Using Semantic Clustering to Enhance the Navigation Structure of Web Sites

Giuseppe Scanniello
Department of Mathematics and Computer Science, University of Basilicata, Italy
giuseppe.scanniello@unibas.it

Damiano Distante
Faculty of Economics, Tel.M.A. University, Italy
damiano.distante@unitelma.it

Michele Risi
Department of Mathematics and Computer Science, University of Salerno, Italy
mrisi@unisa.it

Abstract

This paper presents an automatic approach based on semantic clustering to enhance the navigation structure of Web sites. The approach extends the navigation structure of a Web site by introducing a set of links that enable the navigation from each page of the site to other pages 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 having similar or related content. The additional links are then introduced into each page of the site by using AJAX code to dynamically inject links from the page to the others within the same cluster.

A prototype of a supporting tool and the results from a case study conducted to assess the feasibility of the approach are also presented.

1. Introduction

The success of the World Wide Web derives in part from global scale and simple access to information. In a similar way, the success of a Web site, particularly informative Web sites, depends in part on its navigability, i.e., the facility for the user to navigate through its content and access its information.

To enable contents navigation, Web designers organize the contents of Web site into *nodes* (self-contained uniquely identifiable information units from/to which the user can navigate) and define *links* between them. Links between nodes are arranged to form particular access structures (e.g., guided tours, indexes, etc., a.k.a. navigation patterns) in order to support specific user information access goals (e.g., quickly access a selection of the most important or most recent contents of the site) [35].

Nodes and *links* are indeed the fundamental modeling primitives used by most known Web design methodologies to define the *navigation model* of a Web site (hypertext design), i.e., the model intended to specify how the user will be able to navigate the contents of the Web site. The list of such methodologies includes, among others, OOHDM [34], UWAT+ [16], UWE [24], and WebML [6].

When any of these design methodologies are properly used to design and develop a Web site, it is expected that its resulting navigation structure (which implements the navigation model) satisfies the user requirements in terms of contents access and navigability.

Very often, however, due to time-to-market constraints, missing skills, or poor acceptability support [19], such methodologies are not used in industrial practice. Other times, they are used only for developing the first version of the site, but then they are neglected during the following maintenance phase. Both practices may lead, at a certain time, to Web sites suffering of poor navigability: certain contents may become difficult to reach (e.g., because of too long navigation paths) or even unreachable; new published contents may lack links to existing related contents; etc. In such situations, it might be particularly useful to enhance the navigation structure of a Web site by adding dynamically built *semantic navigation maps*, i.e., maps of links connecting any given page of the site to pages showing similar or related contents.

In this paper we present an automatic approach based on semantic clustering to enhance the navigation structure of a Web site with automatically recovered semantic navigation maps. In particular, the approach uses Latent Semantic Indexing [12], a well known information retrieval technique, to compute a dissimilarity measure between the pages of the site and a Graph-Theoretic clustering algorithm [18] to group pages having similar or related content. AJAX code is dynamically injected to include a semantic navigation map in each page of the Web site. The approach has been implemented by a prototype of a supporting tool. To assess the feasibility of both the approach and the correctness of the tool prototype a case study has been conducted on a real world Web site.

The remainder of the paper is organized as follows. Section 2 briefly discusses a number of works related to ours. Section 3 presents the process to recover semantic relations among the contents of a Web site and enhance its navigation structure with our defined semantic navigation maps. Section 4 briefly describes a prototype of a tool we developed to support the process. Section 5 reports the results from a case study we conducted on a real-world Web site with the purpose of validating the feasibility of the whole approach. Finally, Section 6 concludes the paper providing some final remarks and describing future work.

2. Related Work

Recently, researchers have extensively studied the problem of defining reverse engineering techniques and tools for analyzing Web applications and support software engineers during their maintenance and evolution [1][2][3][4][5][14][17][31][32].

Ricca and Tonella [31] propose ReWeb, a tool for analyzing the structure and the evolution of static Web sites. They define a conceptual model for representing the structure of a Web site and several structural analyses relying on such model, from flow analysis to graph traversal algorithms and pattern matching. They also represent the evolution of a Web site by using colours as time indicators. This is useful to determine how the original structure of a Web site degrades during its life.

In [1], Antoniol *et al.* propose a methodology for reengineering a static Web site. The recovered design is based on the Relationship Management Data Model (RMDM) and the ER+ Model proposed within the Relationship Management Methodology (RMM) [22]. The reverse engineering phase consists of abstracting the navigational structure of the Web site and identifying the entities of the application domain and the relationship among them. The recovered ER+ diagram is restructured and the forward engineering phase is performed by following the RMM methodology.

Di Lucca *et al.* [15] propose ReUWA, a process and a supporting tool to recover user-centered conceptual models from existing Web applications. The approach refers to models defined by the UWA design methodology [38], but other Web design methods could be used instead.

