Int J Softw Tools Technol Transfer (2009) 11:469–484
DOI 10.1007/s10009-009-0125-2

SPECIAL SECTION ON WEB SYSTEMS EVOLUTION

An approach and an Eclipse-based environment for enhancing the navigation structure of Web sites

Giuseppe Scanniello · Damiano Distante · Michele Risi

Published online: 30 October 2009

Abstract This paper presents an approach based on information retrieval and clustering techniques for automatically enhancing the navigation structure of a Web site for improving navigability. The approach increments the set of navigation links provided in each page of the site with a semantic navigation map, i.e., a set of links enabling navigating from a given page to other pages of the site showing similar or related content. The approach uses Latent Semantic Indexing to compute a dissimilarity measure between the pages of the site and a graph-theoretic clustering algorithm to group pages showing similar or related content according to the calculated dissimilarity measure. AJAX code is finally used to extend each Web page with an associated semantic navigation map. The paper also presents a prototype of a tool developed to support the approach and the results from a case study conducted to assess the validity and feasibility of the proposal.

Keywords Web site evolution · Navigation structure · Navigation evolution · Reverse engineering · Clone detection · Clustering · Information retrieval · Latent semantic indexing · Semantic clustering · Semantic navigation map

G. Scanniello (✉)
Department of Mathematics and Computer Science,
University of Basilicata, Potenza, Italy
e-mail: giuseppe.scanniello@unibas.it

D. Distante
Faculty of Economics, Tel.M.A. University, Rome, Italy
e-mail: damiano.distante@unitelma.it

M. Risi
Department of Mathematics and Computer Science,
University of Salerno, Salerno, Italy
e-mail: mrisi@unisa.it

1 Introduction

One of the key factors of success of the World Wide Web has been global and simple access to information thanks to an unlimited number of links connecting any part of the Web to any other. Everyone using the Web today to publish or access information takes it for granted that any available page will be accessible to anyone who is connected to the Internet [31].

Similarly, the success of a Web site, particularly information intensive Web sites, depends in part on easy and quick access to the information it provides, i.e., on its navigability. The ease of navigation, indeed, is one of the critical factors in determining the usability of a Web site, i.e., the capability of the Web site to support the effective, efficient and satisfactory accomplishment of user tasks [40], particularly information gathering tasks.

The importance of navigation in Web sites[1] is also demonstrated by the attention devoted to this aspect by basically all most known Web engineering methods [22], including OOHDM [35], WebML [6], UWE [23], and UWAT+ [14]. All of these methods, indeed, devote a specific design activity and a specific design model to define the navigation structure of a Web site, i.e., the organization of contents into units of consumption and the navigation enabled by links between these units. This design activity is usually named *Navigation* (or *Hypertext*) *Design*. One of the goals of this activity is the production of the *Navigation Model* that, independently from the considered Web engineering method, is usually based on two main modeling concepts: *Node* and *Link*. Nodes are defined as self-contained uniquely identifiable units of information from/to which the user can navigate in the considered Web

[1] Here and elsewhere in the paper, the term "Web site" can be generalized into that of "Web application", as our focus is on their navigation structure.

site. Links are used to connect nodes and to enable navigation between them. Links between nodes are arranged to form particular *access structures* (a.k.a., *navigation patterns*, e.g., *guided tours*, *indexes*, etc.) to support specific user information access goals (e.g., quick access to the most important or most recent contents of the site) or specific application requirements (e.g., provide a guided tour to the contents of the site on a specific topic) [22].

The adoption of a Web engineering approach to develop a Web site, by organizing the entire development process and by providing developers with the right support in methods, models, and tools for design and implementation, helps in producing higher quality Web sites that better satisfy stakeholders' goals and final users' expectations. In particular, when any of these approaches is used, it is expected that the resulting Web site will have a navigation structure that, by implementing the designed navigation model, satisfies the users' requirements in terms of contents reachability and navigability.

However, due to time-to-market constraints, lack of proper skills, and/or poor acceptability support [3,18], such Web engineering approaches very often are not used in the industrial practice and little effort is devoted, in particular, to design a Web site prior to implement it. Other times, Web engineering approaches are only used for developing and deploying the first version of the site and then neglected during the rest of its life time, when new contents and/or functionalities are introduced, and others removed.

Both the just described practices may lead, at a certain time, to Web sites suffering from poor usability and incorrect functioning. Regarding navigability, in particular, it may happen that: (i) certain contents become difficult to reach (because of navigation paths too long or too complex) or unreachable (because of missing navigation paths to them); (ii) new published contents may miss links to existing related contents and vice versa; (iii) existing access structures (e.g., guided tours and indexes) may become incomplete, missing to include links toward new related published contents, etc. As a consequence, the user may never get to some of the contents of her/his interest, or s/he may abandon the site because of poor navigability.

It is in these situations, but not only in these, that it may be particularly useful in applying the approach proposed in this paper to enhance the navigation structure of a Web site. The approach and the supporting tool we propose enable the automatic extension of the navigation structure of a Web site by incrementing each of its pages with a set of links connecting the page with others pages of the site presenting similar or semantically related content. We call this set of links as *Semantic Navigation Map*.

As it will also be described later in this paper, another context in which our approach can be profitably applied is that of Web sites built by means of a Content Management System (CMS). Usually, such systems natively support only a category-based access to the contents (often called "articles") of the built Web sites. Navigation between similar or semantically related contents belonging to different categories has to be supported by means of links expressly defined and kept up-to-date by the developer or content editor of the site. Our approach, once integrated in a CMS, will support the automatic generation and update of such links.

The approach presented in this paper uses Latent Semantic Indexing (LSI) [11] to compute a dissimilarity measure between the pages of a Web site based on the content they present, and a graph-theoretic clustering algorithm [16] to identify clusters (groups) of pages having similar or related content. Links connecting a given page to others pages within the same cluster, i.e., our defined Semantic Navigation Maps, are dynamically injected into each page of the site by means of AJAX code [7]. In order to automate the process of semantic navigation map recovery and injection, and to facilitate the adoption of the approach, we have developed a prototype of a supporting tool as an Eclipse plug-in. The approach and the tool have been applied with success in a case study, also presented in this paper, that proved their feasibility and validity.

This paper is an extension of the work presented in [36] and, compared to it, this paper provides the following new contributions:

- A refined version and a more detailed description of the recovery process.
- A detailed description of the tool support, with details on its architecture and implementation technologies.
- An extended version of the case study, which now involves three real-world Web sites.
- A discussion of the possible applications of the approach.

The rest of the paper is organized as follows: Sect. 2 discusses a number of works related to Web systems evolution, particularly to navigation restructuring and to techniques applied in our approach. Section 3 describes the process to recover semantic relations among the contents of a Web site and to enhance its navigation structure with our defined semantic navigation maps. Section 4 briefly presents a prototype of a tool developed to support the approach. Section 5 reports the results from a case study conducted on three real-world Web sites with the purpose of validating the approach and assessing its feasibility and advantages. Finally, Sect. 6 concludes the paper by providing some final remarks and describing avenues for future work.

2 Related work

In the last decade, the problem of defining methods and tools for the analysis and evolution of Web systems, and in particular their navigation structure, has been extensively studied [1,4,8,13,15,32,33]. Specific forums have also been created to provide researchers and practitioners with venues for discussing issues and propose solutions on the disciplined evolution of Web-based systems [41]. We synthesize and compare some of these works to ours.

A tool for analyzing the navigation structure of a Web site, its evolution during the time, and for identifying possible restructuring operations to improve navigability has been proposed by Ricca and Tonella in [32]. While supporting the analysis and restructuring of Web sites, the ReWeb tool does not support either the automatic application of the proposed changes or the restructuring of the navigation structure of the site on a semantic base. Antoniol et al. in [1] propose a methodology for reengineering a static Web site. The recovered model is based on the Relationship Management Data Model (RMDM) and the ER+ Model proposed within the Relationship Management Methodology (RMM). The methodology also enables the restructuring of the navigation structure of a Web site, but differently from our approach, the restructuring is neither automatic nor based on page content similarity.

Bernardi et al. have recently proposed REUWA [2], a process and a supporting tool for the semi-automatic recovery of user-centered conceptual models from existing Web applications, according to the Ubiquitous Web Applications (UWA) design methodology. This approach also supports the evolution tasks on the navigation structure of a Web site, but at current time changes have to be applied manually and no support for recovering links between pages with similar content is provided.

Other authors have proposed approaches for the model-based evolution of Web applications and, in particular, of their navigation models. Garrido et al. in [17] introduced Web Model Refactorings as behavior preserving transformations for the navigation and presentation models of a Web application aimed at improving its design and external quality. A catalog of refactorings to improve the quality of a navigation model has also been proposed by Cabot et al. in [5]. Both these proposals have not yet a tool support to enable the automatic implementation of the model refactorings into the analyzed application and, in our knowledge, none of the proposed refactorings deals with contents' similarity. Finally, Lowe and Kong [27] proposed NavOptim, an approach to the evaluation and improvement of Web sites' navigational structures based on the optimization of a navigational effort metric which considers the semantic cohesion between pages and task cases. Semantic cohesion (similarity) between the contents of the pages and task cases is calculated using semantic vector spaces but no automatic restructuring of the Web site is supported.

