ENHANCING NAVIGABILITY IN WEBSITES BUILT USING WEB CONTENT MANAGEMENT SYSTEMS

DAMIANO DISTANTE
Faculty of Economics, Unitelma Sapienza University
Italy
damiano.distante@unitelma.it

MICHELE RISI
Dept. of Maths and Computer Science, University of Salerno
Italy
mrisi@unisa.it

GIUSEPPE SCANNIELLO
Dept. of Maths and Computer Science, University of Basilicata
Italy
giuseppe.scanniello@unibas.it

Websites built using Web Content Management Systems (WCMSs) usually provide their users with three alternative access structures to surf their contents: indexes of categories, breadcrumb trails, and sitemaps. In addition, to find contents of his/her interest, a user can perform more or less advanced full-text searches. In this paper we propose an automatic approach to extend the navigation structure of websites developed using WCMSs with Semantic Navigation Maps (SNMs), a complementary navigation structure that enables linking and navigating contents based on their lexical similarity. The approach uses an information retrieval technique (namely, Latent Semantic Indexing) to identify lexical similarities between textual contents, and a fuzzy clustering algorithm to form groups of similar web pages. For each page of the website, a set of navigation links towards pages showing similar content and a measure of such similarity is provided. The paper presents the approach to generate SNMs, an implementation for the Joomla! open source WCMS, and the results of an empirical evaluation involving two real world websites built using this WCSMs.

Keywords: Website navigation structure, website navigability, web systems evolution, web content management systems (WCMSs), Semantic Navigation Maps (SNMs), Latent Semantic Indexing (LSI), clustering techniques.

1. Introduction

A Web Content Management System (WCMS) is a software system that enables its users to easily and quickly create, edit, organize, and publish contents in a website, with little training and without the need for computer programming knowledge. Thanks to the significant benefits carried by the adoption of these kinds of systems and to the availability of robust and reliable open source options for them, the number of websites built and managed using WCMSs is continuously growing.

---

1 A rich list of commercial and open source WCMSs can be found in Wikipedia. A review for many of them can
2 Such “contents” are usually referred as “articles” or “pages” in a WCMS jargon, and in this paper we will use these three terms interchangeably.
The list of functionalities usually offered by a WCMS includes:

- Automatic generation of website navigation and content organization.
- Rich text content editing through WYSIWYG authoring tools.
- Templating, i.e., easily switching between different prebuilt or customized presentation themes, thanks to separation of contents from the code that makes them visible.
- Security management, which includes session management and role-based and permission-based access control to contents and operations.
- Modularity, which enables a WCMS to be extended with additional modules and plug-ins.

More advanced features a WCMS may offer are: version control and archiving; granular editing privileges; search engine optimization (SEO); global and site-specific content sharing; automated content approval workflow; multilingual website management; integrated file manager for non-content assets; page caching technology; user profiling and customization.

Focusing on navigation features, websites relying on WCMSs usually offer their users two possibilities to find contents of their interest: (i) surf the website using its navigation structures; (ii) perform full-text searches by means of a search engine provided by the system or integrated in the site with an external service such as Google Site Search.

Navigation structures automatically built and dynamically managed by WCMSs are usually instances of the following types:

- Hierarchical navigation indexes enabling contents navigation based on their classification into categories and sub-categories. Tag-based navigation can be considered a variant of this type of navigation.
- Breadcrumb trails showing the path to the current page all the way from the home page.
- Sitemaps showing the list of pages forming the website in a hierarchical organization.

Additional navigation structures to support specific information access goals have to be purposely created by the user, manually or with the help of some module extending the WCMS. Examples of such navigation structures are represented by “guided tours” enabling the ordered navigation between the members of a collection of contents sharing some property, and links connecting similar contents.

The research presented in this paper is aimed at extending the navigation structure usually provided by WCMSs and synthesized above with Semantic Navigation Maps (SNMs) [37]. It is a complimentary navigation structure that enables navigating contents based on their similarity and correlation. In this paper, we describe the proposed approach, the underlying techniques, and an implementation for the Joomla! WCMS. We also report on the results of an empirical evaluation conducted with experts on a real world website developed with this system.

The work presented in this paper is built on that presented in [16] and with respect to it proposes an enhanced version of the approach to recover SNMs and the application of

\[ \text{http://www.google.com/sitesearch.} \]
SNMs to the realm of websites built via WCSMs. The main new contributions can be summarized as follows:

- The approach has been described better and more details on how it identifies and builds SNMs are given.
- An implementation of the overall approach to propose links to pages with similar content (i.e., SNMs) as a module for the Joomla! WCMS which enables introducing SNMs into websites developed with this system and, in general, to any website based on a WCMS.
- A validation of the approach and the developed technology on two real world websites developed with Joomla! is presented. This validation is conducted with experts and is based on the correctness of the SNMs the approach and the Joomla! module recovered from the original websites.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we highlight our approach to recover SNMs and its underlying techniques. In Section 4, we propose an implementation of the approach for Joomla!, while we report on an empirical evaluation conducted on two real world websites in Section 5. Final remarks and future work conclude the paper and are provided in Section 6.