Other authors have proposed approaches to redesign some or all of the aspects of a Web application. As an example, an approach to redesign Web application business processes guided by the UWAT+ methodology is proposed by Tilley *et al.* in [36], while in [28] Kong *et al.* propose NavOptim, an approach to redesign the navigation structure of a Web site.

A topic activity in the reengineering of Web applications consists of gathering the software entities that compose the system into meaningful and independent groups. Such an activity is also known as clustering. In the past a large number of approaches based on clustering algorithms have been suggested [7][10][13][32][33]. For example, different authors have used clustering algorithms to identify similar Web pages. In [32] Ricca and Tonella enhance the approach based on the Levenshtein edit distance proposed by Di Lucca *et al.* in [13] (a pairs of pages is a clone if the Levenshtein edit distance [27] between the strings encoding the page structures is zero) using a hierarchical clustering algorithm to identify clusters of duplicated or similar pages to be generalized into a dynamic page. Differently from the approach proposed in [13], the distance of cloned pages belonging to the same cluster is not zero. Similarly, in [35] the authors propose a semiautomatic approach based on an agglomerative hierarchical clustering algorithm to identify and align static HTML pages whose structure is the same and whose content is in different languages. The aligned multilingual pages are then merged into MLHTML pages. De Lucia *et al.* [11] 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 that ranges from 0%, for different pages, up to 100%, for identical pages.

Ricca *et al.* in [33] describe the results of an empirical study to group pages in Web site according to their content. Clustering is based on the similarity of the keywords within the page content. Keywords are weighted so that more specific keywords receive a higher score. Indeed, the authors used Natural Language Processing techniques to weight each keyword, according to its relevance and specificity. Similar pages are then grouped together by adopting a hierarchical clustering algorithm.

In [10] a comparison among clustering algorithms to identify similar pages at the content level is presented. Indeed, three variants of the agglomerative clustering algorithm, a divisive clustering algorithm, k-means, and a competitive clustering algorithm have been considered. This comparison revels that the selected clustering algorithms generally produce comparable results. The authors also show that adopting the competitive clustering algorithm heuristics to prune surely bad configuration can be

used. To compare pages an LSI based similarity measure is used. Differently, Rajapakse and Jarzabek in [30] analyze the results obtained by applying different cloning approaches. The study reveals that cloning equally affect small, medium or large Web applications, and the number of clones increase over the time. The authors also showed that cloning could be even worse than that of traditional applications.

LSI has been also widely employed on different application domains [7][9][25][26][29]. For example, De Lucia *et al.* [7] propose a software artifact management system that integrates an LSI engine to recovery traceability links among software artifacts. Kuhn *et al.* in [25] describe 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 different person. 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.

3. The Proposed Process

The process we propose to enhance the navigation structure of a Web site is described by the activity diagram with object flow depicted in Figure 1. In this visual notation, rounded rectangles represent process phases and subphases, while rectangles represent the produced artifacts. The overall process is composed of three main phases, namely *RecoveringSemantic-NavigationMaps*, *InjectingClientSideCode* and *DeployingNewWebSite*. Note that the firsts two phases are parallel and can be concurrently executed. In the following subsections the phases and subphases of the proposed process are detailed.

3.1 Recovering Semantic Navigation Maps

This phase consists of three subphases: *ComputingDissimilarity*, *GroupingSimilarPages* and *RemovingPages*. The ComputingDissimilarity subphase extracts contents from each of the pages of the given Web site and computes the dissimilarity between any pairs of pages using a measure based on Latent Semantic Indexing (LSI) [12]. A dissimilarity matrix is produced as output of this activity.

The GroupingSimilarPages subphase uses this matrix to identify groups of similar pages, i.e., pages with similar or related content. Pages included in single clusters, i.e., clusters containing only one page, are removed during the RemovingPages subphase and are not considered in the following iterations of the process. The GroupingSimilarPages and the RemovingPages are repeated until no single clusters remain. Further details on the three subphases of the RecoveringSemanticNavigationMaps phase are provided in the following.

3.1.1 Computing Dissimilarity

In this phase the content of the client-side HTML pages of the given Web site is extracted. This content is then used to compute the dissimilarity between pairs of pages from the semantic point of view.