Methods and tools for the analysis and evolution of Web sites rely on several base techniques. These include clustering algorithms, refactoring, information retrieval and clone detection techniques, etc. As it will be described in detail later on in this paper, the proposed approach uses as base techniques LSI, a well-known information retrieval technique, and a graph-theoretic clustering algorithm. LSI is usually used to abstract concepts from texts (and thus from any document in which a text can be found) and compute a similarity measure between texts, based on their semantics. In our approach, we use LSI to compute a similarity measure between the pages of a Web site, thus identifying pages with similar or semantically related content.

LSI has been widely adopted in different application domains and for different purposes [9,24,25,28,29]. For example, Kuhn et al. [24] described an approach to group software artifacts using LSI. The approach is language-independent and tries to group source code containing similar terms in the comments. Different levels of abstraction to understand the semantics of the code (i.e., methods and classes) are considered. A method based on LSI to compare German literature texts is proposed in [29]. Indeed, the author evaluates whether texts by the same author are alike and can be distinguished from the ones by other authors. Texts by the same author are more alike and tend to form separate clusters. The author also observed that LSI separates prose and poetry texts in two separate clusters. In our work we use LSI to compute a dissimilarity measure between the pages of a Web site based on the content they show to the final user.

Clustering is a topic analysis technique in the maintenance and evolution of Web applications that is aimed at gathering the entities composing the software system (at different levels of abstraction) into meaningful and independent groups. Indeed, a large number of analysis and reverse engineering methods proposed in the literature, including some of those cited above, are based on clustering algorithms [2,9,12,33,34]. For example, different authors have used clustering algorithms to identify Web pages showing similar content and/or having similar HTML structure. Ricca and Tonella [33] enhance the approach proposed by Di Lucca et al. in [12] (a pairs of pages are clones if the Levenshtein edit distance [26] between the strings encoding the HTML page structures is zero) using a hierarchical clustering algorithm to identify clusters of duplicated or similar static pages that could be generalized into a dynamic Web page. Differently from the approach proposed in [12], the distance of cloned pages belonging to the same cluster is not zero. Similarly, in [37], the authors propose a semiautomatic approach based on an agglomerative hierarchical clustering algorithm to identify and align static HTML pages having the same structure and content expressed in different languages. The aligned multilingual pages are then merged into MLHTML pages. De Lucia et al. [10] also propose a semiautomatic approach based on the Levenshtein edit distance to compute the similarity of two pages at the structural, content, and scripting code levels. Clones are characterized by a similarity threshold

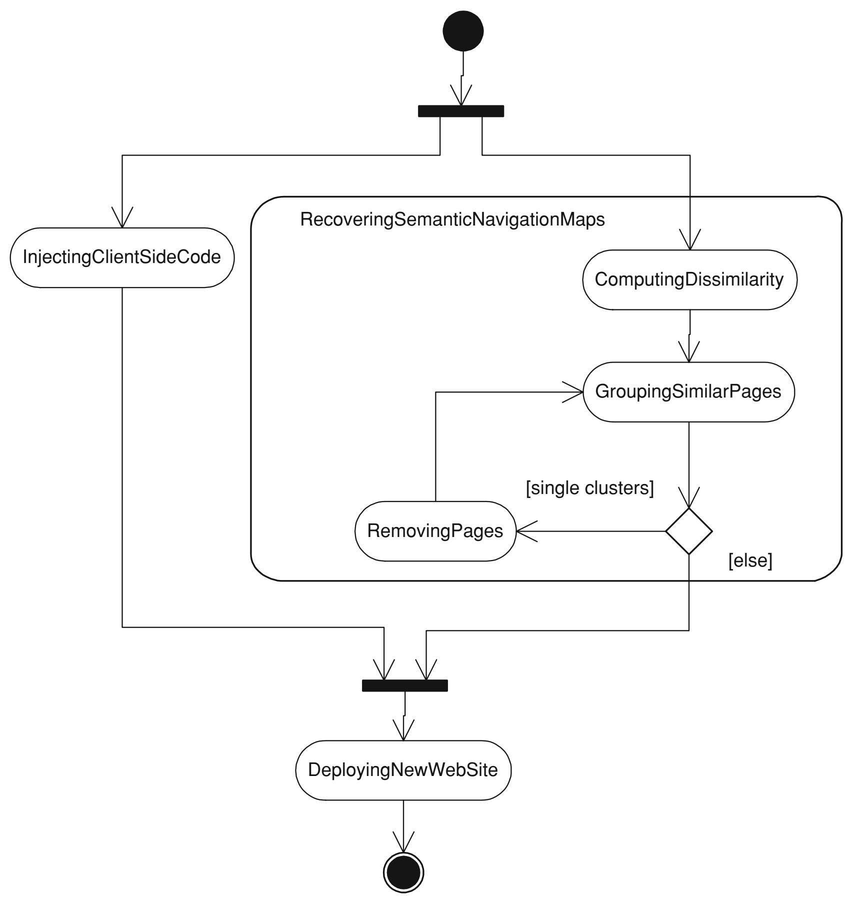


Fig. 1 The overall recovery and injection process

that ranges from 0%, for completely different pages, up to 100%, for identical pages. An approach based on a general process that first compares pages at the structural level (i.e., the Levenshtein edit distance) and then groups them using a competitive clustering algorithm (i.e., Winner Takes All) is proposed by De Lucia et al. in [8]. Ricca et al. propose in [34] an approach to Web sites understanding based on clustering of client-side HTML pages with similar content. In this work the authors first apply Natural Language Processing (NLP) techniques to associate each page with a set of keywords characterizing it and then they use a hierarchical clustering algorithm to group pages having common keywords and, thus, related content. Keywords are weighted so that more specific and relevant keywords receive a higher score. While we also adopt a clustering algorithm to group pages with similar content, we compute similarity between pages using LSI instead of NLP and keywords identification and weighting.

In [9], a comparison among clustering algorithms to identify similar pages at the content level is presented. In this work, three variants of the agglomerative clustering algorithm, i.e., a divisive clustering algorithm, k-means, and a competitive clustering algorithm, have been considered. The study reveals that the investigated clustering algorithms generally produce comparable results. To compare pages, an LSI-based similarity measure is used.

3 The process

The process to recover semantic navigation maps from a Web site is schematically represented by the UML activity diagram depicted in Fig. 1. In this diagram, rounded rectangles represent process phases and subphases. Overall, the process consists of three main phases: *RecoveringSemanticNavigationMaps, InjectingClientSideCode* and *DeployingNewWebSite*. The RecoveringSemanticNavigationMaps phase groups pages with similar or related content into clusters. This phase is executed only once and has to be repeated only when the content of the Web site changes because of the addition or the removal of pages, or because of the modification of the content of some pages. The InjectingClientSiteCode extends each client-side page of the site with the AJAX code enabling the runtime querying (via the interaction with a servlet on the server machine) the database of the identified clusters and the displaying of the semantic navigation map associated to each page. Similar to the previous phase of the process, this phase is executed only once and has to be repeated only for the new pages added to the site. The AJAX code injected in each page is instead executed at runtime every time the page is displayed. The DeployingNewWebSite phase consists in deploying the enhanced version of the site, i.e., the client-side pages extended with the AJAX code and the servlet used to retrieve data on the recovered clusters of similar pages.

In the following subsections we describe more in detail each of the phases and subphases of the process.

3.1 RecoveringSemanticNavigationMaps

The phase RecoveringSemanticNavigationMaps is composed of three subphases: *ComputingDissimilarity*,*Grouping-SimilarPages* and *RemovingPages*. ComputingDissimilarity extracts the textual content of each page (i.e., the text that is presented to a user) of the analyzed Web site and then computes the dissimilarity between any pairs of pages using a measure based on Latent Semantic Indexing (LSI) [11]. A dissimilarity matrix is produced as output of this activity. This matrix is used by the subphase of GroupingSimilarPages to identify groups of pages with similar or related content. The pages included in single-clusters, i.e., clusters containing only one page, are discarded by the RemovingPages subphase and are not considered in the following iterations of the process. The subphases GroupingSimilarPages and RemovingPages are repeated until no single-clusters remain. Details on these subphases are provided in the following.

3.1.1 ComputingDissimilarity

This phase is composed of four sequential subphases (see Fig. 2). ExtractingPageContent extracts the textual content of the client-side HTML pages of the given Web site. The extracted content has then to undergo a normalization subphase (i.e., *NormalizingContent*) in which non-textual tokens are eliminated (i.e., operators, special symbols, numbers, etc.), terms composed of two or more words are split (e.g., "mail_address" is turned into "mail" and "address") and terms with a length less than three characters are not considered. A stemming algorithm is also used to reduce inflected (or sometimes derived) terms to their stem. Finally, all the terms contained in a stop word list are removed. The terms within the stop word list should be selected according to their relevance for the content domain of the considered Web site. Indeed, irrelevant terms should be inserted in that list.

BuildingConceptSpace is in charge of computing the concept space of a Web site on its normalized content by adopting LSI. This technique has been originally developed to overcome the synonymy and polysemy problem occurring with the Vector Space Model (VSM) [20]. In fact, LSI explicitly considers dependencies among terms and among documents (corresponding to Web pages in our case), in addition to the associations between terms and documents. This technique

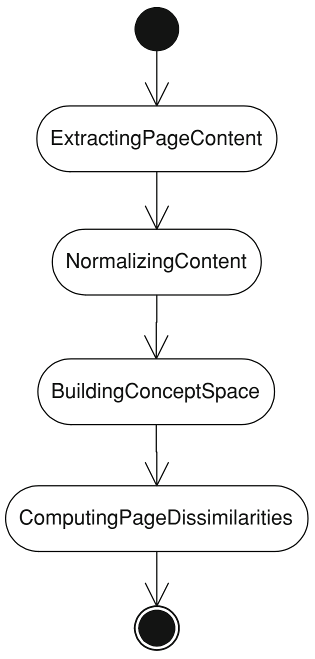


Fig. 2 The ComputingDissimilarity's subphases

assumes that there is a latent structure in word usage that is partially obscured by variability in word choice.

