XXX

X Y

Dipartimento di Informatica

Dottorato in Informatica XXIII ciclo

Università degli Studi di Bari "Aldo Moro"

Via E. Orabona, 4 - 70125 Bari, Italy

xyo@di.uniba.it

S.S.D.: INF/01

Supervisor: Prof. x y

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Science

XXXX Date to define

Credits

This dissertation was typeset using these open-source programs:

- TeXShop LaTeX Editor
  available at: `http://www.uoregon.edu/~koch/texshop/`
- MacTex Distribution
  available at: `http://www.tug.org/mactex/`

Thesis Supervisor

Prof. Giovanni Semeraro

Chairperson of the Supervisory Committee

Member of the Supervisory Committee

Member of the Supervisory Committee

Submitted *XXXX Date to define*

Contents

Contents iii

List of Figures v

List of Tables vii

Acknowledgments 1

Abstract 3

1 Introduction 5

1.1 Automated information gathering 5

1.2 Mining the Web 5

1.3 Structured Data in the Web 8

1.4 The role of Information Extraction 9

1.5 The role of information network analysis 12

1.6 Outline of Thesis 13

2 Structured data and Logical List 15

2.1 Introduction 15

2.2 Definitions and Problem Formulation 17

2.3 Methodology 19

2.3.1 List Extraction 19

2.3.2 Dominant List Identification 20

2.3.3 Logical List Discovery 21

2.4 Experiments 22

2.4.1 Results 22

2.5 Conclusions and Future Works 23

3 Website hierarchies **25**
3.1 Introduction 25
3.2 Related Work 27
3.3 Problem Definition 30
3.4 Methodology 30
3.4.1 Sequence dataset generation 30
3.4.2 Frequent sequences generation 31
3.4.3 Pruning 32

4 Urls embedding **33**

Bibliography **35**

List of Figures

2.1 An example of Amazon Web page 16
2.2 An example of Rendered Box Tree 18
2.3 An example of *logical list* for Amazon's products. 19

List of Tables

2.1 Discovered *logical list* elements for Web sites dataset. 24

Acknowledgments

Abstract

Chapter 1

Introduction

1.1 Automated information gathering

trovare strutture, anche per mezzo di algoritmi di data mining

1.2 Mining the Web

The World Wide Web is the largest and most widely known repository of hypertext. Hypertextual documents (also called web documents or web pages) contain information about every topic and every real-world entity, authored and edited by millions of people and written in hundreds of languages. Moreover, hyperlinks creating connections between web pages make the Web as the largest and connected information network.

The Web has many unique characteristics which make Web Mining, defined as the automatic discovery of interesting and valuable information from web sources, an interesting and challenging task [Liu06]. In the follow will be analyzed the main characteristics:

- **Dimension**. The amount of data / information on the Web is huge and still growing. The Web is the first medium where the number of information producers, responsible for building without censure new facts, ideas, and opinions, is the same magnitude order of information consumers.
- **Dynamicity**. Each second thousands of web pages are created, destroyed and modified. This make the Web a dynamic information network, where the information structure and the information content change frequently. Keeping up with these changes and monitoring them are important issues for many applications.

- **Heterogeneity**. The Web heterogeneity depends both from web pages' format and from the writing style. In the first case, the heterogeneity is due by the fact that in the Web does not exist a standard format for web pages which can be classified in three categories [CKGS06]: i) unstructured pages; ii) structured pages; iii) semi-structured pages.
  Unstructured pages, also called free-text documents, are written in natural language. No structure can be found, and only information extraction (IE) techniques can be applied with a certain degree of confidence.
  Structured pages are normally obtained from a structured data source, e.g. a database, and data are published together with information on structure. The extraction of information is accomplished using simple techniques based on syntactic matching.
  Semi-structured pages are in an intermediate position between unstructured and structured pages. These documents possess anyway a kind of structure which is enclose in free-text. Extraction techniques are often based on the presence of regular patterns as HTML tags.
  In the last case, the heterogeneity is due by the fact that web page are created by millions of people having different culture, skills, language, etc.. This means that web pages may present the same or similar information using completely different words and/or formats. This makes extraction and integration of information from multiple pages a challenging problem.

- **Connection**. The Web is generally represented as an information network where nodes are web pages and edges are hyperlinks. Hyperlinks existing among web pages within websites and across different websites have different roles and functionality. Within a site, hyperlinks serve as information organization mechanisms. Across different sites, hyperlinks represent implicit conveyance of authority to the target pages. In this case, links to authority pages are based on the assumption that as human we have trust in veracity of these kind of pages.

- **Noise**. The richness of web content has also made it progressively more difficult to leverage the value of information. Differently from other media, contents publication in the Web does not require editorship and approval from some authority. This unregulated atmosphere contributes not only to the big volume and wide diversity of information, yet also to the presence of low quality, misleading, erroneous, and redundant data. Moreover, noise comes from another source. A typical Web page contains many pieces of information, e.g. the main content of the page, navigation links, advertise-

ments, copyright notices, privacy policies etc. For particular applications, only a part of information is useful (e.g. main content) while the rest is considered noise.