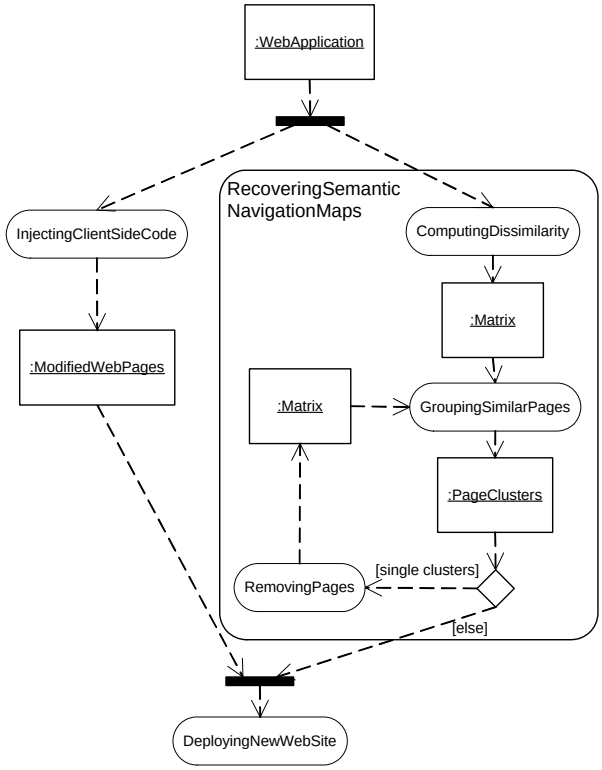

Figure 1. The proposed process.

The content of the pages has to undergo a normalization phase 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. We also use a stemming algorithm to

reduce inflected (or sometimes derived) terms to their stem. Finally, a stop word list to remove irrelevant terms is also used.

To build the concept space of the Web site we use LSI [12]. This technique has been originally developed to overcome the synonymy and polysemy problem occurring with the Vector Space Model (VSM) [21]. In fact, LSI explicitly considers dependencies between terms and between documents (corresponding to Web pages in our case), in addition to the associations between terms and documents. This technique assumes that there is a latent structure in word usage that is partially obscured by variability in word choice.

To apply LSI, a term-by-content matrix *A* has to be built. This matrix is *m* x *n*, where *m* is the number of terms within the content of the pages 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 *i-th* term in the *j-th* page. To derive the content latent structure of a Web site we apply on this matrix a Singular Value Decomposition (SVD) [12]. 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 a *r* x *r* diagonal matrix of singular values and *T* and *D* have orthogonal columns. SVD also allows 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. [39]. In our case, we calculate the number of singular values according to the Guttman-Kaiser criterion [20][23]. This criterion considers the diagonal matrix S of the singular values, which are a kind of eigenvalue, of *A* in descending order. The number of singular values in S more than 1 is selected as the value of *k*.

Each term and page of the considered Web site could be represented by a vector in the *k* space of the underlying concepts. In our approach we consider the pages, where the rows of the reduced matrices of singular vectors are taken as coordinates of points representing the pages in a *k* dimensional space. The concept space is then used to build a dissimilarity matrix. To build this matrix, pairs of pages are compared considering the cosine between the vectors of the pages in the content space. The similarity between a pair of pages ranges from -1 (when they have a different latent structure) to 1 (when the latent structure is the same). To compute page dissimilarities we normalize the cosine similarity measure from 0 (when the latent structure is the same) to 1 (when they have a different latent structure). Hence, given two pages p_1 and p_2, the dissimilarity between these two pages is defined as:

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

where Vp_1 and Vp_2 are the vectors corresponding to the pages p_1 and p_2 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.

3.1.2 Grouping Similar Pages

This phase uses a Graph-Theoretic clustering algorithm [18] to group pages that are similar at the content level. 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. The nodes of each tree of the obtained forest are included in a cluster.

In our approach the Graph-Theoretic clustering algorithm is used on the strongly connected graph corresponding to the dissimilarity matrix computed in the phase ComputingDissimilarity. Nodes are pages and edge weights are the dissimilarity measures between the pairs of pages. To get the clusters of similar pages we use as pruning threshold the arithmetic mean of the edge weights of the built MST.

In case the clustering algorithm identifies single clusters, the phase RemovingPages is performed in order to remove from the dataset the pages included in the single clusters. The new dataset is then provided as input to the Graph-Theoretic clustering algorithm. The phases GroupingSimilarPages and RemovingPages are iterated until no single clusters are identified. To improve the output quality of this phase the software engineer might manually modify the set of automatically identified clusters. Also, the single clusters identified in the early iterations can be involved in this manually performed activity.

Figure 2 shows an example of the clustering results obtained by using the Graph-Theoretic clustering algorithm on 23 pages of the Web site used in the case study (see Section 5). In particular, this figure shows the built MST and the clusters identified by using 0.33 as threshold. The clustering algorithm identified 9 different groups of pages with similar or related content. Among the identified clusters three were

single clusters. Note that neither the names nor the title of the pages are reported for readability reasons. The weights of the edges less than the threshold are not shown as well.

3.1.3 Removing Pages

The phase RemovingPages is executed in case the Graph-Theoretic clustering algorithm identifies one or more single clusters. This phase is in charge of removing the rows and the columns corresponding to the pages that have been placed within single clusters from the dissimilarity matrix. The obtained matrix is then provided as input to GroupingSimilarPages. RemovingPages is executed until no single clusters are identified by the clustering algorithm. We decide to remove the pages of single clusters from the set of pages since they have a low level of semantic similarity with the remaining pages.