2. Related work

Our approach does not employ ontologies, nor it applies technologies specific for the Semantic Web (e.g., OWL [29], RDF [31], or SPARQL [42]) to explicitly annotate contents in order to index and search them. Instead, we use Latent Semantic Indexing (LSI) [12][19] to discover the latent semantics of contents and a clustering algorithm to group contents that are found to be similar. The data on the recovered clusters are then used to build a complementary navigation structure (i.e., SNMs), which enables navigating between the contents of the site based on their lexical similarity. As such, our work relates to the area of Semantic Web for its intent, but less for the adopted techniques and technologies.

LSI has also been widely applied in data mining research. A method based on LSI to compare German literature texts is proposed by Nakov [27]. The study evaluates whether texts by the same author are alike and can be distinguished from the ones by a 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. A related research by Liu et al. [24] propose a local LSI method called “local relevancy weighted LSI” to improve text classification by performing a separate single value decomposition (SVD) on the transformed local region of each class. Simko and Bielikova [40] propose a method for automated metadata generation for the educational knowledge discovery problem employing, among others, LSI as a technique for data mining.

IR techniques have been also widely used in different areas of software engineering (e.g., [14][20][30][36]). In this section, we first focus on related work concerning approaches based on IR and/or clustering techniques applied to web applications/systems and then we discuss approaches that apply these techniques to software engineering.
2.1. Web Applications and Systems

Clustering is the technique of gathering the software entities that compose a system into meaningful and independent groups. In the past a large number of approaches based on clustering algorithms have been suggested also in the field of web applications. For example, different authors have used clustering algorithms to identify similar web pages. Ricca and Tonella [35] enhance the approach based on the Levenshtein edit distance proposed by Di Lucca et al. [15] (a pair of pages is a clone if the Levenshtein edit distance 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. The same authors describe [34] 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 and relevant keywords receive a higher score. While we also adopt a clustering algorithm to group articles with similar/related content, we compute the similarity measure using LSI instead of Natural Language Processing (NLP) and keywords identification and weighting. The same authors present a semi-automatic 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 [41]. The aligned multilingual pages are then merged into MLHTML pages.

De Lucia et al. [9] propose a semi-automatic 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. 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 successively proposed by De Lucia et al. [10].

De Lucia et al. also present the results of an empirical investigation on different clustering algorithms [11]. The main goal is to compare well-known clustering algorithms in the identification of similar pages at the content level. Page similarity is computed using a measure based on LSI. 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.

Scanniello et al. [37] propose a reengineering approach to enhance legacy web application with SNMs. This approach recovers SNMs analyzing the content of the legacy application and then adds these SNMs to web pages. A remarkable difference with the information retrieval approach presented here is that the clustering algorithm is not fuzzy (i.e., it is hard clustering algorithm). Then, each page is only in a single SNM, so reducing the navigation capability of the reengineered web application. In fact, it is quite common in a real web system that a page contain heterogeneous contents that are semantically related to different pages, so needing that it has to appear in more SNMs. In some sense, the approach presented here fills in a gap in that approach.

There are several differences between our proposal and those we have discussed above. First of all, we defined an approach based on LSI and fuzzy clustering. This
approach has been implemented as a module for the Joomla! WCMS. The module can be easily integrated into many other open source and commercial WCMSs. In this paper, our proposal has been also empirically validated with experts to assess the quality of the complimentary navigation structures automatically obtained on two real web applications based on Joomla!. Among the very rich collection of open-source and commercial extensions available for this WCMS\(^4\), there can be found a number, belonging to the category “Structure and Navigation”, which have an intent similar to ours. Some of them relate articles based on the similarity of their titles, by considering terms present in the titles as keywords [33][39]. Others relate articles based on the value assumed by a set of keyword tags, specified by the user [32][21] or automatically calculated [1] when publishing new articles. Finally, others support the user in manually creating links between articles, thus providing her with full control on these links, but at a full charge management effort. To the best of our knowledge, none of these extensions enable automatically relating articles based on the similarity of their full textual content, and none of them adopts LSI and clustering techniques as it happens for ours.

### 2.2. IR and Software Engineering

IR techniques have been deeply exploited for a number of objectives of the research in the area of software engineering. As an example, Antoniol et al. [2] suggest the use of the IR technique VSM (Vector Space Model) to the recovery of traceability links between documentation and source code. More recently, De Lucia et al. [10] propose a software artifact management system that integrates an LSI engine to recovery traceability links among high and low level software artifacts.

