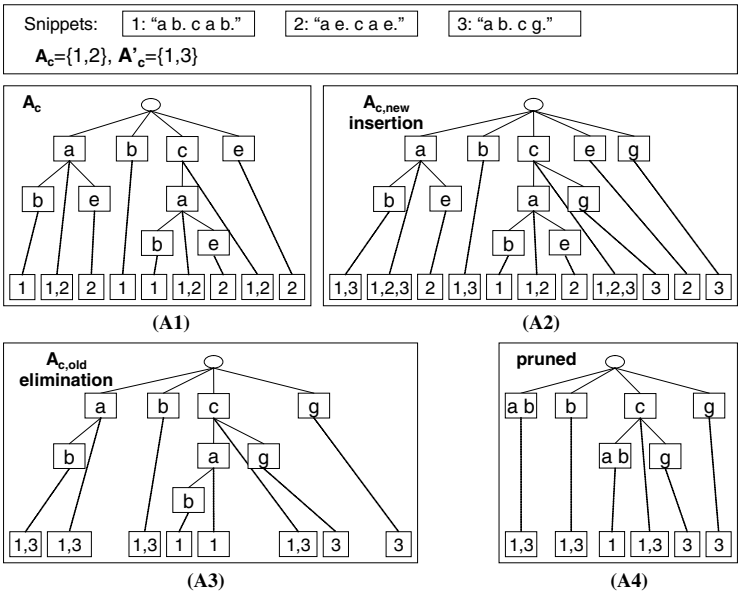

Fig. 2. Incremental Evaluation

and pruning takes place at the end. 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 2 illustrates an example for the case where $A_c = \{1, 2\}$ and $A'_c = \{1, 3\}$.
- *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.

We have to note that sf can be considered as a cache of snippets and recall that snippet generation is more expensive than clustering. The gained speedup is beneficial both for a standalone WSE as well for a Meta WSE, since fetching and parsing of snippets are reused. The suffix tree sf has to be constructed from scratch whenever the user submits a new query and is incrementally updated while the user browses the information space by selecting zoom points.

4 Implementation and Experimental Evaluation

The implementation was done in the context of `Mitos`[4] [17], which is a prototype WSE[5]. `FleXplorer` is used by `Mitos` for offering general purpose browsing and exploration services. Currently, and on the basis of the top-K answer of each submitted query, the following five facets are created and offered to users:

- the hierarchy or clusters derived by HSTC
- web domain, a hierarchy is defined (e.g. $csd.uoc.gr < uoc.gr < gr$),
- format type (e.g. pdf, html, doc, etc), no hierarchy is created in this case
- language of a document based on the encoding of a web page and
- (modification) date hierarchy.

When the user interacts with the clustering facet we do not apply the re-clustering process (i.e. steps (1) and (2) of the on-demand algorithm). This behavior is more intuitive, since it preserves the clustering hierarchy while the user interacts with the clustering facet (and does not frustrate the user 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 $K - C$ objects).

4.1 Experimental Results

Clustering Performance. It is worth noting that the most time consuming subtask is not the clustering itself but the extraction of the snippets from the cached copies of textual contents of the pages[6]. 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 performed using a Pentium IV 4 GHz, with 2 GB RAM, running Linux Debian.

Table 1. 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 HSTC	0.117	0.189	0.311	0.449	0.648

Table 1 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

[4] http://groogle.csd.uoc.gr:8080/mitos/

[5] Under development by the Department of Computer Science of the University of Crete and FORTH-ICS.

[6] The snippets in our experiments contain up to two sentences (11 words maximum each) where the query terms appear most times.

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 HSTC is at least two times faster than STC. This is because HSTC does not have to intersect and merge base clusters.

Dynamic Taxonomies Performance. Loading times of `FleXplorer` have been thoroughly measured in [21]. In brief, the computation of zoom-in 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.