LSI is applied on a term-by-content matrix A, which is built on the normalized content of the considered Web site. In particular, this matrix is $m \times n$, where m is the overall number of different terms appearing in the pages of the site and n is the number of considered pages. An entry $a_{i,j}$ of the term-by-content matrix A represents a measure of the weight of the ith term in the jth page. To derive the content latent structure of a Web site, we apply on this matrix a Singular Value Decomposition (SVD) [11]. Using this technique the matrix A (having rank r) can be decomposed in the product of three matrices, $T \cdot S \cdot D^T$, where S is an $r \times r$ diagonal matrix of singular values and T and D have orthogonal columns. SVD also provides a simple strategy for optimal approximate fit using smaller matrices and using only a subset of k concepts corresponding to the largest singular values in S.

The selection of a "good" value of k (i.e., the singular values of the dimensionality reduction of the latent structure) is an open issue. Guidelines to select the suitable value of k have also been proposed in the past, e.g., percentage of number of terms, fixed number of factors, etc. [38]. In our approach, we calculate the number of singular values according to the Guttman–Kaiser criterion [19,21]. This criterion considers the diagonal matrix S of the singular values, which are a kind of eigenvalue, of Ain descending order. The value of k is the number of singular values in S more than 1. The rationale for adopting this criterion relies on the fact that it does not require any human intervention, thus fully automating the usage of LSI.

Terms and pages could be graphically represented by vectors in the k-dimensional space of the underlying concepts of a Web site. In our approach the rows of the reduced matrices of singular vectors are taken as coordinates of points representing the pages in a k-dimensional space. To build the dissimilarity matrix of a Web site, we first compute the cosine between all the pairs of vectors representing the pages in the k-dimensional space. The cosine value ranges from-1 (when the two pages have a different semantics) to 1 (when the semantics is the same).

The computation of the dissimilarity matrix requires that the dissimilarities among all the pairs of pages have to be computed. Indeed, the dissimilarity between the pairs of pages is computed in the subphase ComputingPageDissimilarities normalizing the cosine similarity measure from 0 (when the semantics is the same) to 1 (when they have a different semantics). Hence, given two pages p_1 and p_2, the dissimilarity between these two pages is defined as:

$$d_{\mathrm{lsi}}(p_1, p_2) = \frac{1 - \cos(Vp_1, Vp_2)}{\max_{\forall Vp_i,\, Vp_j \in W}(1 - \cos(Vp_i, Vp_j))}$$

where Vp_1 and Vp_2 are the vectors corresponding to the pages p_1 and p_2, respectively, in the space W of the content of the given Web site. Let us note that this measure cannot be considered a distance as it does not obey the triangle inequality rule. However, this does not influence the possibility of using clustering algorithms as Oudshoff et al. show in [30]. Let us note that the rationale for adopting the LSI technique depends on the fact that it has been successfully employed on different applications domains to identify semantic dependences among different entities (e.g., literature texts, software artifacts, source code, Web pages, etc.) [9,24,25,28,29].

3.1.2 GroupingSimilarPages

This subphase uses a graph-theoretic clustering algorithm [16] to group pages that are similar at the content level according to the dissimilarity measure described above. Generally, a graph-theoretic clustering algorithm takes as input an undirected graph and then constructs a Minimal Spanning Tree (MST). Clusters are identified pruning the edges of the MST with a weight larger than a given threshold. Nodes within each tree of the obtained forest are included in a cluster.

The graph-theoretic clustering algorithm is used on the strongly connected graph corresponding to the dissimilarity matrix computed in the phase ComputingDissimilarity. Nodes represent Web pages and edge weights are the dissimilarity measures between the pairs of pages of the built MST. Clusters are identified by pruning the edges with weights less than a given threshold. In our case this threshold is computed as the arithmetic mean of the MST edge weights. We use the arithmetic mean by aiming at fully automating the clustering process. However, we are conscious that different pruning thresholds may produce better results. Future work is needed to investigate this concern.

In case the graph-theoretic clustering algorithm identifies single-clusters, the subphase RemovingPages is executed. This subphase is in charge of removing from the dataset the pages that the used algorithm included in single-clusters. The so obtained dataset is then provided again as input to the GroupingSimilarPages phase. The phases GroupingSimilarPages and RemovingPages are automatically iterated until no single-clusters are identified. This enabled us to improve the overall quality of the clustering process. Furthermore, the overall quality of the clustering process could be improved manually by refining the clusters automatically identified by this phase. All the pages of a given Web site may be involved in this activity.

Figure 3 shows an example of the clustering results obtained using the graph-theoretic clustering algorithm on 23 pages of the Web site of the National Gallery of London (i.e., one of the Web sites used as case study). In particular, this figure shows the built MST and the clusters identified using 0.33 as pruning threshold. The clustering algorithm identified nine different groups of pages with similar or related content. Among the identified clusters, six were single-clusters. Note that neither the names nor the titles of the pages are reported in the figure for readability reasons. Edge weights less than the threshold values are not shown as well.

3.1.3 RemovingPages

As mentioned above, the subphase RemovingPages is executed in case the clustering algorithm identifies one or more single clusters. This phase is in charge of removing, from the dissimilarity matrix, the rows and the columns corresponding to the pages that have been placed within single-clusters. The obtained matrix is then provided as input to GroupingSimilarPages. RemovingPages is executed until no single-clusters are identified by the clustering algorithm. The motivation for removing the pages of the single-clusters lies in the fact that they should have a low level of semantic similarity with the remaining pages.

3.2 InjectingClientSideCode

This phase actually enhances the navigation structure of the considered Web site by introducing a semantic navigation map within each of its pages. Indeed, each page is extended to dynamically include a set of links toward the pages placed by the clustering process in the same cluster. To this end, we use AJAX and DOM technologies to, respectively, query the server that maintains the results of the clustering process and to display the semantic navigation maps within the pages of the Web site.

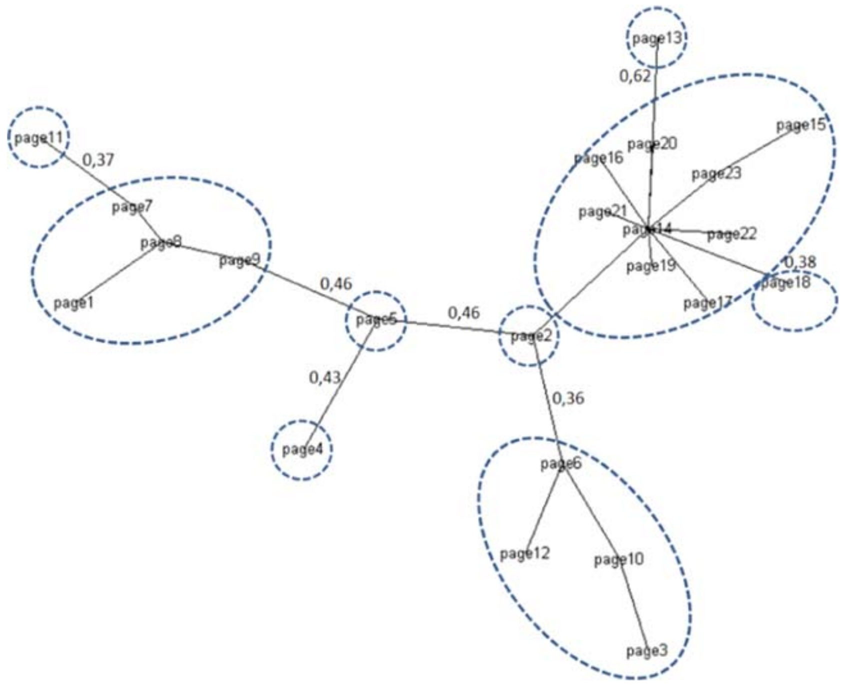


Fig. 3 Example of clusters of pages obtained for the NationalGallery.org.uk case study

Fig. 4 The Javascript code used to get data on a page cluster

```
function getCluster(divBlock,pageCode){
  mapList = divBlock;
  var req = newXMLHttpRequest();

  req.onreadystatechange = getReadyStateHandler(req, displayMap);
  req.open("POST","<%=request.getContextPath()%>/servlet/retrieveTrackingMap", true);
  req.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
  req.send("code=" + pageCode);
}
```

Fig. 5 The Javascript code used to visualize a semantic navigation map

```
function displayMap(rdftXML){
  var rdfs = rdftXML.getElementsByTagName("rdf")[0];
  var items = rdfs.getElementsByTagName("item");
  mapList.innerHTML = '';
  var code = '';
  for (var i = 0; i < items.length; i++){
    code +=
      items[i].getElementsByTagName("rank")[0].
      firstChild.nodeValue + "<a href=\'" +
      items[i].getElementsByTagName("link")[0].
      firstChild.nodeValue +"\'>" + "<b>" +
      items[i].getElementsByTagName("title")[0].
      firstChild.nodeValue +"</b>" + "</a><br/>" +
      items[i].getElementsByTagName("desc")[0].
      firstChild.nodeValue;
  }
  code = createMultiTab(code, 8);
  mapList.innerHTML = code;
  activateMultiTab(mapList);
}
```

For the first objective we use a Javascript function that retrieves the list of pages within the cluster to which the given page belongs. Figure 4 shows the Javascript code used to get this list. In particular, the Javascript function establishes a connection with the servlet *retrieveTrackingMap* to get the list of pages to be visualized within the semantic navigation map.