Scanniello and Marcus [38] propose a static approach for concept location in source code based on VSM and the BorderFlow clustering algorithm. This algorithm is used to cluster the software prior to concept location, using structural information. The work of Revelle et al. [13] introduces a hybrid (i.e., static + dynamic) concept location technique, which uses textual search of an execution trace, which is pruned using web mining analysis algorithms. A more complete discussion on the use of IR techniques to concept location is available in the survey by Dit et al. [14].

Kuhn et al. [23] 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. The authors consider different levels of abstraction to understand the source code (i.e., methods and classes). More recently, Corazza et al. [6] propose a clustering based approach that uses lexical information extracted from four zones in Java classes, which are weighed using a probabilistic model and indexed by VSM. The Expectation-Maximization (EM) algorithm is applied. Classes are then grouped using a customization of the k-medoids clustering algorithm. Successively, the same authors empirically analyzed the effect of differently considering the lexical information coming from six zones in java source code [7]. These two papers represent a first attempt in the software engineering to consider the lexical information in source code in a non flat way. The partition of the content of a web site in more zones could represent a possible future

---

\(^4\) A rich and up-to-date list of extensions for Joomla! can be found at http://extensions.joomla.org/
direction for the work we present in this paper. In the context of software clustering and architecture recovery, Risi et al. [36] propose an approach for evolving software systems built on LSI, to get similarities among software entities (e.g., programs or classes), and the k-means clustering algorithm, to form groups of software entities that implement similar functionality. The authors also compare the results achieved by applying the general definition of LSI and its variation based on the fold-in and fold-out mechanisms on 90 versions of 5 Java/C++ systems. The results indicated that the use of the fold-in and fold-out mechanisms do not produce worse results provided that there are not huge variations in the text of two subsequent versions of a given system. This result encouraged us to use the fold-in and fold-out mechanisms also in the problem at hand: improving the navigability of websites developed with WCMSs.

IR techniques have been also used to define and measure cohesion [26] and coupling [30] in object oriented software. Both the measures are based on the analysis of the unstructured information embedded in the source code, such as comments and identifiers. Marcus et al. [26] compare the new measure with an extensive set of existing metrics and use them to construct models that predict software faults. Differently, conceptual coupling of classes is proposed by Poshyvanyk et al. [30]. This measure indicated the degree to which identifiers and comments from different classes are related to each other. This measure, capture new dimensions of coupling not captured by other metrics, has been empirically investigated in the context of change impact analysis.

In this section, we do not present all the research work concerning the application of IR and/or clustering to software engineering tasks. This would be out of the scope of the paper. Our goal here is to show that these techniques are successfully used in several and heterogeneous context of software engineering, so justifying their adoption in the problem we are dealing with in this paper.

3. The proposed approach to recover Semantic Navigation Maps

3.1. Overall description of the approach

As discussed in Section 1, navigation natively supported by most WCMSs is usually limited to category and tag based access to contents (in the context of these systems often referred as “articles”, ultimately corresponding to webpages), sitemaps, and breadcrumbs trails. Additional navigation structures to support specific information access goals have to be manually created and kept up to date by the user of the WCMS. This is also due to the lack of support of most of these systems for any specific web design methodology (Casteleyn et al. [5] show representative list of such methodologies), which would enable designing the navigation structure of the website, prior to generating it. In particular, navigation between related contents belonging to different categories has to be supported with links explicitly defined and kept up to date.

Our proposal to improve navigation in websites built with WCMSs is based on the automatic discovery of the relations that exist, at content level, between the articles published in the site, and on incrementing the navigation structure provided by WCMSs with SNMs. A SNM is a set of links that connect a given article to others which are found
to have a correlation to it. We have defined an approach to recover SNMs that has proven validity in real world websites [37]. In this work we propose a variant of this approach, which adopts a more effective clustering techniques, and an implementation that enables integrating the approach it the context of WCMSs and in Joomla! in particular.

The approach uses the LSI technique [12][19] to compute a similarity measure between the articles of the website, based on their textual content. Then we propose the use of a fuzzy clustering algorithm [21] to identify clusters (groups) of articles with a similar content. Links connecting a given article to others within the same cluster are then proposed to the user visiting the site as additional navigation paths. Each of the links is also accompanied by a measure of the similarity between the current article and the pointed one.

The rationale for using a fuzzy clustering algorithm to group articles relies on the fact that an article could be associated to several concepts of the domain of the considered website, and for this reason it should belong to more clusters. The use of a graph-theoretic algorithm, as we did in [37], causes each article to belong exactly to a cluster, and thus it is associated with only one concept of the domain of the website.