- **Virtual society**. The Web can be consider a large social network where people can communicate and influence other people. In fact, the Web is not only about data, information and services, but also about interaction among people, organization and automated systems.

The previous characteristics present both challenges and opportunities for extraction and mining of information and knowledge from the Web.
The aim of the Web Mining is to discover useful information or knowledge from the *web hyperlink structure*, *page content*, and *usage data*. Although it borrows heavily from Information Retrieval, Machine Learning, Statistics, etc., the characteristics of the Web make methods belonging to these fields non directly applicable on web pages or websites. Web Mining algorithms can be classified in three main categories based on the type of data used for the mining process:

- **Web Structure Mining**. It extracts previously unknown relationships among web pages analyzing the hyperlinks structure of the Web (also called web graph). Examples of web structure mining algorithms are clustering of connected web pages having a similar template [](Blanco), discovering of authoritative web pages [] [](inserire riferimento pageRank and HITS), websites hierarchies, etc.. Traditional Data Mining does not perform such tasks because there is usually no link structure in a relational table.

- **Web content mining**. It extracts or mines useful information or knowledge from web page contents. It belong to this group algorithms to automatically classify and cluster web pages according to their topics, to extract useful data discovering patterns in web pages such as descriptions of products, postings of forums, product listing, etc.. Although these algorithms can appear similar to traditional Data Mining or Text Mining algorithms, characteristics of web pages (e.g., presence of HTML tags) make algorithms belonging to these fields non directly applicable on web pages. An important sub-field of the Web Content Mining is the Web Information Extraction which goal is to extract structured data from web pages and map these data in relational tables(see Sec. 1.4).

- **Web usage mining**. It refers to the automatic discovery and analysis of patterns in clickstream and associated data collected or generated as a

result of user interactions with web resources on one or more websites.The discovered patterns are usually represented as collections of pages, objects, or resources that are frequently accessed by groups of users with common needs or interests. Web usage mining applies many data mining algorithms on web logs data properly collected and pre-processed. The major application areas for Web usage mining are personalization, system improvement, site modification, business intelligence, and usage characterization [SCDT00].

1.3 Structured Data in the Web

quali sono queste strutture

A large amount of information from the Web is represent in form of semi-structured data, that is a combination of unstructured text with data having a structure or a schema [AGM02]. Structured data contained in web pages are typically **data records** generated dynamically from an underlying structured source like a relational database (e.g., product listing of a Amazon web page) or from a static template (e.g., menu, navbar, etc.).

Extracting such data records is useful in several application domains because it enables us to obtain and integrate data from multiple sources (websites and web pages) to provide value-added services, e.g., customizable web information gathering, comparative shopping, meta-search, etc. With more and more companies and organizations disseminating information on the Web, the ability to extract such data from web pages is becoming increasingly important.

Differently from traditional textual documents, structured data in web pages are enriched by hyperlinks that enable information to be splitted in multiple and interdependent web pages. These hyperlinks can be used to identify collections real-world entities (e.g., web pages of courses, professors, products, news, etc) and relationships among entities. Figure **??** shows an Amazon web page returned for the query *Computer*. In such page it is possible identify several groups of structural data. In particular, links in structured data that describe the product listing allow us to obtain key information related to real-world entities (e.g., computers) while links contained in the other structured data allow us to navigate the organization of the website and identify meaningful connections among objects (e.g., identify computers belongs to the same brand).

Since web documents are neither well structured such as database nor completely unstructured such as pure textual documents, traditional Data Mining or Text Mining techniques can not be directly applied.

In the first case, Data Mining techniques are based on the assumption that data used to learn models share a common schema having well defined tables, attributes (columns), tuples (rows), keys, and constraints. Moreover, such data should be independent among them. Web pages break this assumption because they contain heterogeneous data and their hyperlinks define interdependence relationships. Moreover, web pages are codified in HTML a language markup that, differently from other language markups used to store data such as XML, was projected just for data rendering. From this reason, the Web can be considered a modern legacy system, since such a large body of data cannot be easily accessed and manipulated.

In the second case, Text Mining techniques fail to learn accurate models on web pages because they require collection of documents written with consistent styles (e.g., news articles) and are not able to handle complex information with elements possessing various semantic roles (e.g., a navigation menu, main content, calendar, table, and logotype) and providing different functionalities (e.g., a link, button, and element with drag-and-drop function) [QD09]. In fact, differently from textual documents, web documents have multiple representations which provide different information. One is the text representation written in HTML; the other is the visual representation rendered by a web browser. Text Mining algorithms focus on the text representation while ignore the visual information.

Consequently, there is a strong need in the computer science field of creating techniques and approaches that, using textual, structural, and visual information of web pages, are able to extract schema from structured data and align the data using that schema. Then the goal of Web Information Extraction is that to transform the Web from a legacy system to the biggest, structured, and easily accessible database.