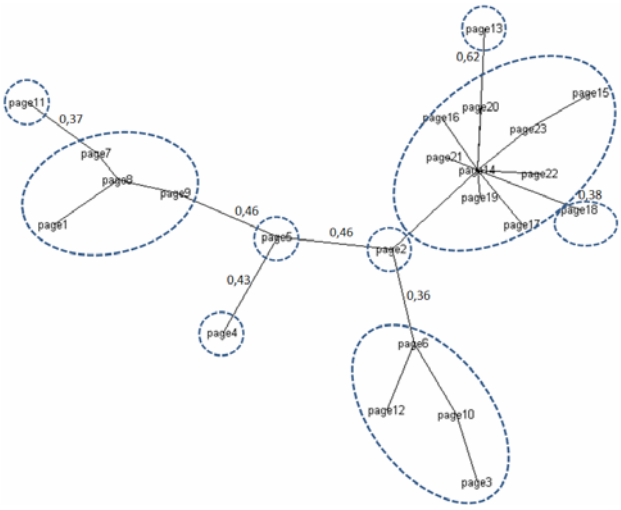

Figure 2. Clusters of pages.

3.2 Injecting AJAX Code into Client Pages

This phase aims at enhancing the navigation structure of the Web site by introducing semantic navigation maps. Each page of the site is extended to dynamically include a set of links towards the pages that the grouping process has placed in the same cluster.

We use AJAX and DOM technologies to respectively query the server that maintains the data on the clusters of similar pages and to display the semantic navigation maps. For the first objective we use Javascript function to retrieve the list of pages within the same cluster of a given page. For the second purpose we use a different Javascript function to interpret the server answer and to properly display the map of links in the considered Web page by injecting HTML code into the DOM of the page. Indeed, we use the DIV tag and injected the HTML code at the end of each page. At run-time the Javascript code is used to identify the area where the semantic navigation map has to be visualized.

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, thus reusing them en each page of the evolved Web site.

It is worth noting that the HTML pages of a given Web site are modified once for all. This is possible because the clusters are independently detected (see Figure 1) and are dynamically obtained querying the server. Let us also note that 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 consistence reasons, the identification of similar pages should also be performed when pages are removed.

3.3 Deploying the Enhanced Web Site

In the *DeployingNewWebSite* phase the pages enhanced with the semantic navigation maps and the data on the recovered clusters are deployed on the server. The deployed pages require a server component to retrieve all pages with similar or related content. To enable the communication between each enhanced page of the Web site and the server where the corresponding cluster is stored we have developed a servlet. The software engineer has to deploy this servlet only once. Finally, when the enhanced pages and the servlet have been deployed, the evolved Web site is enabled to be accessed by the users.

In this phase, the software engineer should also perform testing to find possible faults. Currently, our tool prototype does not provide any specific support to test the new Web site.

4 Tool Prototype

To support all the phases of the proposed process, we have implemented a prototype of a supporting tool in Java. Figure 3 shows the layered architecture of the tool using a UML Package Diagram. *Data* contains the pages of the Web site and all the intermediate representations produced by the modules of the tool prototype. The *Computing Dissimilarity* component

uses the *HTML Parser* component to extract the content of the HTML client-side pages of the application. HTML Parser integrates an open source HTML parser written in Java (HTMLParser ver. 1.6), available under the GPL license at sourceforge.net/projects/htmlparser.

The extracted content is stored in the local file system of the tool prototype to avoid extracting the same content 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[1] implementation of the LSI information retrieval technique. This implementation is available under GPL license from cran.r-project.org.

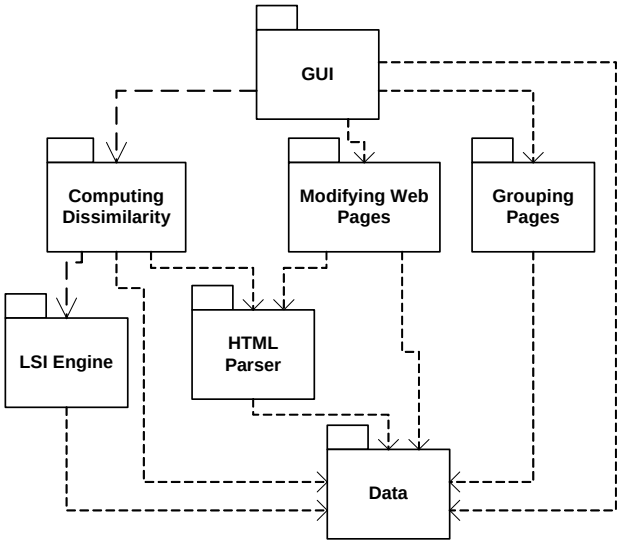

Figure 3. The architecture of the tool prototype.