Overall Performance. In this experiment we measured the overall cost, i.e. cluster generation times (snippet generation and clustering algorithm execution) and the dynamic taxonomies times (to compute the zoom points and and to load the new clustering labels to the corresponding facet). Moreover, we compare

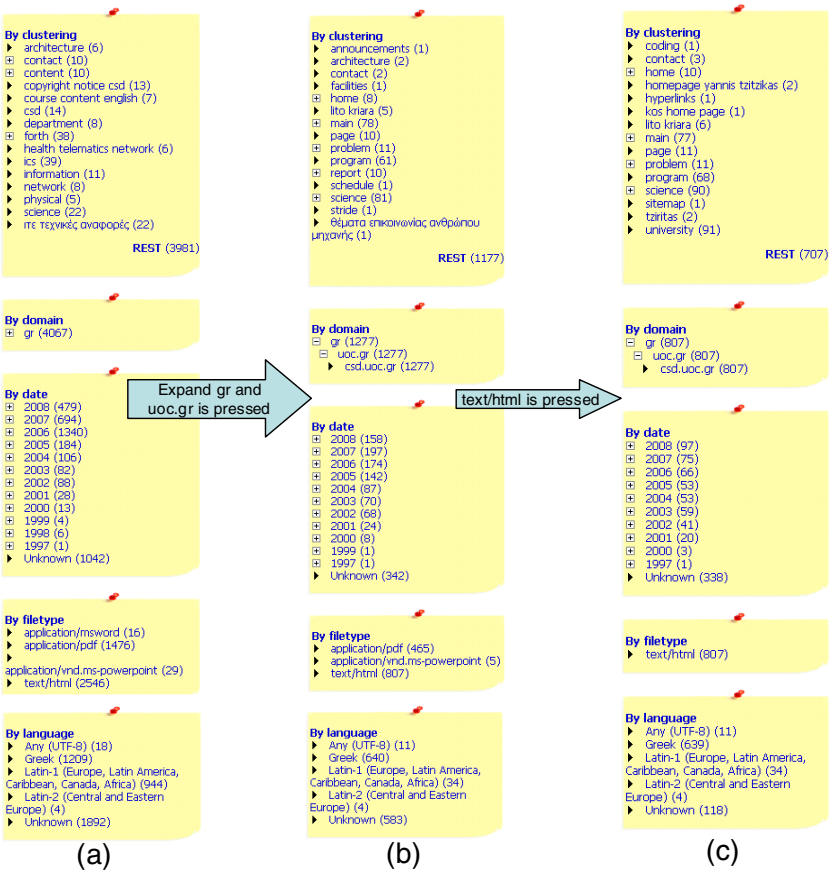

Fig. 3. Steps (a)-(c) of running scenario

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 we used includes: (a) the execution of the query *crete* which returns 4067 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 (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 3 shows the facet terms after steps (a), (b) and (c), as they are displayed in the left bar of the WSE GUI. We set $K = 10000$ (i.e. the whole answer set is loaded) and repeated the above steps for the following values of C:100, 200 ... 500. We do not measure the cost of the query evaluation time. In all experiments `FleXplorer` computes count information.

Table 2. Top-C Comparison of Incremental/Non-Incremental Algorithms (in seconds)

	Step (a)	Step (b)	Step (c)
top-100	$\|c_a\| = 100$	$\|c_a \cap c_b\| = 43$, overlap=43%	$\|c_b \cap c_c\| = 85$, overlap=85%
Non-Incr.	0.914	0.443	0.204
Incr.	0.931	0.431	0.101
top-200	$\|c_a\| = 200$	$\|c_a \cap c_b\| = 71$, overlap=35.5%	$\|c_b \cap c_c\| = 113$, overlap=56.5%
Non-Incr.	1.266	1.245	0.789
Incr.	1.245	0.965	0.68
top-300	$\|c_a\| = 300$	$\|c_a \cap c_b\| = 74$, overlap=24.6%	$\|c_b \cap c_c\| = 201$, overlap=67.7%
Non-Incr.	1.676	2.534	1.383
Incr.	1.65	2.527	0.761
top-400	$\|c_a\| = 400$	$\|c_a \cap c_b\| = 85$, overlap=21.5%	$\|c_b \cap c_c\| = 252$, overlap=63%
Non-Incr.	2.246	3.067	1.944
Incr.	2.118	3.335	0.942
top-500	$\|c_a\| = 500$	$\|c_a \cap c_b\| = 97$, overlap=19.4%	$\|c_b \cap c_c\| = 324$, overlap=64.8%
Non-Incr.	2.483	3.495	2.001
Incr.	2.493	3.652	0.751

Table 2 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 an incremental approach of HSTC, for the $top - C$ elements. It is evident that for top-100 and top-200 values, the results are presented to the user almost instantly (around 1 second), making the proposed on demand clustering method suitable as an online task. Moreover, we can see that there is a linear correlation between time cost and the top-C value. Finally, calculating and loading clusters for the top-500 documents, costs around 3 seconds making even big top-C configurations a feasible configuration.