For the second purpose, we use a different Javascript function (see Fig. 5) to interpret the server answer and to properly display the map of links in the considered Web page by injecting HTML code (see Fig. 6) into the DOM of the page. To inject the HTML code we add the DIV tag at the end of each page. This code is different for each page of the Web site as it includes the name of the page for which it has been generated. For example, the code shown in Fig. 6 regards the page *michelangelo.html*.

At run-time, a Javascript code is used to overlap the semantic navigation map on the original Web page. This has the effect of displaying the map on the right-hand side of the page. However, the end user can place the map wherever

Fig. 6 The HTML code injected in the Web pages

```
<!-- Begin Semantic Navigation Map -->
<div class="LocalTrackingMap">
        <div id="localmap"></div>
        <script src="script/localTrackingMap.js" type="text/javascript"></script>
        <script>getCluster("localmap","collection\artist\michelangelo.html");</script>
</div>
<!-- End Semantic Navigation Map -->
```

s/he wants (see Figs. 11, 12) using drag and drop functionalities. The semantic navigation map also preserves its position when the user scrolls the page. Furthermore, if necessary, the map can be folded. Also, note that the semantic navigation map will provide a multi-tab visualization in case it contains a number of links larger than 8 (see Fig. 12).

All the functions required for managing the communication with the server, parsing messages from it, and visualizing the semantic navigation maps, have been collected in a Javascript library (i.e., localTrackingMap.js), in order to be reused in each page of the evolved Web site.

Note that the HTML pages of a given Web site are modified once for all. This is possible because the clusters are independently detected (see Fig. 1) and are dynamically obtained querying the server. Furthermore, the identification of pages with similar or related content should be performed when the content of existing pages is modified or new pages are deployed in the Web site. For consistency reasons, the identification of similar pages should also be repeated when pages are removed.

3.3 Deploying the enhanced web site

In the DeployingNewWebSite phase the enhanced pages and the data on the recovered clusters are deployed on the server. The deployed pages require a server component that retrieves data on the semantic navigation map recovered and associated to each client-side page. To enable the communication between each enhanced client-side page and the server where the corresponding cluster is stored, we have developed a servlet (i.e., retrieveTrackingMap). The developer has to deploy this servlet only once. Finally, when the enhanced pages and the servlet have been deployed, the Web site is enabled to be accessed by the users.

In this phase, the developer should also perform testing to find possible faults of the new version of the Web site. Currently, our tool prototype does not provide any specific support for testing. Future work will also go in this direction. To this end a Web browser extension could be developed to record, edit and debug test cases.

4 Tool prototype

To support all the phases of the proposed approach, we have implemented a prototype of a supporting tool as an Eclipse plug-in. In the following subsections we first present the logical architecture of the tool and then describe how it supports the different phases of the process.

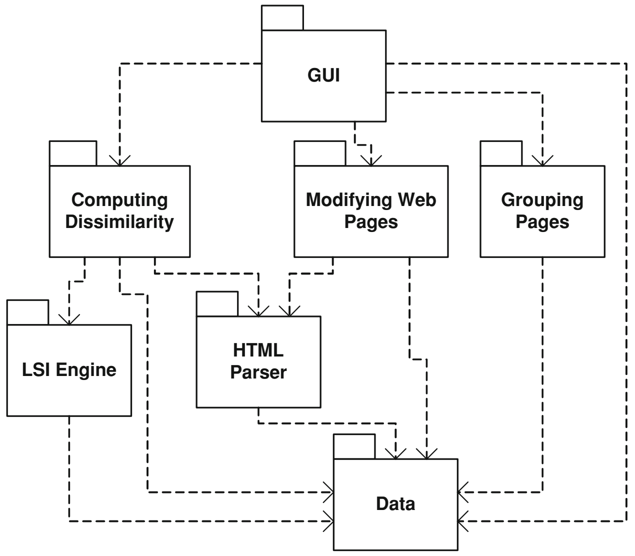


Fig. 7 The architecture of the tool prototype

4.1 Tool prototype architecture

Figure 7 shows the layered architecture (i.e., the logical architecture) of the system prototype by means of a UML Package Diagram. The *Data* component contains the pages of the Web site and all the intermediate data produced by the process phases. The *Computing Dissimilarity* component uses the *HTML Parser* component to extract the content of the HTML client-side pages. HTML Parser integrates an open source HTML parser written in Java (HTMLParser ver. 1.6), available under the GPL license at sourceforge.net.[2]

The content extracted from each page of the site is stored in the local file system (i.e., the Eclipse workspace) to avoid repeating this operation for the same page more than once. The Computing Dissimilarity module uses the *LSI Engine* component to compute the dissimilarity matrix of the page content. To this end this component integrates an R[3]

[2] HTMLParser ver. 1.6 can be downloaded from http://sourceforge.net/projects/htmlparser.

[3] R is a free software environment for statistical computing and graphics. It is worth noting that there is a very proficient and active community that evolves the environment and the majority of the available libraries are available under GPL license.

implementation of the LSI information retrieval technique. This implementation is available under GPL license from http://www.cran.r-project.org. The rationale for using R lies in the fact that it is open source and a huge and very active community works to improve it.

The dissimilarity matrix produced by the Computing Dissimilarity component is stored in the Eclipse workspace. This choice is motivated by the fact that the computation of this matrix is expensive and it is performed only in case new pages are added or remove from the original Web site (see Sect. 3.2). For example, the computation of the dissimilarity matrix for the National Gallery Web site, which included 2017 pages, took about three hours using a laptop equipped with a 1.5 GHz Intel Centrino with 1.5 GB of RAM, a 60 GB Hard Disk and Windows XP Professional SP 3 as operating system.

The dissimilarity matrix is successively provided as input to the *Grouping Pages* component. This component is in charge of detecting pages that are similar. To this end an R implementation of the graph-theoretic clustering algorithm (available under GPL license from cran.r-project.org) has been integrated. Grouping Pages also removes from the dissimilarity matrix rows and columns corresponding to the pages within single-clusters. Also, note that the R implementations of the LSI engine and the clustering algorithm are all integrated within the system prototype using Rserve,[4] a TCP/IP server allowing programs to use R facilities. Despite the availability of simpler methods to integrate R software components with Java code, we decide to use Rserve to eventually distribute the computation on different nodes on the Web. This was due to the fact that the time needed to execute the tool on large-sized Web sites could be considerable.

Modifying Web Pages injects the needed HTML and Javascript code into the pages of the Web application to enable the communication with the server and to display the semantic navigation maps. The point where the HTML and Javascript code has to be injected is identified using the HTML Parser component. The *GUI* component enables the pages selection of the Web site and to display the groups of similar pages detected by the tool (see Fig. 9). Automatically identified clusters can be manually refined by adding/removing pages to/from them, thus improving the overall quality of the similar page groups.

4.2 Using the Eclipse plug-in

The developed Eclipse plug-in fully supports all the phases of the process depicted in Fig. 1. As a first operation, the plug-in requires the creation of an Eclipse project. To this end, the plug-in proposes a wizard, which allows specifying the name of the project and its workspace. Successively, the developer has to select the Web site whose navigation structure has to be enhanced. This is possible by right-clicking the project within the *Package Explorer view* and choosing *Select Source Directory* within the menu *semantic navigation map* (see Fig. 8). The plug-in will provide a window for choosing the path of the folder where the pages of the original Web site are stored. Note that these pages have to be stored on the file system of the personal computer where the plug-in is installed. The plug-in could be extended to work on Web pages maintained on remote servers. This is a further direction to extend the proposed supporting tool.