The dissimilarity matrix produced by Computing Dissimilarity is stored in the file system of the tool prototype. This is due to the fact that the computation of this matrix is very expensive. For example, on the pages used as case study (see Section 5) the computation of the dissimilarity matrix takes about three hours using a laptop equipped by a 1.5 GHz Intel Centrino with 1,5 GB of RAM, a 60GB 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) is integrated. Grouping Pages also removes from the dissimilarity matrix rows and columns corresponding to the pages within single clusters.

Modifying Web Pages injects the Javascript code into the HTML Web pages of the given Web application to enable the communication with the server and to display the semantic navigation maps (see Section 3.2 for details).

The point where the Javascript code has to be injected is identified using the HTML Parser component. The *GUI* component implemented in Java enables the software engineer to select the pages of the given Web application and to display the groups of similar pages detected by the tool. Pages can be added or removed, thus improving the overall quality of the identified groups of similar pages. It is worth noting that the R implementations of the LSI engine and the clustering algorithm are all integrated using Rserve (from cran.r-project.org). This is 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 in order 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 site could be considerable.

5. Case Study

The approach and the tool prototype have been assessed on the official Web site of the National Gallery of London (available at www.nationalgallery.org.uk). In the following subsections we present and discuss the results obtained from this case study.

5.1 Context and Results

To assess the approach and the prototype we have dumped the pages of the considered Web site on June 9th, 2008. To this end, we used a freeware dumper (HTTrack Website Copier) available at www.httrack.com. This tool 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.

[1] 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.

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

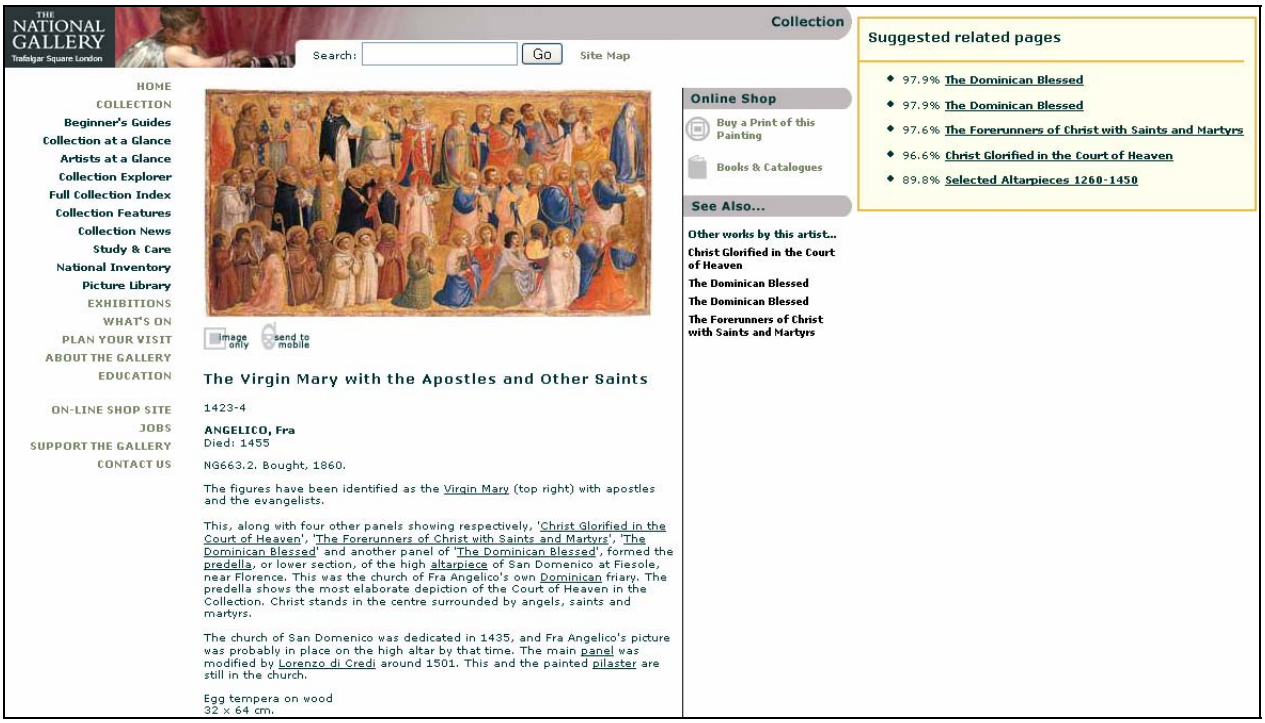

Figure 4. A page of the Web site enhanced with the Semantic Navigation Map.

The mirror of the National Gallery Web site has been successively analyzed to prune 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). The total number of HTML pages we have selected and considered in the presented case study is 2017.

Table 1. Descriptive statistics for the case study.