In case new articles are added to the WCMS, the process incrementally performs the indexing, thus significantly reducing the time needed by this task. However, this has the drawback that the link correctness of the SNMs may be affected and some links to be missed. In fact, new terms of an inserted or a modified article may not contribute to the existing latent semantic space and then links among the new content and existing ones may be missed. In case the number of missed links becomes significant, the new latent semantic space can be re-computed on all the contents. To this aim, the new latent semantic space may be computed every week/month to update the complementary navigation structure of the WCMS according to the newly added contents.

3.2. The recovery process and the underlying techniques

The diagram in Figure 1 depicts the overall process (an UML activity diagrams with object flow). In the diagram, rounded rectangles represent process phases whilst rectangles represent the intermediate artifacts produced at the end of each phase.

The process is composed of two subsequent phases: Computing Similarity and Grouping Articles. The first phase is in charge of computing dissimilarity between pairs of articles present in the content repository. Successively, the second phase is performed to identify a suitable grouping of these articles. In the following we detail each of these phases and the underlying techniques.
3.2.1. Computing similarity

This phase is detailed by the UML activity diagram shown in Figure 2. The Extracting Text phase is in charge of extracting the textual content of each article available in the WCMS. The content has then to undergo a normalization sub-phase in which non-textual tokens are eliminated (i.e., operators, special symbols, numbers, html tags, 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 two characters are not considered. A stemming algorithm [25] is also used to reduce inflected or derived terms to their stem. The terms contained in a stop word list are removed as well.

The pre-processed text is then used to get the $A$ term-by-document matrix (i.e., the object Term-by-Document Matrix) and the list of the terms of all the articles (i.e., the object Dictionary). A generic entry $a_{ij}$ of this matrix denotes the number of times that the $i$-th term in the $j$-th document appears. For the weight associated to each pair $<\text{term}, \text{document}>$ we used the term frequency–inverse document frequency, also known as $\text{tf}_idf$, according to the chosen local (i.e., $lw$) and global (i.e., $gw$) weighting scheme, which is defined as:

$$
\text{tf}_idf(A) = lw \cdot \log(A) \cdot gw \cdot \text{idf}(A)
$$

(1)

In particular for every term $t_i$ and document $d_j$ in $A$ the term frequency–inverse document frequency is defined by:

$$
\text{tf}_idf(A_{[t_i,d_j]}) = \log(a_{[t_i,d_j]}) + 1 \left\{ \ln \left| \frac{1}{d_j \cdot A_{[t_i,d_j] > 0}} \right| + 1 \right\}
$$

(2)
where the term $|\mathcal{D}|$ is the overall number of documents. The global weighting scheme in the previous equation is forwarded to the weighting step of the folding-in phase.

The Dictionary object has to be persistent since it will be used in the subsequent interactions of the process by the Folding-in Extraction phase to build the object Term-Vector.

The normalized and weighted content is then used by the LSI phase to compute the concept space by adopting the LSI technique. This phase produces the persistent LSI Space object, which has to be persistent since it is used in the subsequent interactions of the process. In particular, it will be used by the Folding-in phase to add new articles to the preexisting latent semantic space.

The LSI technique has been originally developed to overcome the synonymy and polysemy problem occurring with VSM [19]. In particular, LSI explicitly considers dependencies among terms and among documents (articles 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 may be partially obscured by the used words in a document.

LSI is applied on a term-by-document matrix $A$, which is built on the content of the articles. 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 articles. An entry $a_{ij}$ of the term-by-document matrix $A$ (with rank $r$) represents a measure of the weight of the $i$-th term in the $j$-th article. On this matrix a Singular Value Decomposition (SVD) is applied to decompose it in the product of three matrices, $T \cdot S \cdot D^T$. The matrix $S$ 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 a subset of $k$ concepts (the space of the underlying 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 concept space) is an open issue and a number of strategies have been proposed in the past (e.g., percentage of number of terms, fixed number of factors, etc. [44]). In our approach, we calculate the number of singular values according to the Guttman-Kaiser criterion [18] to avoid any manual decision about the selection of $k$.

The matrices $T$, $S$ and $D$ obtained by applying SVD can be reduced to $k$, thus resulting into the truncated matrices $T_k$, $S_k$ and $D_k$.