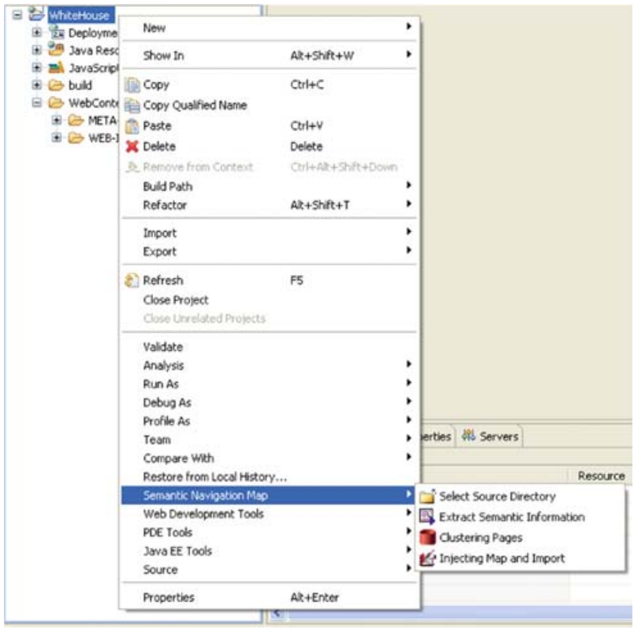
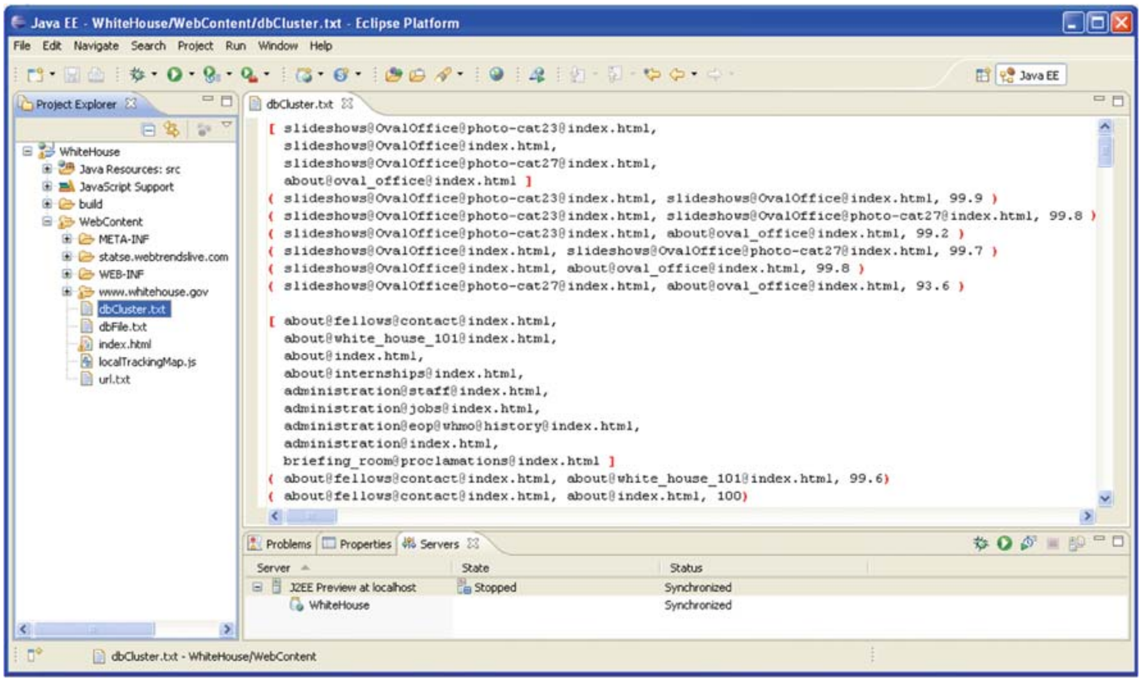


Fig. 8 Semantic navigation map menu

The clustering process starts by right-clicking the project and choosing *Clustering Pages* within the menu *semantic navigation map* (see Fig. 8). The plug-in will produce a file (i.e., dbcluster.txt) containing all the automatically identified clusters. Indeed, it also contains all the pairs of pages composing each cluster and their similarity level (i.e., the cosine between the vectors of the pages in the content space). To improve the overall quality of the clustering process, the plug-in enables by manually refining the automatically identified clusters. If needed, the developer can refine a cluster by modifying the set of included pages and their similarity levels. Figure 9 shows the report of the clustering results showed within the plug-in. Note that in the current version of the plug-in the developer can only edit the report containing the identified clusters by simply using the plug-in text editor. The plug-in does not perform any check on the correctness of the modifications made on the report.

The developer uses then the Eclipse plug-in to extend the pages of the Web site by automatically introducing the semantic navigation maps (see Fig. 8). Successively, the new

[4] Rserve can be downloaded from http://www.cran.r-project.org.



```
[ slideshows@OvalOffice@photo-cat23@index.html,
  slideshows@OvalOffice@index.html,
  slideshows@OvalOffice@photo-cat27@index.html,
  about@oval_office@index.html ]
( slideshows@OvalOffice@photo-cat23@index.html, slideshows@OvalOffice@index.html, 99.9 )
( slideshows@OvalOffice@photo-cat23@index.html, slideshows@OvalOffice@photo-cat27@index.html, 99.8 )
( slideshows@OvalOffice@photo-cat23@index.html, about@oval_office@index.html, 99.2 )
( slideshows@OvalOffice@index.html, slideshows@OvalOffice@photo-cat27@index.html, 99.7 )
( slideshows@OvalOffice@index.html, about@oval_office@index.html, 99.8 )
( slideshows@OvalOffice@photo-cat27@index.html, about@oval_office@index.html, 93.6 )

[ about@fellows@contact@index.html,
  about@white_house_101@index.html,
  about@index.html,
  about@internships@index.html,
  administration@staff@index.html,
  administration@jobs@index.html,
  administration@eop@whmo@history@index.html,
  administration@index.html,
  briefing_room@proclamations@index.html ]
( about@fellows@contact@index.html, about@white_house_101@index.html, 99.6)
( about@fellows@contact@index.html, about@index.html, 100)
```

Server	State	Status
J2EE Preview at localhost	Stopped	Synchronized
WhiteHouse		Synchronized

Fig. 9 Improving groups of similar pages

pages and the Javascript library will be automatically added to the project workspace. Finally, the plug-in enables the deployment of the enhanced Web site on a suitable Web server.

5 Case study

The approach and the tool prototype have been assessed on three real-world Web sites: the National Gallery of London (NGL),[5] Play Shakespeare (PS),[6] and the White House (WH).[7] In the following subsections we present and discuss the results obtained from this case study.

5.1 Context

Some pages of NGL presented a navigation menu (see for example Fig. 11) to promote a quick access to the Web site content. As the careful reader may object, in the case in point, the recovered semantic navigation maps may result in duplicate navigation structures (the navigation menu) and may have biased the obtained results. In order to address these concerns, we have investigated the effectiveness of the approach on the additional Web sites PS and WH. The rationale for choosing PS and WH relies on the fact that their pages lacked a navigational menu that could positively affect the results of the analysis.

To collect the pages of the selected Web sites, a freeware dumper can be used. In particular, we used HTTrack Website Copier.[8] The motivation for performing the dump is that the source HTML files were not available for the download. Regarding NGL the dump was executed on June 9th, 2008. In particular, HTTrack Website Copier downloaded 6573 HTML pages using the index page as starting page and following the hyperlinks connecting the pages stored until the sixth level of depth was reached in the folder hierarchy of the Web site. The mirror of NGL has been successively analyzed to prune duplicated HTML pages created by the dumper, documents currently not considered by the approach (e.g., PDF and Word files), and multimedia objects (e.g., JPG images and flash animations). Regarding the HTML pages, we limited the analysis to the ones presenting information and content on the museum entire *Permanent Collection* and *Long Term Loans*. The pages of the works of art shown during the years in the gallery have also been considered (i.e., the pages within the *Exhibition* section of the NGL Web site). The total number of HTML pages selected and considered in the presented case study is 2017.

The dump of the PS Web site was executed on January 16th, 2009. The dumper downloaded 5570 pages starting from the index page and following all the hyperlinks connecting the pages until the fifth level of depth. Similarly to

[5] http://www.nationalgallery.org.uk.

[6] http://www.playshakespeare.com.

[7] http://www.whitehouse.gov.

[8] Available at http://www.httrack.com.

Table 1 Descriptive statistics of the analyzed Web sites

	NGL	PS	WH
Number of analyzed pages	2017	1770	313
Number of identified clusters	243	141	17
Number of iterations	5	4	5
Number of pages within single-clusters	1387	639	219
Percentage of pages placed in single-clusters	68%	36%	70%
Number of pages within clusters containing at least two pages	630	1131	94
Number of pages within the largest cluster	32	436	25
Mean number of pages within the clusters	2.6	8.0	5.5
Number of characters within the analyzed pages	3.460K	13.716K	1.853K
k-value	1719	1503	273