Number of analyzed pages	2017
Number of identified clusters	243
Number of iterations	5
Number of pages within single clusters	1387
Number of pages within clusters containing at least two pages	630
Number of pages within the largest cluster	32
Mean number of pages within the clusters	2.6

The selected pages were then provided as input to the tool prototype, which grouped them in 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. Some descriptive statistics on the case study are summarized in Table 1.

The tool prototype has been also used to inject the AJAX code required to dynamically add to each of the analyzed pages the set of links connecting it with the other pages in the same cluster. An example of the resulting pages is shown in Figure 4. It is worth noting that this page differs from its original version for the presence of the semantic navigation map that is shown on the right hand side. The semantic navigation map presents a set of links towards pages that have been found similar in content to the considered one. Each link shows: (*i*) the title of the target page; (*ii*) the percentage of similarity with the current page obtained using LSI (i.e., the cosine between the vectors of the pages in the content space); (*iii*) a description of the target page obtained from data in its description meta tag, if available.

5.2 Discussion

By analyzing the pages of the Collection section of the 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 Figure 4). In particular, given an artist and his/her work of art the menu enables users to directly access the pages presenting the other works of art by the same artist exhibited at the National Gallery of London. Additional links are also proposed to navigate and to access related pages.

In our case study, we observed that most of the links presented by this menu were also proposed by analogous links in the semantic navigation map automatically identified by the tool. As an example, if we compare the navigation menu and the semantic navigation map for the page shown in Figure 4 we can notice that are basically the same. The only difference we can note is the link *Selected Altarpieces 1260-1450* that is present in the semantic navigation map. This link enables the user to access a page presenting works of art from different artists on the same theme (i.e., altarpieces). In some other cases, we also noted that some pages of the Collection section presented a navigation menu with a number of links larger than the ones proposed by the semantic navigation maps. However, the semantic navigation maps of the majority of these pages present the same links as the corresponding navigational menu. This indicates that the links of the identified semantic navigation maps are generally correct.

Overall, the results obtained from the conducted case study suggest two considerations. The first concerns correctness: links included in the recovered navigation maps actually propose pages with content similar or related to the one showed by the considered page, and thus of possible interest for the user. The second consideration concerns completeness: the list of proposed links includes most of the pages with actually related content.

The careful reader may object that in the case in point the recovered semantic navigation maps may result in duplicate navigation structures (the navigation menu). This is true; indeed the purpose of the conducted case study was to assess the correctness and the degree of completeness of the produced navigation maps, a goal we can state the case study reached. We can thus expect that our approach will provide correct and complete semantic navigation maps also for those Web sites where such semantic navigation structures are not available.

6. Conclusion

In the following we provide some final remarks on the work presented in this paper and describe some possible future extensions.

6.1 Final Remarks

Our research is meant to automatically recover semantic relations between the contents 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 (LSI) [12], a well known information retrieval technique, and then groups the pages using a Graph-Theoretic clustering algorithm [18]. Finally, the navigation structure of the Web site is enhanced by adding hyperlinks among pages within the same cluster using AJAX code [7].

When we start this work, we consider the client-side HTML pages of the site as the elementary granules of information (nodes) to analyze, index and link, and assume that the content (text) included in a page is uniquely identified by the page URL. As such, the recovery process and the supporting tool we have developed are applicable to Web sites in which the content associated to a page does not depend on the user logged on nor other context variables, thought it may change during time[3]. Our approach is instead applicable no matter of the technologies used server-side to produce the front-end of the application, provided that this is represented by HTML pages.

It is worth noting that though semantic navigation maps may be useful when the navigation structure of the site was not properly (or actually) designed or when it gets out-of-date compared with the Web site contents, also properly designed Web sites may benefit from introducing them. Indeed, semantic navigation maps represent an additional navigation structure, complementary to those obtained from implementing the navigation model designed with one of the above cited Web design methodologies. While the latter are usually designed to satisfy a specific navigation requirement (e.g., provide access to subsets of contents grouped by category or having some characteristic of interest for the user, etc.), the former are intended to make explicit the latent semantic relations between the contents of the site and keep this relation up-to-dates when the content of the site evolves.

Similarly to Web site search engines such as Google Site Search[4] or FreeFind[5], semantic navigation maps represent a navigation structure built by analyzing and indexing the contents of the Web site. Differently from them, the index provided by a semantic navigation map are not the result of a search executed by the user at run-time, but a navigation

[4] www.google.com/coop/cse

[5] www.freefind.com

structure provided by default by the site to its user. More importantly, the index is obtained considering as "search input parameters" not a particular keyword or set of keywords, but the full text of the page the user is visiting.

To support the software engineer the approach has been implemented in a tool prototype. Both the approach and prototype have been also assessed on a real-world Web site, namely the official site of the National Gallery of London.

