Intelligent Web Navigation

Inma Hernández
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
Avda. Reina Mercedes s/n
41012 Sevilla Spain
inmahernandez@us.es

Abstract
Virtual integration systems retrieve information according to the user’s interest. This information is retrieved from several web applications, but it is presented to the user uniformly, in an online process. Therefore, response time is a significant factor. An essential part of any information retrieval system is navigation through pages. Usually web pages contain a high number of links, some of them leading to interesting information, but most of them having other purposes, like advertising or internal site navigation. Traditional crawlers follow every link in each page, in order to analyze the target page, and classify it as interesting or irrelevant. This means having to retrieve, analyze and classify thousands of pages for every single site, which is a costly task. This problem can be solved with the combination of a web page classifier, to distinguish between interesting and irrelevant pages, and a link classifier, which automatically identifies links leading to interesting pages. This kind of navigation is more efficient and has a lower cost than traditional crawlers. Moreover, navigation model is automatically extracted from the site, instead of being handcrafted, reducing the supervision from the user.

Keywords: Navigation, Information Retrieval, Virtual Integration

1. MOTIVATION
Information retrieval obtains all documents relevant to a set of keywords, representing the user’s interests, while ignoring all non-related documents. Most IR systems rank the documents according to their relevance to the user, considering features like popularity or correlation to the keywords.

There are two different strategies to locate Web pages: those that are accessible just by following static links are called the Publicly Indexable Web (PIW) [32]. However, there are many pages that cannot be accessed this way, instead, they are behind HTML forms, that must be filled in and submitted. These pages constitute a part of the Internet known as the Deep Web, and they are our main focus of research.

Automated access to deep web information takes two different approaches, namely Virtual Integration (also known as Metasearch) and Surfacing (also known as Crawling) [22]. In virtual integration, users define their interests using keywords that can be as complex as needed, from simple terms to high-level structured queries, and as a response, the system retrieves information related to this keywords. This information is retrieved from many different sources, but it is presented uniformly to the users in a transparent way. This process is online, and therefore response time is an important issue. As opposed to virtual integration, surfacing is an off-line process that intends to collect all pages behind a web form by submitting pre-computed queries, and not taking into account the user’s specific requirements.

Virtual integration systems explore information inside several Web applications, and extract whatever information the user may consider relevant, using information extraction techniques. Therefore, information retrieval and information extraction are two complementary steps in the virtual integration process: the former retrieves all the relevant pages, and the latter extracts required information from these pages.

An essential part of any information retrieval system is the process performed to navigate pages, analyse them, and use the information contained in them to reach further relevant pages. Navigation can be approached in a blind way, e.g., following every possible link in every page. Traditional exhaustive crawlers take this approach, as its goal is to get as many pages as possible.
There is a main drawback to this approach: usually, web pages contain a large number of links, most of them non-relevant to the user (i.e., advertising or links to partner web sites). If the system has to follow all those links, a lot of time will be wasted, making it less efficient.

As a motivating example, Figure 1, shows a web page from a well known e-commerce site. Only three links (the ones marked with a rectangle) lead to relevant pages, while the others have other purposes, like advertising, internal site navigation, or suggestions for the user. As a result, the percentage of useless links is very high. We have observed than most web pages contain more than 70% of links that are useless for information retrieval purposes.

As opposed to blind navigation, other approaches include some criterion to decide which links to visit and which ones not, therefore reducing the number of irrelevant visited links. Focused crawlers, for instance, hold the criterion of avoiding pages not related to a certain topic (and therefore links leading to them). Relevancy criterions can be handcrafted by the user or automatically decided by the navigator, after analysing the web site, and extracting a navigation model. The latter is what we consider an automated intelligent navigator. In order to decide which pages and links are relevant, intelligent navigators include some reasoning process, usually in the form of a classifier.

The goal of our PhD Thesis is to analyze existing navigation techniques, to select those that are applicable in this context, improve and adapt them if possible, in order to achieve an automated intelligent navigator.

The rest of the article is structured as follows. Section 2 describes related work in the navigation and web page classification areas; Section 3 lists some of the conclusions extracted from the research and concludes the article.