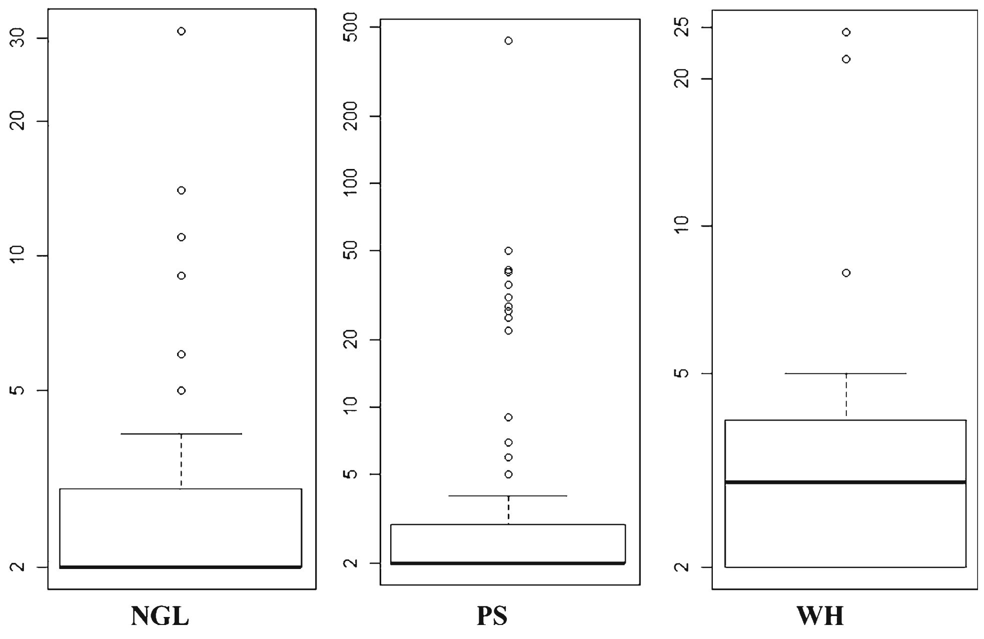


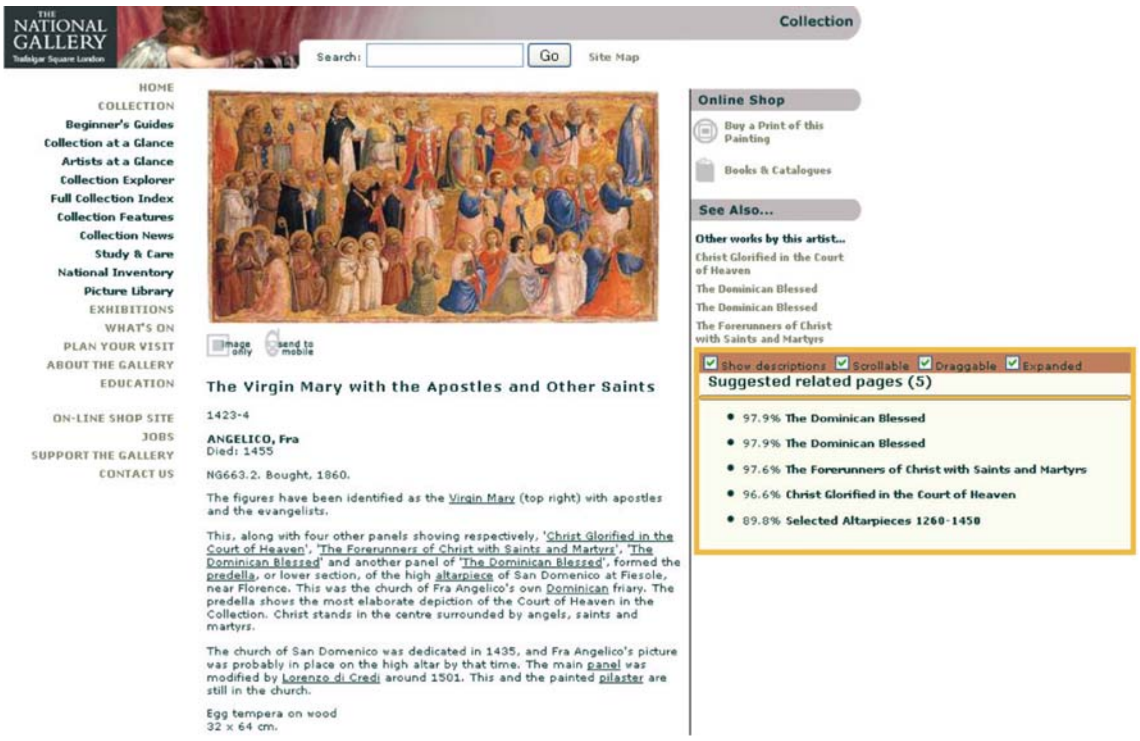
Fig. 10 Cardinality distribution of the clusters for the analyzed Web sites

the NGL Web site, the downloaded pages have been analyzed to prune duplicated and meaningless pages, irrelevant documents and multimedia objects. As a consequence of the pruning, the number of HTML pages was 1770.

Regarding the WH Web site, the dump was performed on January 22nd, 2009. The number of downloaded pages was 424 following all the hyperlinks connecting the index page until the fifth depth level. Also, in this case a pruning phase has been performed to remove duplicated and meaningless pages, irrelevant documents and multimedia objects that we do not treat in the approach. Once the pruning was performed the number of remaining pages was 313.

5.2 Results

For each analyzed Web site, the selected pages were then provided as input to the plug-in for the following clustering and semantic maps recovery and injection. In the case on NGA, the 2017 pages were grouped into 243 different clusters. The largest cluster contained 32 pages and included pages from the collection *Scientific Instruments and Inventions from the Past* of the gallery. On the other hand, the mean number of pages within the clusters was 2.6. Regarding the PS Web site, 141 were the identified clusters, with an average of pages per cluster equals to 8 and the large cluster counting 436 pages. For the WH Web site, the 313 analyzed pages were grouped into 17 clusters having a mean of 5.5 pages and the large cluster composed of 25 pages. These and other descriptive statistics on the case study are summarized in Table 1. The cardinality distribution of the clusters containing at least two pages for each analyzed Web site is graphically summarized by the boxplots shown in Fig. 10. Note that the median for each Web site is either 2 or 3. Overall, even clusters containing 2–3 pages are useful and meaningful, as each of them identifies a set of 2–3 pages having similar and correlated content.


THE NATIONAL GALLERY Trafalgar Square London

Collection

Search: Go Site Map

HOME
COLLECTION
Beginner's Guides
Collection at a Glance
Artists at a Glance
Collection Explorer
Full Collection Index
Collection Features
Collection News
Study & Care
National Inventory
Picture Library
EXHIBITIONS
WHAT'S ON
PLAN YOUR VISIT
ABOUT THE GALLERY
EDUCATION

ON-LINE SHOP SITE
JOBS
SUPPORT THE GALLERY
CONTACT US

Image only
send to mobile

The Virgin Mary with the Apostles and Other Saints

1423-4

ANGELICO, Fra
Died: 1455

NG663.2. Bought, 1860.

The figures have been identified as the Virgin Mary (top right) with apostles and the evangelists.

This, along with four other panels showing respectively, 'Christ Glorified in the Court of Heaven', 'The Forerunners of Christ with Saints and Martyrs', 'The Dominican Blessed' and another panel of 'The Dominican Blessed', formed the predella, or lower section, of the high altarpiece of San Domenico at Fiesole, near Florence. This was the church of Fra Angelico's own Dominican friary. The predella shows the most elaborate depiction of the Court of Heaven in the Collection. Christ stands in the centre surrounded by angels, saints and martyrs.

The church of San Domenico was dedicated in 1435, and Fra Angelico's picture was probably in place on the high altar by that time. The main panel was modified by Lorenzo di Credi around 1501. This and the painted pilaster are still in the church.

Egg tempera on wood
32 x 64 cm.

Online Shop
Buy a Print of this Painting
Books & Catalogues

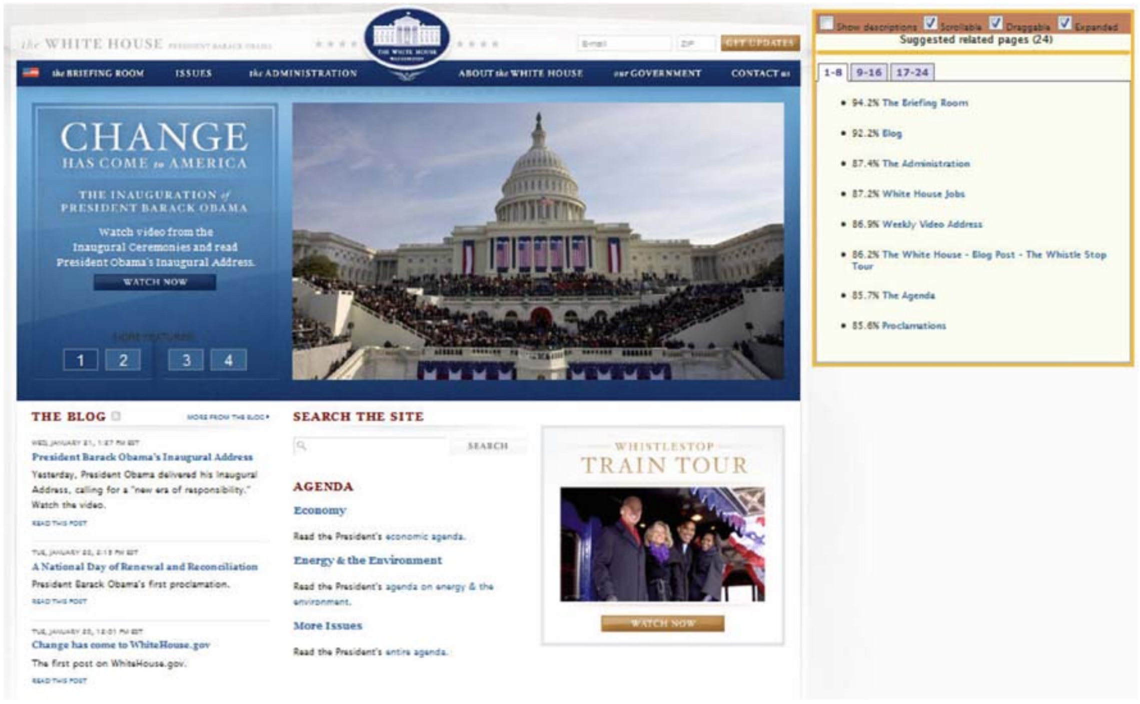
See Also...
Other works by this artist...
Christ Glorified in the Court of Heaven
The Dominican Blessed
The Dominican Blessed
The Forerunners of Christ with Saints and Martyrs

[✓ Show descriptions] [✓ Scrollable] [✓ Draggable] [✓ Expanded]

Suggested related pages (5)

- 97.9% The Dominican Blessed
- 97.9% The Dominican Blessed
- 97.6% The Forerunners of Christ with Saints and Martyrs
- 96.6% Christ Glorified in the Court of Heaven
- 89.8% Selected Altarpieces 1260-1450

Fig. 11 The enhanced version of a page from the National Gallery of London Web site


the WHITE HOUSE PRESIDENT BARACK OBAMA

E-mail ZIP GET UPDATES

the BRIEFING ROOM ISSUES the ADMINISTRATION ABOUT the WHITE HOUSE our GOVERNMENT CONTACT us

CHANGE HAS COME to AMERICA

THE INAUGURATION of PRESIDENT BARACK OBAMA

Watch video from the Inaugural Ceremonies and read President Obama's Inaugural Address.