To incrementally add new articles and the corresponding contents to preexisting latent semantic space the Folding-in phase is used, which takes as input the objects Term-Vector and LSI Space. Term-Vector is produced by the Folding-in Extraction phase, which extracts all the terms that are both in the new article and in the Dictionary object. The Folding-in phase uses the fold-in method [43] to avoid re-computing SVD each time a change is made to the term-by-document matrix. If on one side the folding-in method can be computed very fast, on the other side, its accuracy may degrade very quickly. In the latter case the term-by-document matrix has to be calculated considering all the documents to index. In fact, folding-in terms or documents is a much simpler alternative that uses an existing SVD to represent new information.
Moreover, the resulting vector $F$ (i.e., the Term-Vector object) can be weighted re-using the global weights of the first matrix $A$. In particular the term frequency-inverse document frequency used to weight the new matrix is:

$$tf_{-idf}(F) = lw \cdot \log tf(F) \cdot gw_{-idf}(A)$$  \hspace{1cm} (3)

To fold-in a new document (i.e., a vector $d$ whose dimension is $m \times 1$, where $m$ is the number of terms of the term-by-document matrix $A$) into an existing latent semantic space, a projection $d'$, of $d$ onto the span of the current term vectors (columns of $Tk$) is computed by $d' = dT \cdot Tk \cdot Sk^{-1}$. Finally, the latent space of the new document will be added to the original latent space by computing $Tk \cdot Sk \cdot d'$. The interested reader can find further details in [4] and [21].

The phase **Computing Similarity between Articles** is used to get the similarity between each pair of articles in the latent semantic space. Accordingly, the cosine between each pair of vectors can be used to get their similarity. In our case we defined a cosine based dissimilarity measure to get a quantitative indication of how the content of the articles is different:

$$d(i, j) = \frac{1-\cos(V_i, V_j)}{\max_{v_l, v_m}(1-\cos(V_l, V_m))}$$  \hspace{1cm} (4)
where \( i \) and \( j \) represent the articles and \( V_i \) and \( V_j \) the corresponding vectors in the \( k \) space \( W \), namely the latent semantic space of a given website. This measure assumes values ranging from 0 (when the content of two articles is the same) to 1 (when they have a different content). Let us note that the measure \( d \) 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 [28].

The output of Computing Similarity between Articles is a matrix (i.e., Articles Similarity Matrix) containing all the similarity values between each pair of articles. Finally, this matrix is used to group articles that are similar at the content level using the fuzzy \( c \)-means clustering algorithm (i.e., the Grouping Articles phase).

### 3.2.2. Grouping articles

We adopted the fuzzy \( c \)-means clustering algorithm to group articles into \( c \) clusters. Similarly to the fuzzy logic, an article has a degree of belonging or membership to clusters (i.e., one or more clusters) rather than completely belonging to a single cluster. Compared to other fuzzy clustering methods, we used an extended version that accepts a dissimilarity matrix as input.

\( u_{iv} \) is the membership of the article \( i \) to the cluster \( v \) is non negative. Moreover, the sum of all the memberships of a given article \( i \) is 1. The clustering is carried out through an iterative optimization of the following objective function:

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{u_{iv}^{r} u_{jv}^{r} d(i, j)}{2 \sum_{j=1}^{n} u_{jv}^{r}}
\]

(5)

where \( n \) is the number of articles and \( i \) and \( j \) are all the possible pairs that can be obtained on these articles. Differently, the maximum number of cluster to identify is \( c \). In our case, \( c \) is equal to \( \lceil n/2 \rceil \) because we experimentally observed that it allowed us to identify neither too large nor too small clusters. On the other hand, \( r \) is the membership exponent used in the fit criterion and it is larger than 1 (i.e., it assumes value between 1 and \( \infty \)). In case \( r \) is close to 1, the clustering algorithm behavior is comparable with the crisper version of this algorithm (i.e., the standard \( k \)-means clustering algorithm), whereas, for larger value of \( r \), the fuzziness level of the clusters is higher. Finally, \( d(i, j) \) is the dissimilarity between the articles \( i \) and \( j \), as previously defined in the formula (4). The iterations will stop when:

\[
\max_{i,j=1..k} \left| u_{iv}^{(t+1)} - u_{iv}^{(t)} \right| < \epsilon
\]

(6)

The \( \epsilon \) value ranges from 0 and 1 and represents a termination criterion, whereas \( t \) indicates the iteration step. It is worth mentioning that the tuning values of the adopted clustering algorithm should be chosen according to the content of the articles to be grouped.
The output of the iterations is a matrix of membership \( u_{i=1..n, v=1..c} \). The clusters are defined using a threshold \( th = 1/c \) on the membership values. This threshold was chosen based on the obtained experimental results. In addition, such a choice for \( th \) reduced the number of parameters the user has to select. Successively, the algorithm groups the articles starting from each given article identifies a list containing articles with similar content. In particular, this list is computed through the union of the clusters to which the article participates.

The SNM associated to a particular article is built by linking the article to all the articles included in the clusters to which the given article belongs. Each link is associated with the measure of similarity calculated between the source and target articles.