Comparing the incremental and the non-incremental algorithm, we observe a significant speedup whenever the overlap is more than 50%, for our scenario. At step (a) the suffix tree construction is the same for both algorithms as the suffix tree sf has to be constructed from scratch. For step (b) there are small variations due to the small overlap, so the time saved from the snippets generation/parsing is compensated by the time needed for eliminating old objects. Specifically, the incremental algorithm is faster for the top-200 case and slower

for the top-{400, 500} cases which have the lowest overlap. For the other cases performance is almost the same. Notice that although the top-100 case has the biggest overlap of all, there are no differences in the execution time of the two algorithms. This is probably due to the fact that the overlapping documents have fast snippet generation times, while the rest are big sized. At step (c) the benefit from the incremental approach is clear, since it is almost twice as fast as the non incremental one. Specifically, the best speedup is in the case of top-500, where overlap reaches 65% and the execution time of the non-incremental is 2.001, while for the incremental is just 0.751.

5 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, and (d) reporting experimental results that prove the feasibility and the effectiveness of this approach. In the future we plan to conduct a user study for investigating what top-C value most users prefer. Finally, we plan to continue our work on further speeding up the incremental algorithms presented.

References

1. Special issue on Supporting Exploratory Search. Communications of the ACM 49(4) (April 2006)
2. Ben-Yitzhak, O., Golbandi, N., Har'El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E., Sznajder, B., Yogev, S.: Beyond basic faceted search. In: Procs. of the Intern. Conf. on Web Search and Web Data Mining (WSDM 2008), Palo Alto, California, USA, February 2008, pp. 33–44 (2008)
3. Crabtree, D., Gao, X., Andreae, P.: Improving web clustering by cluster selection. In: Procs. of the IEEE/WIC/ACM Intern. Conf. on Web Intelligence (WI 2005), Compiegne, France, September 2005, pp. 172–178 (2005)
4. Cutting, D.R., Karger, D., Pedersen, J.O., Tukey, J.W.: Scatter/Gather: A cluster-based approach to browsing large document collections. In: Procs. of the 15th Annual Intern. ACM Conf. on Research and Development in Information Retrieval (SIGIR 1992), Copenhagen, Denmark, June 1992, pp. 318–329 (1992)
5. Dakka, W., Ipeirotis, P.G.: Automatic extraction of useful facet hierarchies from text databases. In: Procs. of the 24th Intern. Conf. on Data Engineering (ICDE 2008), Cancún, México, April 2008, pp. 466–475 (2008)
6. Ferragina, P., Gulli, A.: A personalized search engine based on web-snippet hierarchical clustering. In: Procs. of the 14th Intern. Conf. on World Wide Web (WWW 2005), Chiba, Japan, May 2005, vol. 5, pp. 801–810 (2005)
7. Gelgi, F., Davulcu, H., Vadrevu, S.: Term ranking for clustering web search results. In: 10th Intern. Workshop on the Web and Databases (WebDB 2007), Beijing, China (June 2007)