WATCH NOW

1 2 3 4

THE BLOG MORE FROM THE BLOG

WED, JANUARY 21, 1:27 PM EST
President Barack Obama's Inaugural Address
Yesterday, President Obama delivered his Inaugural Address, calling for a "new era of responsibility." Watch the video.
READ THIS POST

TUE, JANUARY 20, 2:13 PM EST
A National Day of Renewal and Reconciliation
President Barack Obama's first proclamation.
READ THIS POST

TUE, JANUARY 20, 12:01 PM EST
Change has come to WhiteHouse.gov
The first post on WhiteHouse.gov.
READ THIS POST

SEARCH THE SITE

SEARCH

AGENDA

Economy
Read the President's economic agenda.

Energy & the Environment
Read the President's agenda on energy & the environment.

More Issues
Read the President's entire agenda.

WHISTLESTOP TRAIN TOUR
WATCH NOW

[ ] Show descriptions [✓ Scrollable] [✓ Draggable] [✓ Expanded]

Suggested related pages (24)

1-8 9-16 17-24

- 94.2% The Briefing Room
- 92.2% Blog
- 87.4% The Administration
- 87.2% White House Jobs
- 86.9% Weekly Video Address
- 86.2% The White House - Blog Post - The Whistle Stop Tour
- 85.7% The Agenda
- 85.6% Proclamations

Fig. 12 The enhanced version of a page from the White House Web site

The Eclipse plug-in was finally used to inject the semantic navigation map code into the pages of all the considered Web sites. Examples of the enhanced version of the pages obtained applying the approach on the Web sites NGL and WH are shown in Figs. 11 and 12, respectively. It is worth noting that the new Web sites have been deployed on a local machine. This was due to the fact that we assessed the approach and the system prototype on the local copy of the selected Web sites.

Let us also note that the pages shown in Figs. 11 and 12 differ from their original versions in the presence of the semantic navigation maps (see the right-hand side of the pages). Each semantic navigation map presents a set of links toward pages that have been found similar in content. Each link shows:

(i) the title of the target page,
(ii) the percentage of similarity with the current page obtained using LSI and
(iii) a description of the target page obtained from data in its description meta tag, if available.

5.3 Discussion

By analyzing the pages of the Collection section of the NGL Web site (i.e., the pages presenting information on the permanent collection and long term loans of the museum), we noted the presence of a navigation menu on the bottom right-hand side (see Fig. 11). In particular, given an artist and his/her work of art, the menu enables users to directly access the pages presenting other works of art of the same artist exhibited at the National Gallery of London. Additional links are also proposed to navigate and to access related pages. Since NGL presented a menu, we compared it with the recovered semantic navigation maps to assess the effectiveness of the approach. This comparison revealed that most of the links presented by the navigation menu were proposed by analogous links in the recovered semantic navigation map. As an example, if we compare the navigation menu and the navigation map of the page shown in Fig. 11 we can observe that they show basically the same links. The only difference we can notice is the link *Selected Altarpieces 1260-1450* that is present in the semantic navigation map and not in the navigation menu. This link allows accessing a page showing the works of art from different artists on the theme of altarpieces. On the other hand, we also noted that some pages of the Collection section presented a navigation menu with a number of links larger than the ones within semantic navigation maps. However, the maps of the majority of these pages present the same links as the corresponding navigation menu. This indicates that the links of the identified semantic navigation maps are correct.

Even on the PS Web site the plug-in produced interesting results. We observed that the majority of the links included in the semantic navigation maps connected pages with similar or semantically related content. For example, for this Web site, the larger identified cluster mainly contained all the pages presenting information regarding the characters of the Shakespeare's plays. Indeed, 327 were the pages within this cluster presenting information on the play characters. Overall, we can also observe that clusters containing 2 or 3 pages are useful and meaningful, as each of them identifies a set of 2 or 3 pages, out of the totality of analyzed pages, having similar or correlated content.

Due to the size of the considered Web sites we cannot analyze the completeness of the identified maps. However, some considerations can be made according to the descriptive statistics presented in Table 1. In particular, on PS and WH the average number of pages within the identified clusters seems correct. Also, the number of pages within clusters containing at least two pages enables us to believe that the obtained results could be appropriate. A further observation is motivated by the number of single-clusters identified by the Eclipse plug-in. This could bias the usefulness of the approach as single-clusters do not contribute to the enhancement of the navigational structure of a Web site. Hence, the larger is the percentage of pages placed in clusters containing at least two pages, the more is the usefulness of the approach. In fact, we observed that on the Web sites NGL and WH the percentage values of the single-clusters with respect the total number of analyzed pages (see Table 1) were 68 and 70%, respectively. On the other hand, a smaller percentage (i.e., 36%) was achieved on PS. Such a difference was due to the fact that PS presents pages richer in content compared to the other two Web sites.

Overall, the results obtained on the Web sites selected as case study suggest three considerations. The first concerns the subjective satisfaction of the users. This issue will be addressed in the future by conducting special designed investigations, e.g., survey questionnaires and interviews [22,39]. The latter two considerations regard correctness (links included in the navigation maps actually propose pages with content similar or related to the one showed by the considered page) and completeness (the list of proposed links includes most of the pages with actually related content) that will be discussed in the following subsection.

5.3.1 Assessing correctness and completeness

The correctness and the completeness of the approach have been assessed on the static pages of SRA (Student Route Analysis) a dynamic Web site implemented by one of the author. SRA was developed to provide the students in Politics Science at the University of Salerno with statistics and information on their academic carrier. SRA was composed of 380 files distributed in 45 folders according to a meaningful classification. Overall, the SRA Web site was composed of 24 html pages and 45 dynamic pages.

To quantitatively assess the produced results, we used two well-known metrics, namely precision and recall. The precision and recall have been used to assess the correctness and the completeness of the automatic identified clusters, respectively. In our case the precision is the number of actual pairs of similar pages identified by the tool over the total number of identified pairs, while the recall is the ratio between the number of actual pairs of similar pages identified by the tool over the total number of actual pairs of similar pages.

The obtained precision value was 0.89, while the recall was 0.57. This indicates that almost all the links included in each semantic navigation map are correct. Moreover, about two-third of the actual links between pairs of pages have been correctly detected. Although the achieved results are encouraging a further investigation is needed to further assess the effectiveness of our approach in terms of correctness and completeness.

6 Conclusion and future work

Our research is meant to automatically recover semantic relations between the pages of a Web site and build semantic navigation maps, accordingly. To this aim, we have defined a process that first computes the dissimilarity between Web pages using a measure based on Latent Semantic Indexing and then uses the computed dissimilarity to group pages showing similar or related content, by means of a graph-theoretic clustering algorithm. Finally, the navigation structure of the Web site is enhanced by adding links between pages within the same cluster. We defined the set of links added to a page to connect it to other pages of the same cluster as semantic navigation map. To automate the application of the approach, we have developed a supporting tool as an Eclipse plug-in. Both the approach and the tool have been validated in a case study involving three real-world Web sites.

When we have started this work, we considered the client-side HTML pages of the site as the elementary granules of information (a.k.a. nodes) to analyze, to index and to connect by means of hyperlinks, and assumed that the content (text) included in a page and showed to the final user is uniquely identified by the page URL. Therefore, the proposed process and the supporting tool are applicable to Web sites in which the content associated to a page neither depend on the user logged on, nor on other context variables. Nevertheless, it is not required that the contents of the site remain unchanged over the time.[9] Our approach is instead applicable independently from the technologies used server-side to produce the front-end of the application, provided that this front-end is represented by HTML pages.

Semantic navigation maps may be particularly useful when the navigation structure of the site is not found to be properly designed or when it has degraded during the Web site life time. However, even properly designed Web sites may benefit from the use of the semantic navigation maps. In fact, they represent an additional navigation structure that can beneficially complement those deriving from the navigation model defined for the site. Indeed, while this model is usually designed to satisfy specific navigation requirements (e.g., provide access to subsets of contents grouped by category or having some characteristic of interest for the user, etc.), our semantic navigation maps are intended to make explicit the latent semantic relations between the contents of the site and keep this relation up-to-date when these contents evolve.

Similar to Web site search engines such as Google Site Search (available at http://www.google.com/coop/cse) or FreeFind (available at http://www.freefind.com), semantic navigation maps represent a navigation structure built by analyzing and indexing the contents of the Web site based on their meaning. Differently from them, the index provided by a semantic navigation map is not the result of a search query executed by the user at run-time, but a navigation structure provided by default by the site to its users. More importantly, the navigation map is obtained considering as "input parameters" of the search query not a particular keyword or set of keywords, but the full text of the page the user is visiting. The same consideration applies if we compare the semantic navigation maps recovered by our approach to the results that can be obtained using recently appeared semantic Web search engines such as semaGER[10] and Cognition.[11] These search engines find semantic keywords and Web pages suitable, on a semantic base, to the search terms specified as input by the user. These engine, to the best of our knowledge, are not intended to analyze the pages of a given Web site and make manifest as links the semantic relations between them (which is what our approach does).

In the future, we plan to apply the approach on other Web sites, different in size and application domain, and to conduct controlled experiments to investigate the effectiveness of the recovered navigation maps. These experiments will aim at assessing whether the enhanced version of the sites better satisfies the users' expectations in terms of contents navigation and information access.