4. An implementation of the approach: the Joomla! extension module

In order to provide support for our approach, we have implemented an extension module for Joomla!, a well-known and widely employed open source WCMS for publishing contents on the Web and intranets. Joomla! is written in PHP, stores data in a MySQL database, and includes features such as page caching, RSS feeds, printable versions of pages, and support for language internationalization. Joomla! easily allows modular extensions and integrations through plug-ins, modules, and components. Accordingly, new functionality can be added to a website without hacking the core code of Joomla!.

Among the supported extensions types, modules complement the content contained in a page. This was the rationale for developing a Joomla! extension module to integrate our proposal in this WCMS.

Figure 3 shows the UML Package Diagram of the layered architecture (i.e., the logical architecture) of the Joomla! extension module. As it might be clear, the architecture does not depend on the specific WCMS considered, and the module can be adapted to different WCMSs. The Data component represents the database of the WCMS.

![Figure 3. The layered architecture of the plug-in to introduce SNMs in websites developed with Joomla!](image)
This component is also in charge of managing the persistence of the links between pairs of articles. The Computing Dissimilarity component computes similarities between the articles using an LSI technique. LSI Engine is the component in charge of computing similarities among article while the component that groups articles according to their similarity is Grouping Articles. This component implements the fuzzy clustering algorithm used. The GUI component is mainly in charge of starting the clustering-based process and enabling the user to manually refine (if required) the identified groups of articles through the Modifying Articles sub-component. This component also enables the selection of articles and websites to be considered in the clustering. Finally, the GUI component represents the user interface of the plug-in. By means of this interface the owner of a website built with Joomla! can either start the indexing or re-indexing of the articles published in the WCMS, or can add a new article to the pre-existing latent semantic space. These steps are performed by the Update Strategy sub-component.

A screenshot of the interface for selecting the articles in the clustering and to generate the map is provided in Figure 4.

![Screenshot of the interface](image)

Figure 4. The user interface of the Joomla! plug-in for introducing SNMs in the website front-end.

The Map Layout module is in charge of displaying the map inside in the web pages of the WCMS. In particular, this module injects the map taking into account different pattern layout: Absolute positioning, Relative positioning and Floating positioning. The first pattern displays the SNM in a given position for all the web pages, whereas the second one links the SNM to the position and the state (i.e., hidden or visible) of another view module present in the web page. The latter layout pattern displays the map as an overlapping layer on the web page. The SNM is always visible and will follow the scrolling operations of the user.

Finally, it is also possible to customize several parameters of the proposed process and clustering algorithm. In particular, in the Configuration Manager, the editor can
change the configuration of the pre-processing phase (i.e., the use of a stemmer and/or
the language used for the articles) or the tuning values of the fuzzy c-means clustering
algorithm (see on the right hand side of Figure 4). It is worth mentioning that the user
selects the articles in the recovery process, so providing more flexibility to our proposal.

The tuning values of the fuzzy c-means clustering algorithm have been chosen to get
optimal results in terms of navigation capability of the website selected. In particular, we
selected \( r = 1.12 \) (i.e., the fuzzy exponent), while the maximum number of iteration was
fixed to 10000 and 1e-18 (i.e., fuzzy tolerance). In the case the maximum number of
iteration is reached, the clusters are provided as output without further refinements.
Future work will be devoted to identify a heuristic to quickly select these tuning values.
This part of our research is still in progress and represents the most challenging.

5. Evaluation

The feasibility and validity of our approach have been assessed on two real world Italian
websites, namely gazzettinobr\(^6\) and ballettodelsud\(^7\). The first website is a local news
website which counts more than 10K articles while the second is the website of an Italian
theatre and dancing company which counts around 1K pages. Both websites are
implemented with Joomla!.

Table 1 shows some descriptive statistics. At the date when we applied our approach the
gazzettinobr website counted around 10000 articles while ballettodelsud included about
500 articles. Our analysis of the gazzettinobr website involved 517 articles while for
ballettodelsud the analysis involved 339 of its articles. The selection of articles was
obtained in both cases automatically by restricting the application of our approach to the
main sections of the website.

In the case of gazzettinobr the 517 articles were grouped into 238 clusters (or SNMs),
which the larger cluster containing 16 articles, and with each cluster containing as 4.09
articles as average after 257 iterations of the fuzzy clustering algorithm. For the
ballettodelsud website we analyzed 339 articles which were grouped into 136 different
clusters, with the largest cluster containing 18 articles and the mean number of articles
within each clusters being 10.23. In this case, the number of iterations needed by the
fuzzy c-means clustering algorithm to group the articles was 185.

Table 1. Descriptive statistics for the analyzed websites.