2. RELATED WORK

Figure 2, shows a virtual integration process with intelligent navigation, starting with a set of user keywords which leads to a collection of response pages. Each one of these pages can be a Web page containing the answer to the query made by the user (detail page), or a paginated list of links to detail pages (hub page). In case the server lacks the answer to the query, the response page usually shows an informative message, and optionally some suggestions (no-results page). Finally, when some unexpected error arises, the response page may just contain some error description (error pages). Hence, a classifier is employed to distinguish between the different kinds of response pages (Web Page Classifier), and supports the navigator in discarding both no-results and error pages and retrieving detail pages. If a hub page is detected, it is further analysed in order to discriminate between the links it contains. A usual hub page has different types of links, including advertising, internal site navigation, links leading to detail pages, and...
links to the previous/following hub page to be recursively analysed (i.e., "More" or "Next" links). A link classifier (which we also call Navigator), makes this distinction in order to follow only relevant links. Finally, when detail pages have been identified and retrieved, an information extractor is used to extract relevant structured data.

There are several approaches to the Web classification problem, including content-based classifiers, structure-based classifiers, neighbour-based classifiers, and hybrid techniques. Regarding navigators, there are also some approaches. Some proposals include focused crawlers, recorders, user-defined navigators, and automated navigators. Next, we briefly describe and analyse the related work (summarised in Figure 3).

2.1. Web Page Classification

Web page classification has been extensively researched, and several techniques have been applied with successful experimental results. In general, we classify them according to the type and location of the classification features. There are three main trends in feature types: content-based, structure-based and hybrid classifiers. As for feature location, most approaches obtain features from the page to be classified, while others get them from neighbour pages.

Content-based classifiers ([20], [31] and [34]) categorize a web page according to the words and sentences it contains. This kind of classifiers group all pages within the same topic, assigning them the same class label. As for structure-based classifiers ([3], [6], [10], [16], [19], [33], [36] and [37]), the main feature used to classify pages is their physical organisation of contents, usually expressed in a tree-like data structure, like DOM Tree. Also, there are hybrid approaches, [12] and [23], which take into account both content and structural features. Nevertheless this distinction, there are some abstract classification techniques that sort out pages on the basis of any given feature, e.g., PEBL [39].

All the previous classifiers consider different kinds of features, but in all cases those features are extracted from the page to be classified. There are also classifiers that extract features from the neighbour pages, being the neighbour of a page another page that has a link to the first one. All these proposals are content-based, and usually rely on features such as the link anchor text, the paragraph text surrounding the anchor [15], the headers preceding the anchor, or a combination of these [18]. These techniques are a first approach to the link classification problem.

2.2. Navigation

Crawlers navigate pages by following every link they find. Therefore, the number of pages to be visited grows at a very fast pace. This is useful for certain tasks, like indexing pages for a searching engine, but for our retrieving goal it is not an efficient approach. One example of a traditional crawler is [32].
Recorders are also blind navigators, although they are more specific than crawlers, and less flexible. Anupam et al. [2] present WebVCR, a recording tool with a VCR-like interface to record user’s navigation steps through a web site and store them in a file, in order to replay them automatically in the future. Baumgartner et al. [8] proposes another recording system, based on Lixto, includes an embedded browser in which the user can perform navigation steps easily. Pan et al. [27] introduced the WARGO system, a wrapper creation tool, and a language to express navigation sequences called NSEQL. Sequences expressed in NSEQL are provided by the user to build a wrapper for information extraction, but the system has a browser-based interface to record the user navigation steps without requiring any programming skills from them. Often, it is necessary to leave some steps undefined until runtime, i.e., when a hub page is shown to the user, the number of clicks the system has to perform on a Next link depends on how many results the search returned. To solve this problem, the WARGO system endows the navigation language with a few extraction capabilities.

The previous proposals interact directly with the web browser interface, so they do not have to deal with problems like scripts, posting forms, having access to pages that require authentication, or navigating sites that keep information about the user, e.g., session IDs. They depend completely upon the user’s knowledge, who is responsible for defining the navigation sequences, providing the values to fill in the forms and for redefining the sequences whenever the target web site changes its structure or content. Recording navigation steps makes the navigation inflexible, and not using classifiers results in a more error-prone system.