Future work will also be devoted to investigate the effect of adopting and combining different page similarity measures to group pages with similar or related content. The possibility of using different pruning thresholds of the edges of the adopted clustering algorithm will also be considered. The effect of using different singular values of the dimensionality reduction of the latent structure of a Web site will be investigated as well. Moreover, since we used the same stop word list for all the considered Web sites it would be useful to investigate how different stop word lists affect the overall quality on the automatically identified semantic navigation maps.

It will also be worth extending both the approach and the tool prototype to make them suitable for dynamic Web sites,

9 A similar limitation also affects search engines, which are able to index only the freely accessible (portion of) Web sites.

10 http://www.semager.com.

11 http://www.cognition.com.

i.e., Web sites for which the content showed into pages and the accessible pages vary depending on some context variable (the user profile, location, etc.) [8,10]. The approach and the tool will be also extended to analyze PDF and Word files. Features to completely customize the approach (e.g., using personalized stop word list, specifying different criterions for the pruning threshold of the used clustering algorithm) will also be implemented in the new version of the plug-in.

We are currently working on developing software components to be integrated in different and widely employed Web systems, e.g., CMSs, e-learning platforms, and e-commerce application frameworks. With the intent of satisfying the user's expectations and requirements, we will also investigate the possibility of adapting the navigation maps according to the user's profile and/or preferences. For example, pages could gain positions (a better score) in the semantic navigation maps in case the user has previously navigated them. Finally, we are considering the feasibility of extending our approach by implementing a browser plug-in that gives the user the possibility of analyzing and producing semantic navigation maps for a Web site of his/her choice.

References

1. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A.: Web site reengineering using RMM. In: Proceedings of the 2nd International Workshop on Web Site Evolution, pp. 9–16, Zurich, Switzerland (2000)
2. Bernardi, M., Di Lucca, G. A., and Distante, D.: Reverse engineering of web applications to abstract user-centered conceptual models. In: Proceedings of the 10th International Symposium on Web Site Evolution, pp. 55–64. IEEE Press (2008)
3. Boldyreff, C., Tonella, P.: Web site evolution. Special Issue J. Softw. Maintenance **6**(1-2), 1–4 (2004)
4. Boldyreff, C., and Kewish, R.: Reverse Engineering to Achieve Maintainable WWW Sites. In: Proceedings of the 8th IEEE Working Conference on Reverse Engineering, pp. 249–257, Stuttgart, Germany, IEEE CS Press (2001)
5. Cabot, J., Gómez, C.: A catalogue of refactorings for navigation models. In: Proceedings of the 8th International Conference on Web Engineering, pp. 75–85, Yorktown Heights, New York, IEEE CS Press (2008)
6. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing web sites. Comput. Netw. **33**(1–6), 137–157 (2000)
7. Cran, D., Pascarello, E., Darren J.: Ajax in Action. Manning Publications Co. (2005). ISBN: 1932394613
8. De Lucia, A., Scanniello, G., Tortora, G.: Identifying similar pages in web applications using a competitive clustering algorithm. J. Softw. Maintenance Evol. **19**(5), 281–296. Wiley, New York (2007)
9. De Lucia, A., Risi, M., Scanniello, G., Tortora, G.: Clustering algorithms and latent semantic indexing to identify similar pages in web applications. In: Proceedings of the 9th IEEE International Symposium on Web Site Evolution, pp. 65–72, Paris, France, IEEE CS Press (2007)
10. De Lucia, A., Francese, R., Scanniello, G., Tortora, G.: Identifying cloned navigational patterns in web applications. J. Web Eng. **5**(2), 150–174. Rinton Press (2006)
11. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inform. Sci. **41**, 391–407 (1990)
12. Di Lucca, G.A., Di Penta, M., Fasolino, A.R.: An approach to identify duplicated web pages. In: Proceedings of the 26th Annual International Computer Software and Application Conference, pp. 481–486, Oxford, UK, IEEE CS Press, (2002)
13. Di Lucca, G.A., Di Penta, M., Antoniol, G., Casazza, G.: An approach for reverse engineering of web-based applications. In: Proceedings of the 8th IEEE Working Conference on Reverse Engineering, pp. 231–240, Stuttgart, Germany, IEEE CS Press (2001)
14. Distante, D., Rossi, G., Canfora, G., Tilley, S.: A comprehensive design model for integrating business processes in web applications. Int. J. Web Eng. Technol. **2**(1), 43–72. Inderscience Publishers (2007)
15. Eichmann, D.: Evolving an engineered web. In: Proceedings of the International Workshop Web Site Evolution, pp. 12–16, Atlanta, GA (1999)
16. Flynn, P.J., Jain, A.K., Murty, M.N.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
17. Garrido, A., Rossi, G., Distante, D.: Model refactoring in web applications. In: Proceedings of the 9th International Symposium on Web Site Evolution, pp. 89-96, IEEE CS Press (2007)
18. Garzotto, F., Perrone, V.: On the acceptability of conceptual design models for web applications. In: Proceedings of Conceptual Modeling for Novel Application Domains—ER'03 Workshops (Chicago, US). LNCS, vol. 2814, pp. 92–104 (2003)
19. Guttman, L.: Some necessary conditions for common factor analysis. Psychometrika **19**, 149–161 (1954)
20. Harman, D.: Ranking algorithms. In: Information Retrieval: Data Structures and Algorithms, pp. 363–392. Prentice-Hall, Englewood Cliffs, NJ (1992)
21. Kaiser, H.F.: The application of electronic computers to factor analysis. Educ. Psychol. Meas. **20**, 141–151 (1960)
22. Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (eds.): Web Engineering: The Discipline of Systematic Development of Web Applications. Wiley, New York (2006)
23. Koch, N., Kraus, A., Hennicker, R.: The authoring process of the UML-based web engineering approach. In: Proceedings of the 1st International Workshop on Web-Oriented Software Technology, pp. 105–119, Valencia, Spain (2001)
24. Kuhn, A., Ducasse, S., Girba, T.: Enriching reverse engineering with semantic clustering. In: Proceedings of 12th Working Conference on Reverse Engineering, 10–20, IEEE CS Press (2005)
25. Landauer, T.K., Dumais, S.T.: Solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychol. Rev. **104**(2), 211–240 (1997)
26. Levenshtein, V.L: Binary codes capable of correcting deletions, insertions, and reversals. Cybern. Control Theory **10**, 707–710 (1966)
27. Lowe, D., Kong, X.: NavOptim coding: supporting website navigation optimisation using effort minimisation. In: 2004 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 91–97, Beijing, China, IEEE CS Press (2004)
28. Maletic, J.I., Marcus, A.: Supporting program comprehension using semantic and structural information. In: Proceedings of 23rd International Conference on Software Engineering, pp. 103-112, Toronto, Ont., Canada (2001)
29. Nakov, P.: Latent semantic analysis for German literature investigation. In: Proceedings of the International Conference, 7th Fuzzy Days on Computational Intelligence, Theory and Applications, pp. 834–841, London, UK, Springer (2001)
30. Oudshoff, A.M., Bosloper, I.E., Klos, T.B., Spaanenburg, L.: Knowledge discovery in virtual community texts: clustering virtual communities. J. Intell. Fuzzy Syst. **14**(1), 13–24 (2003)

31. Pearson, J.M., Pearson, A.: An exploratory study into determining the relative importance of key criteria in web usability: a multi-criteria approach. J. Comput. Inform. Syst. (2008)
32. Ricca, F., Tonella, P.: Understanding and restructuring web sites with reweb. IEEE Multimedia **8**(2), 40–51 (2001)
33. Ricca, F., Tonella, P.: Using clustering to support the migration from static to dynamic web pages. In: Proceedings of International Workshop on Program Comprehension, pp. 207–216, Portland, Oregon, USA (2003)
34. Ricca, F., Tonella, P., Girardi, C., Pianta, E.: Improving web site understanding with keyword-based clustering. J. Softw. Maintenance Evol. Res. Pract. **20**(1), 1–29 (2008)
35. Schwabe, D., Rossi, G.: An object-oriented approach to web-based application design. Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet **4**(4), pp. 207–225 (1998)
36. Scanniello, G., Distante, D., Risi, M.: Using semantic clustering to enhance the navigation structure of web sites. In: Proceedings of the 10th International Symposium on Web Site Evolution, pp. 55–64, IEEE CS Press (2008)
37. Tonella, P., Ricca, F., Pianta, E., Girardi, C.: Restructuring multilingual web sites. In: Proceedings of the 18th International Conference on Software Maintenance (ICSM 2002), pp. 290–299, Montreal, Canada, IEEE CS Press (2002)
38. Wild, F., Stahl, C., Stermsek, G., Neumann, G., Penya, Y.: Parameters driving effectiveness of automated essay scoring with LSA. In: Proceedings of the 9th Computer Assisted Assessment Conference (CAA 2005), pp.485–494, Loughborough, UK (2005)
39. Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., Wesslen, A.: Experimentation in Software Engineering—An Introduction. Kluwer Academic Publishers Group, London (2000)
40. Tsakonas G., Papatheodorou C.: Exploring usefulness and usability in the evaluation of open access digital libraries. Int. J. Inform. Process. Manage. **44**(3), 1234–1250. Pergamon Press, Inc., New York (2008)
41. Tilley, S.: Ten years of web site evolution. In: Proceedings of the 10th IEEE International Symposium on Web Site Evolution, pp. 11–17, IEEE Press (2008)