<table>
<thead>
<tr>
<th></th>
<th>gazzettinobr</th>
<th>ballettodelsud</th>
</tr>
</thead>
<tbody>
<tr>
<td># of analyzed articles</td>
<td>517</td>
<td>339</td>
</tr>
<tr>
<td># of clusters</td>
<td>238</td>
<td>136</td>
</tr>
<tr>
<td># of iterations</td>
<td>257</td>
<td>185</td>
</tr>
<tr>
<td># of articles within the largest cluster</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td># of clusters with one article</td>
<td>52</td>
<td>32</td>
</tr>
<tr>
<td>Mean # of articles within each cluster</td>
<td>4.09</td>
<td>10.23</td>
</tr>
<tr>
<td># of manually verified clusters to evaluate precision</td>
<td>21</td>
<td>13</td>
</tr>
</tbody>
</table>

\(^6\) Il Gazzettino di Brindisi, www.ilgazzettinobr.it
\(^7\) Balletto del Sud, www.ballettodelsud.it
Figure 5 shows an example of the enhanced version of the pages obtained applying the approach to the ballettodelsud website. In particular, the developed module integrates in the bottom left-hand side of each page the SNM highlighted in blue. Each SNM presents a set of links connecting the current page to pages stored in the same web site that have been found similar (e.g., showing similar content) to it. In addition, each link reports the associated rank value obtained computing the cosine between the current page and the pointed one.

By examining SNMs recovered for a representative sample of articles for both the analyzed websites we observed that most of the proposed links featuring a similarity measure > 33% were in fact related to the main topic of the examined article. This threshold could be used to filter links to be shown by each SNM.
Figure 6 shows an example of the enhanced version of the pages obtained applying the approach to the gazzettinobr website. In particular, the developed module integrates in the right-hand side of each page the SNM.

To further and deeply evaluate the results produced by the approach we used the precision, a measure from the information retrieval theory [17]. This measure has been used to assess the correctness of the links within the analyzed SNMs. A reader may object that the only assessment of the precision could provide only a partial evaluation of the approach. This is true, in fact, the completeness of the identified SNMs should be also assessed using a special conceived measure, e.g. recall. Unfortunately, this is not possible in practice because the computation of the precision and recall on the entire set of articles analyzed for the two websites used would need several man/hours. As an example, verifying the correctness of all the possible links between the articles of the ballettodelsud website would require performing 57,291 comparisons between all the pairs of articles (339*338/2). If 2 minutes are needed to discriminate if a link between two articles is correct or not, the total effort to compute the recall just for the ballettodelsud website only would be 1910 man/hours (about 1 year working 40 hours in a week).

For the above reasons, in our study, we considered the correctness of the recovered SNMs, without applying fold-in. We computed the precision measure by analyzing the groups of similar articles identified by the proposed approach. We involved three experts,
who were in the development team of neither the Joomla! plug-in nor in the websites analyzed here. These experts worked to identify articles with similar content and iterated until an agreement was reached. Let us note that the experts considered a pair of articles to be similar at the content level when: i. they shared a similar semantics, ii. used nearly the same terms, and iii. when a link among them could be meaningful from the website end-user point of view.

Let $A$ be the set of pairs of similar articles identified by the tool and $B$ the set of actual pairs of similar articles; we have defined the precision as follows:

$$\text{precision} = \frac{|A \cap B|}{|A|} \%$$

(7)

The precision measure represents the percentage of similar articles (i.e., links between similar articles) identified by the tool that is actually correct. This measure assumes values ranging from 0 (when all the articles found similar by the tool are instead different, i.e., all the links proposed by the SNMs are wrong) to 100 (when all the articles found to be similar by the tool are actually so, i.e., all the links proposed by the generated SNMs are correct). Similarly to [3], we decided to estimate the precision by randomly selecting 10% of the total SNMs (i.e., clusters of articles) generated for each of the analyzed websites (thus, 24 clusters for gazzettinobr and 13 for ballettodelsud) and asking the experts to check similarity of their included articles. The rationale for making this choice relies on the fact that it is also expensive to compute the precision on all the 136+238 clusters. In our empirical evaluation, considering the mean number of articles per cluster (4.75 for gazzettinobr and 10.23 for ballettodelsud), the computation of the precision would have required to evaluate the similarity of more than 7,500 pairs of articles for each expert (4*3/2*238=1428 pairs for gazzettinobr and 10*9/2*136=6120 pairs for ballettodelsud). We verified that the approach produces good results in terms of precision on the considered websites, namely 58% for gazzettinobr and 60% for ballettodelsud. In particular, the correctness of the proposed navigation links confirms the validity of the approach.