6.2 Future Extensions

In the future, we plan to apply the approach on other Web sites of different size and from different application domains. We also plan to conduct controlled experiments to investigate the effectiveness of Web sites, whose navigation structure is enhanced using our approach. These experiments will aim at assessing whether the enhanced version of the sites better satisfy the users' expectations in terms of contents navigation and information access. A further experimentation should be also conducted to evaluate the accuracy of the clusters of pages identified as similar at the semantic level by the tool prototype. With the same intent of meeting the user's expectation and requirements, we also mean to investigate the feasibility of adapting the presented results (proposed navigation links) based on the user profile and its expressed interest on the application content.

Future work will also be devoted to investigate the effect of adopting and combining different measures, means and thresholds to group pages with similar or related content. It will be also worth extending both the approach and the tool to make them suitable for dynamic Web sites, i.e., Web sites for which the content showed into pages and the accessible pages varies depending on some context variable (the user profile, location, etc.). The approach and the tool will be also extended to analyze PDF and Word files. Finally, we plan to develop software components to be integrated in different and widely employed Web applications, e.g., content management systems, e-learning platforms and e-commerce application frameworks.

References

[1] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia. "Web Site Reengineering using RMM". *Proc. of the 2nd International Workshop on Web Site Evolution*, Zurich, Switzerland, 2000, pp. 9-16.

[2] C. Boldyreff and P. Tonella. "Web Site Evolution". Special Issue of the *Journal of Software Maintenance*, Vol. 16, no 1-2, 2004, pp. 1-4.

[3] C. Boldyreff, M. Munro, and P. Warren. "The evolution of Websites". *Proc. of the International Workshop on Program Comprehension*, Pittsburgh, Pennsylvania, IEEE CS Press, 1999, pp. 178-185.

[4] C. Boldyreff and R. Kewish. "Reverse Engineering to Achieve Maintainable WWW Sites". *Proc. of the 8th IEEE Working Conference on Reverse Engineering*, Suttgart, Germany, IEEE CS Press, 2001, pp. 249-257.

[5] F. Calefato, F. Lanubile, and T. Mallardo. "Function Clone Detection in Web Applications: A Semiautomated Approach". *Journal of Web Engineering*, Vol.3, No.1, Rinton Press, 2004, pp. 3-21.

[6] S. Ceri, P. Fraternali, and A. Bongio. "Web Modeling Language (WebML): a Modeling Language for Designing Web Sites". *Computer Networks* 33(1-6): 137-157, 2000.

[7] D. Cran, E. Pascarello and J. Darren. "Ajax in Action" Manning Publications Co. October, 2005. ISBN: 1932394613

[8] A. De Lucia, G. Scanniello, and G. Tortora "Identifying Similar Pages in Web Applications using a Competitive Clustering Algorithm". In *Journal on Software Maintenance and Evolution*, Vol. 19, No. 5, September-October 2007, Wiley, pp.: 281-296.

[9] A. De Lucia, F. Fasano R. Oliveto, and G. Tortora. "Recovering traceability links in software artifact management systems using information retrieval methods". Transaction on Software Engineering and Methodology, vol. 16, no. 13, article no. 13, 2007.

[10] A. De Lucia, M. Risi, G. Scanniello, and G. Tortora "Clustering Algorithms and Latent Semantic Indexing to Identify Similar Pages in Web Applications", *Proc. of the 9th IEEE International Symposium on Web Site Evolution*, Paris, France, October 5-6, 2007, IEEE CS Press, pp. 65-72.

[11] A. De Lucia, R. Francese, G. Scanniello, and G. Tortora. "Identifying Cloned Navigational Patterns in Web Applications". In *Journal of Web Engineering* vol. 5, no.2, Rinton Press, 2006, pp. 150-174.

[12] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. "Indexing by Latent Semantic Analysis", *Journal of the American Society for Information Science*, No. 41, 1990, pp. 391-407.

[13] G. A. Di Lucca, M. Di Penta, and A. R. Fasolino. "An Approach to Identify Duplicated Web Pages". *Proc. of the 26th Annual International Computer Software and Application Conference*, Oxford, UK, IEEE CS Press, 2002, pp. 481-486.

[14] G. A. Di Lucca, M. Di Penta, G. Antoniol, and G. Casazza. "An Approach for Reverse Engineering of Web-based applications". *Proc. of the 8th IEEE Working Conference on Reverse Engineering*, Suttgart, Germany, IEEE CS Press, 2001, pp. 231-240.

[15] G. A. Di Lucca, D. Distante, M. L. Bernardi. "Recovering Conceptual Models from Web Applications". *Proc. of the 24th International Conference on Design of Communication*, (SIGDOC 2006, Myrtle Beach, USA). ACM Press, 2006.

[16] D. Distante, G. Rossi, G. Canfora, and S. Tilley. "A Comprehensive Design Model for Integrating Business Processes in Web Applications". *International Journal of Web Engineering and Technology*, Vol. 2, No. 1, 2007, pp 43-72. Inderscience Publishers, 2007.

[17] D. Eichmann "Evolving an Engineered Web". Proc. International Workshop Web Site Evolution, Atlanta, GA, 1999, pp. 12-16.