Focused Crawling is an improvement over traditional crawlers, in the sense that it does include a Web Page Classifier. Focused Crawling ([1], [4], [5], [7], [13], [14], [25], [28], [29] and [30]) combines a traditional crawler with a topical classifier, keeping the focus on pages with topics that have some interest for the user, and therefore reducing the number of useless retrieved pages. However, focused crawlers improvement is limited to the reduction of useless pages retrieved, but the blind clicking is still an issue. Even when a high number of pages are discarded, they have to be visited and analyzed previously. Moreover, focused crawlers do not classify the functionality of a page, but its topic. When a crawler has submitted a form and gets a response page, both detail and hub pages are the answers to our query, and will probably have the same topic, but they should be processed differently.

User-Defined Navigation is learned by the system in a supervised way, i.e., the user demonstrates how to obtain the interesting information, and the system is able to generalise and acquire this knowledge. EzBuilder [11] is a virtual integration tool used to create information extraction procedures. In order to learn navigation for a web site, the user needs to give an example, submitting forms and navigating to several of the desired pages, and specifying the
Intelligent Web Navigation

structure of the data to be extracted. Then, the system builds a model of the user behaviour, that is employed to navigate automatically.

The previous proposal relies on the user, so that the system can learn a navigation model for each site. This model cannot be generalised to other sites automatically and the authors do not report any technique to update it.

Davulcu et. al. [17] presented a technique to express navigation sequences that relies on navigation maps. These are labelled directed graphs in which nodes represent web pages and edges represent actions that can be executed to navigate from one page to another (submitting a form or following a link). The application designer navigates the site, and his or her actions are captured and added to a graph (navigation map). In some cases, the designer needs to provide extended information about the form fields. Some structural changes in the sites can be handled by the system automatically, while other changes makes it necessary to update the maps manually.

Other proposals take a workflow-based approach to represent navigation sequences, although the creation of these sequences is defined by the user. Montoto et. al. [24] model navigation sequences as activities in a work flow diagram. They detect some structural page patterns which frequently occur in web sites. However, they do not use classifiers to automatically distinguish between them, and furthermore the sequence of navigation steps to reach these pages is defined by the user; actually their technique is an extension of [27].

[38] is an example of a semi-automated navigator based on a content classifier. It considers navigation as a sequence of pages, in which one of them is the goal, and the rest are intermediate. Each page belongs to a class, which is defined by a set of keywords, and a set of actions. When the system finds a page belonging to an intermediate class, these actions mark the steps to get to the next page in the sequence. However, if the system finds a page belonging to a goal class, the actions are extraction rules, in order to extract desired information.

Automated navigators are similar to the former proposals, except for the fact that navigation patterns are automatically extracted from web sites, instead of learnt from the user. Liddle et. al. [21] propose a crawling tool, in which forms are submitted, but no specific information is filled in, in order to retrieve as many results as possible. Once the form is submitted, the system is able to automatically distinguish between an indexed list of results, a complete list of results, an informative page with a no-result message, and an error page. When an hub page is detected because of the presence of a Next link, this link is followed iteratively until the last page (or a sufficiently large number of pages is found), and all the pages retrieved are concatenated in a single result page.

This proposal is very simple, since it is only applicable if we wish to retrieve the summarised information contained in hub pages, but not the extended information in detail pages. Furthermore, this tool retrieves all pages, regardless of their relevance to the user’s keywords.

For Palmieri et. al., [26], navigation patterns are a graph-based description of the different stages of a user navigation in a web site. They state that most web sites are designed according to the same navigation patterns, e.g., a start page with a link to an advanced search form, which is submitted to obtain a response page. They consider both HTML submission methods, POST and GET, which are expressed as different patterns. Once the form has been submitted, this approach analyses the response page to check if data objects of interest are present, in which case they consider it a hub page. Every hub page has a link leading to the next hub page, until all results are exposed to the user. Some heuristics are presented in order to detect and deal with these links.

However, this approach does not allow for the fact that that hub pages may only contain a brief summary of the data object and a link to a detail page. This proposal limits the possible navigation patterns to a couple of handmade samples, and therefore is not applicable to all web sites.

Vidal et. al. [36], considers that navigation patterns are automatically detected sequences of regular expression URLs that lead to relevant pages. This proposal receives a sample page, which represents the class of pages considered relevant, and it returns a navigation pattern, composed of the sequence of regular expressions that describe URLs that lead to the largest number of relevant pages. Just like the former approach, forms are submitted automatically, and
Intelligent Web Navigation

the response page is analysed. If the page belongs to the same class as the one given as example (using a structural classifier), it is considered a relevant page, and it is then retrieved. Small pages are considered to be error pages and they are not processed. Whenever a form is detected, it is submitted again, and process continues recursively. If a large number of links are detected, it is considered a hub page, and a further analysis is needed to select which links will be followed. Links are grouped by their URL similarity, and only one link from each group is followed to check if it leads to a relevant page.