To complete our experimental assessment, we also considered the time needed to compute the similarity among articles and to group them using or not using fold-in (see Table 2). In particular, we have conducted two experimental runs with the ballettodelsud website. In the first run, we performed the overall process to generate the SNMs using the 339 articles and we folded-in a new document with a size ranged from 2Kb to 19Kb. In the latter, we recomputed the same experiment considering only the Computing Similarity phase. We have observed that on the average, the folding-in method provides an improvement of 49% in term of time saving considering the overall process and of 56% considering only the Computing Similarity phase. The performances of the folding-in method seem to be independent form the size of the documents added.
Table 2. Performance of the folding-in method

<table>
<thead>
<tr>
<th>Process</th>
<th>Number of pages</th>
<th>Size (Kb)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Overall process</td>
<td>339</td>
<td>736</td>
<td>170.62</td>
</tr>
<tr>
<td>Overall process with fold-in</td>
<td>1</td>
<td>2-19</td>
<td>104.46*</td>
</tr>
<tr>
<td>2 Computing Similarity</td>
<td>339</td>
<td>736</td>
<td>144.86</td>
</tr>
<tr>
<td>Computing Similarity with fold-in</td>
<td>1</td>
<td>2-19</td>
<td>66.49*</td>
</tr>
</tbody>
</table>

*Average time.

5.1. Threats to validity

Despite our efforts to mitigate as many threats to validity as possible, some are unavoidable. To comprehend the strengths and limitations of the empirical evaluation of our approach, the threats that could affect the results and their generalization are presented and discussed here.

Our evaluation is based on a subset of the whole set of the links of the two studied websites. Using the whole set could lead to different results. However, the number of links analyzed and the way in which they were selected allow us to believe that there should be not a huge difference in the results on the whole set of links and those presented in this paper.

Further threats concern the validity of the approach used to classify the links in a SNM as correct or incorrect. Public datasets are not available and then the approach used is the only possible. Furthermore, we did not consider the subjective satisfaction of actual users of the two websites analyzed. We plan to conduct special conceived investigations to evaluate the benefits and the satisfaction of the users with the improved navigation structure achieved by applying our proposal. In particular, we will design and conduct empirical investigations (e.g., controlled experiments, interviews, and surveys) in academic and professional contexts.

The websites we used in our empirical study were chose primarily because of their availability. We cannot estimate the effect of their attributes, such as, domain, development process, etc., on the results. This point could be subject of future work. Another possible threat to the validity of the results concerned the language of the web site content. The use of web site implemented in different languages (e.g., English) is subject of future work.

Finally, we did not perform any comparison of the approach implemented in our technology with other IR and techniques and clustering algorithms. We did not do this comparison because of out of the scope of the work presented in this paper. Related work suggested that the exploited technologies are a viable solution (e.g., [16][36]).
6. Conclusions and future work

In this paper, we have proposed an approach to extend the navigation structure of websites built with WCMSs (which usually natively support mainly category-based access to contents) with automatically built SNMs. We presented the approach and its implementation for the Joomla! open source WCMS. The results of an empirical assessment on two real world websites were also reported and discussed.

A SNM offers a user visiting the website the possibility to navigate from each page to others featuring similar content. The approach relies on a process that first computes a dissimilarity measure between the articles published in the site using LSI, and then uses a fuzzy-clustering algorithm to group articles with similar content. The navigation structure of the website is then extended by including into each page the corresponding recovered SNM which shows links to pages the user may be interested to visit and the associated measure of similarity.

SNMs intended neither to replace the navigation structure of a website nor to modify its usability. In fact, the original web site implements a navigation model properly designed with one of the web engineering methods proposed in the literature. Conversely, they are intended to complement this navigation structure, making explicit the correlations (from the lexical point of view) that might exist between the different contents in the site. Similar considerations can be done with respect to the usability of the web site. Indeed, the usability of this web site might improve when navigation structure is complemented by SNMs. This point is subject of future and special conceived investigations.

Similarly to the indexing processes performed by search engines such as Google or FreeFind® , our approach uses an information retrieval technique (i.e., LSI). Differently from them, the navigation index provided by a SNM 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, this additional navigation index is obtained considering as “input” of the search query not a particular keyword or set of keywords, but the full text of the page that the user is visiting.

SNMs may be also useful to the editor of a website, using a WCMS for this purpose. In fact, s/he could use the retrieved SNMs to add explicit links between a given article and one or more found related to it with our approach. We aim to extend the module we have developed for Joomla! to include this feature.

As other future work, we plan to conduct empirical studies involving real users, practitioners, and developers to evaluate the benefits and the impact on navigability of complementing the navigation structure of websites relying on WCMSs, both in the frontend and the backend, with our SNMs.

References

29. OWL - Web Ontology Language, www.w3.org/TR/owl-features
31. RDF - Resource Description Framework, www.w3.org/RDF
32. Related Anything, Joomla! module extension, online at http://extensions.joomla.org/extensions/news-display/related-items/5939
39. SPARQL Query Language for RDF, www.w3.org/TR/rdf-sparql-query