[18] P.J. Flynn, A. K. Jain, and M. N. Murty "Data Clustering: A Review". In *ACM Computing Surveys*, vol. 31, no. 3, 1999, pp. 264-323.

[19] F. Garzotto and V. Perrone. "On the Acceptability of Conceptual Design Models for Web Applications". In *Proc. of Conceptual Modeling for Novel Application Domains* – ER'03 Workshops. (Chicago, US, Oct.03), LNCS – 2814/ 2003, p. 92-104.

[20] L. Guttman. "Some necessary conditions for common factor analysis". *Psychometrika*, Vol. 19, 1954, pp. 149-61.

[21] D. Harman. "Ranking Algorithms", In *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall, Englewood Cliffs, NJ, 1992, pp. 363–392.

[22] T. Isakowitz, E. A. Stohr, and P. Balasubramanian, "RMM: a Methodology for Structured Hypermedia Design", *Communications of the ACM*, vol. 38, no. 8, 1995, pp. 34–44.

[23] H. F. Kaiser. "The application of electronic computers to factor analysis". *Educational and Psychological Measurement*, vol. 20, 1960, pp. 141-51.

[24] N. Koch, A. Kraus, and R. Hennicker. "The Authoring Process of the UML-based Web Engineering Approach" *Proc. of the 1st International Workshop on Web-Oriented Software Technology*, Valencia, Spain (2001), pp. 105-119. 2001.

[25] A. Kuhn, S. Ducasse, and T. Girba. "Enriching reverse engineering with semantic clustering", *Proc. of 12th Working Conference on Reverse Engineering*, IEEE CS Press, 2005, pp. 10-20.

[26] T. K. Landauer and S. T. Dumais. "Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge" *Psychological Review*, 1997, vol. 104, no. 2, pp. 211-240.

[27] V. L. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals". *Cybernetics and Control Theory*, vol. 10, 1966, pp. 707-710.

[28] D. Lowe and X. Kong, "NavOptim Coding: Supporting Website Navigation Optimisation using Effort Minimisation". In *2004 IEEE/WIC/ACM International Conference on Web Intelligence*, Beijing, China, 2004, IEEE Computer Society, 91-97.

[29] P. Nakov. "Latent semantic analysis for german literature investigation" *Proc. of the International Conference*, 7th Fuzzy Days on Computational Intelligence, Theory and Applications, London, UK, 2001, Springer-Verlag, pp. 834–841.

[30] C. Rajapakse and S. Jarzabek, "An Investigation of Cloning in Web Applications". *Proc. of 5th International Conference on Web Engineering*, Sydney, Australia, 2005, pp. 252-262.

[31] F. Ricca and P. Tonella, "Understanding and Restructuring Web Sites with ReWeb", *IEEE Multimedia*, vol. 8, no. 2, 2001, pp. 40-51.

[32] F. Ricca and P. Tonella, "Using Clustering to Support the Migration from Static to Dynamic Web Pages". *Proc. of International Workshop on Program Comprehension*, Portland, Oregon, USA, 2003, pp. 207-216.

[33] F. Ricca, P. Tonella, C. Girardi, and E. Pianta, "Improving Web site understanding with keyword-based clustering" *In Journal of Software Maintenance and Evolution: Research and Practice*, vol. 20, n. 1, 2008, pp. 1-29.

[34] D. Schwabe and G. Rossi, "An object-oriented approach to Web-based application design". *Theory and Practice of Object Systems* (TAPOS), Special Issue on the Internet, Vol. 4, Issue 4, October, 1998, pp. 207-225.

[35] W. Schwinger and N. Koch, "Modeling Web Applications", In "Web Engineering", Chapter 3. G. Kappel, B. Pröll, S. Reich, W. Retschitzegger (Eds.) Wiley, 2006.

[36] S. Tilley, D. Distante, and S. Huang. "Web Site Evolution via Transaction Reengineering." In Proc. of the 6th IEEE International Workshop on Web Site Evolution (WSE 2004: Sept. 11, 2004; Chicago, IL). Los Alamitos, CA: IEEE CS Press, 2004.

[37] P. Tonella, F. Ricca, E. Pianta, and C. Girardi, "Restructuring Multilingual Web Sites". *Proc. of the 18th International Conference on Software Maintenance* (ICSM 2002), Montreal, Canada, IEEE CS Press, 2002, pp. 290-299.

[38] UWA Consortium, "Ubiquitous Web Applications". In Proc. of the *eBusiness and eWork Conference 2002*, (e2002: October 2002; Prague, Czech Republic).

[39] F. Wild, C. Stahl, G. Stermsek, G. Neumann, and Y. Penya. "Parameters Driving Effectiveness of Automated Essay Scoring with LSA". *Proc. of the 9th Computer Assisted Assessment Conference* (CAA 2005), Loughborough, UK, pp.485-494.