This proposal is semi-supervised, since the process is automated, but the user has to provide a sample detail page, aside from filling in the parameter values for every form submission. This navigator can only reach a small number of possible results, for two reasons: it does not keep all navigation patterns leading to relevant pages, but only the one that leads to the largest number of them, and besides, links leading to other hub pages are not considered, only links to detail pages, so only relevant pages referenced in the first hub page are retrieved. This navigator has to be combined with an iterator in order to retrieve all possible results. Another question to bear in mind is that navigation systems are ad-hoc. If the site changes its URL nomenclature, the system has to be trained again, thus requiring user intervention, to update navigation patterns.

This proposal has some technical problems to be solved, namely, the heuristic for the treatment of form pages can lead to an infinite loop, in cases in which the server includes the original form in every response page. It does not support form submission using the POST method. Finally, this proposal does not support web sites that keep user session information.

[9] is another automated navigator. It is similar to the former: it relies on a structural-based clustering of pages, and it requires a sample relevant page. All pages are grouped in clusters, according to their structure, and links between pages are analysed in order to find relationships between clusters. Whenever a cluster has a high number of outgoing links to the cluster containing the sample, it is considered a hub cluster. This proposal is not designed to crawl pages behind forms, and therefore should be adapted in order to retrieve pages from the deep web.

3. CONCLUSIONS AND RESEARCH QUESTIONS

The main research question posed in this paper is an intelligent navigation system for virtual integration, as opposed to the blind navigation developed by most of the previous works. Our focus is on virtual integration of information within the Deep Web. In this context, users specify their interests by means of keywords, and information is retrieved from several sources, regardless of the particular details of each web application. Response pages are analysed, navigating through their links and collecting only those pages that contains the required information.

As we mentioned before, virtual integration is an online process, hence information should be presented to the user in a reasonable response time. Therefore, these systems will indeed find benefits in the application of an intelligent navigator, which avoids visiting useless pages, reducing the cost and improving the efficiency of traditional crawling systems. Summarizing, these are some other research challenges concerning intelligent navigation:

1. Response Web pages have to be classified into relevant and irrelevant, and also into the different roles that they play, which determine how to automatically deal with them.
2. Links in hub pages have to be classified before clicking on them, to identify those that lead to relevant pages. Not only the link anchor text has to be considered, but also the whole context of the link.
3. Link and Web page classifiers in this context are better designed with a lazy execution, that is, classification model is built and updated progressively during the process.
4. It is desirable to develop a navigator with as few interaction from the user as possible. Therefore, learning is unsupervised, or at least, very little supervised. Also, it is interesting to make it as general as possible, not having to build an ad-hoc model for every site, but instead developing a general model that can adapt to most sites just by tuning some parameters. This seems a priori a complex task, however.
5. Advanced post-filtering of relevant detail pages is also a desirable feature. For example, when looking for products in an online store, a user may wish to retrieve only those products whose price lay within a certain range. This means that the navigator needs to be endowed with a lite extraction tool, able to extract from each result at least the necessary data to apply the filter.
6. It is necessary to create a standard data set to evaluate classification and navigation proposals. Most proposals pose some experimental results, measured in terms of precision, recall or some other equivalent metrics. However, very few of them use a well-known data set in their experiments. In most cases researchers collect their own set of pages by issuing queries to a search engine, or using a blind crawler. As the experiments are performed on different sets of pages, the experimental results cannot be compared. There are some well known data sets of already classified pages available for this kind of experiments, e.g., the WebKB collection\(^1\), the Reuters-21578\(^2\) news collection or the TEL-8 query interfaces collection \(^3\). [35] reports an attempt to collect a standard data set. Unfortunately, these collections are outdated and updating them is a costly task. We argue that more effort on archiving needs to be done so that new proposals can be compared from an empirical point of view.

Bibliography


\(^1\)WebKB: http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/
\(^2\)Reuters: http://www.daviddi Lewis.com/resources/testcollections/reuters21578/
\(^3\)TEL8: http://metaquerier.cs.uiuc.edu/repository/datasets/tel-8/