8. Hearst, M.A., Pedersen, J.O.: Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In: Procs. of the 19th Annual Intern. ACM Conf. on Research and Development in Information Retrieval (SIGIR 1996), Zurich, Switzerland, pp. 76–84 (August 1996)
9. Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: A browser for heterogeneous semantic web repositories. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 272–285. Springer, Heidelberg (2006)
10. Hyvönen, E., Mäkelä, E., Salminen, M., Valo, A., Viljanen, K., Saarela, S., Junnila, M., Kettula, S.: MuseumFinland – Finnish museums on the semantic web. Journal of Web Semantics 3(2), 25 (2005)
11. Janruang, J., Kreesuradej, W.: A new web search result clustering based on true common phrase label discovery. In: Procs. of the Intern. Conf. on Computational Intelligence for Modelling Control and Automation and Intern. Conf. on Intelligent Agents Web Technologies and International Commerce (CIMCA/IAWTIC 2006), Washington, DC, USA, November 2006, p. 242 (2006)
12. Karlson, A.K., Robertson, G.G., Robbins, D.C., Czerwinski, M.P., Smith, G.R.: FaThumb: A facet-based interface for mobile search. In: Procs. of the Conf. on Human Factors in Computing Systems (CHI 2006), Montréal, Québec, Canada, April 2006, pp. 711–720 (2006)
13. Kules, B., Kustanowitz, J., Shneiderman, B.: Categorizing web search results into meaningful and stable categories using fast-feature techniques. In: Procs. of the 6th ACM/IEEE-CS Joint Conf. on Digital Libraries (JCDL 2006), pp. 210–219. Chapel Hill, NC (2006)
14. Kules, B., Wilson, M., Schraefel, M., Shneiderman, B.: 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, pp. 2008–06 (2008)
15. Mäkelä, E., Hyvönen, E., Saarela, S.: Ontogator - a semantic view-based search engine service for web applications. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 847–860. Springer, Heidelberg (2006)
16. Mäkelä, E., Viljanen, K., Lindgren, P., Laukkanen, M., Hyvönen, E.: Semantic yellow page service discovery: The veturi portal. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729. Springer, Heidelberg (2005)
17. Papadakos, P., Theoharis, Y., Marketakis, Y., Armenatzoglou, N., Tzitzikas, Y.: Mitos: Design and evaluation of a dbms-based web search engine. In: Procs. of the 12th Pan-Hellenic Conf. on Informatics (PCI 2008), Greece (August 2008)
18. Sacco, G.M.: Dynamic taxonomies: A model for large information bases. IEEE Transactions on Knowledge and Data Engineering 12(3), 468–479 (2000)
19. Schraefel, M.C., Karam, M., Zhao, S.: mSpace: Interaction design for user-determined, adaptable domain exploration in hypermedia. In: Procs of Workshop on Adaptive Hypermedia and Adaptive Web Based Systems, Nottingham, UK, August 2003, pp. 217–235 (2003)
20. Stefanowski, J., Weiss, D.: Carrot2 and language properties in web search results clustering. In: Menasalvas, E., Segovia, J., Szczepaniak, P.S. (eds.) AWIC 2003. LNCS (LNAI), vol. 2663, pp. 240–249. Springer, Heidelberg (2003)
21. Tzitzikas, Y., Armenatzoglou, N., Papadakos, P.: FleXplorer: A framework for providing faceted and dynamic taxonomy-based information exploration. In: 19th Intern. Workshop on Database and Expert Systems Applications (FIND 2008 at DEXA 2008), Torino, Italy, pp. 392–396 (2008)

22. Wang, J., Mo, Y., Huang, B., Wen, J., He, L.: Web search results clustering based on a novel suffix tree structure. In: Rong, C., Jaatun, M.G., Sandnes, F.E., Yang, L.T., Ma, J. (eds.) ATC 2008. LNCS, vol. 5060, pp. 540–554. Springer, Heidelberg (2008)
23. Xing, D., Xue, G.R., Yang, Q., Yu, Y.: Deep classifier: Automatically categorizing search results into large-scale hierarchies. In: Procs. of the Intern. Conf. on Web Search and Web Data Mining (WSDM 2008), Palo Alto, California, USA, February 2008, pp. 139–148 (2008)
24. Yee, K., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: Procs. of the Conf. on Human Factors in Computing Systems (CHI 2003), Ft. Lauderdale, Florida, USA, April 2003, pp. 401–408 (2003)
25. Zamir, O., Etzioni, O.: Web document clustering: A feasibility demonstration. In: Procs. of the 21th Annual Intern. ACM Conf. on Research and Development in Information Retrieval (SIGIR 1998), Melbourne, Australia, August 1998, pp. 46–54 (1998)
26. Zeng, H.J., He, Q.C., Chen, Z., Ma, W.Y., Ma, J.: Learning to cluster web search results. In: Procs. of the 27th Annual Intern. Conf. on Research and Development in Information Retrieval (SIGIR 2004), Sheffield, UK, July 2004, pp. 210–217 (2004)