1.4 The role of Information Extraction

introduzione a IE, trovare strutture nascoste nelle pagine web (vedi anche abstract dakkar) e.s. strutture nelle tabelle strutture nelle liste
Information Extraction born originally as a natural language processing task used to extract relevant information both from structured text with tabular information and from free text such as news articles [Eik99]. Goal of Information Extraction is to identify patterns involving syntactic relationships between words or semantic classes of words. Extracted patterns can be used generate a relation of k-tuple (where k is the number of attributes in a record) or a complex

object with hierarchically organized data [CKGS06]. An application example is to extract from a terrorist attack article key information about perpetrators, their affiliation, victims, location, etc.

With the growth of the Web researches tried to apply information extraction techniques on web pages. As said in the previous section, web pages differ from textual documents for several aspects, such as absence grammatical structures, presence of hyperlinks, etc. For this reason, traditional information extraction tools which use linguistic knowledge to extract key information are not suitable for web pages. For example, most of structured data in web pages are rendered using regular HTML tags patterns and information are splitted in interconnected web pages. For these web pages the analysis of regularity on their visual and structural representation and the analysis of the graph behind the website can be used for extracting structured data more accurately and efficiently than natural language processing methods.

Information extraction systems on the Web require to solve five distinct problems:

- **Navigation problem**: finding target web pages, i.e., pages containing data to extract in a website following hyperlinks. Websites, especially data intensive websites, contain both target pages and navigational pages (i.e., web pages contain hyperlinks to target pages or to other navigational pages). A system for extracting structured data should explore web pages in a way to minimize the search space to target pages and as less navigation pages as possible.

- **Data extraction problem**: extracting relevant data records from web pages.

- **Schema synthesis problem**: generating schema behind extracted data-records. In this case solutions are needed to extract and align attributes from data records. These solutions should be able to handle missing values, values embedded in plain text paragraphs, values hierarchically organized, etc.

- **Data mapping**: aggregating data from several websites. This requires data be homogenized since different websites can follow different conventions for naming things, for expressing measured units (e.g. currency, weight of a product, etc.), etc.. Mapping discrete values (e.g. company names) into a standard format and transforms measured values in a common unit improves the quality of the extracted data.

- **Data integration**: merging data from separate web pages. Some websites, especially websites with a large amount of data (e.g. Ebay, Amazon, etc.), split structured data on multiple web pages for avoiding to overload a single page with too much information (e.g., splitting the products listing using the pagination list). Other times information about a single data record are splitted on multiple web pages (e.g., reviews of a product are visualized in a web page different from the web page containing main information about the product self). Large scale programs able to fetch tens of thousands of web pages per second are called *crawler, spiders, web robots, or bots*.

To extract structured data from the Web, several types of wrappers have been created. A wrapper can be defined as a program that extracts structured data of a particular information source and translates them into a relational form [Kus97]. Existing wrappers can be categorized in three main groups:

- **Manual wrapper**: it involves the writing of ad hoc code. Human programmers identify syntactic patterns to extract structured data analyzing source codes (i.e. HTML code) of web pages and understanding their structure. Although this approach is simple, it suffers several disadvantages due Web features. The first drawback is due by heterogeneity and dimension of the Web; manual wrappers are not scalable because websites and web pages with different structures require different wrappers. Moreover, the Web is a dynamic environment where web pages are created, destroyed and modified; wrappers can not automatically adapt to these changes since they are based on a specified grammar and manual maintenance has high cost.

- **wrapper induction**: it consists in the automatic wrappers construction based on inductive learning methods. The first wrappers were created around 1995-1996. In this case, supervised learning approaches are used to automatically generate rules which are then used to extract structured data from unseen web pages. Wrappers based on induction learning can be classified as *zero-order* or *first-order* depending on the inductive learning method used [Eik99]. Wrappers constructed using zero-order learning methods learn models in form of couples *attribute-values*, that is, from database point of view as disconnected relations. Differently from zero-order wrappers, first-order wrappers are able to learn models in form of first-order predicates which describe relations and associations between these relations.

The accuracy of learned rules, and consequently the accuracy of this type of wrappers, strongly depends both the number and the quality of examples (i.e. manually labeled pages and data records). Wrappers generated using supervised learning require a high cost for manual labeling of examples, for generating wrappers from websites with different structure, and for the maintenance in case changes in the websites structure.

- **automatic extraction**: it consists in the automatic wrappers construction based on unsupervised learning. Differently from the previous approaches it is not required the manual labeling effort. This makes wrappers scalable to a large amount of websites and easily maintainable. Modern wrappers are based on this approach.

1.5 The role of information network analysis

Web as Heterogeneus Information Network

Nowadays, most real phenomena can be represented as information networks, that is graphs where the nodes are real-world entities interconnected and interacting among them. Formally an information network can be defined as follow [SH13]:

Definition 1 ***Information Network:*** *is defined as a directed graph* $G = (V, E)$ *with an object type mapping function* $\tau : V \rightarrow A$ *and a link type mapping function* $\phi : E \rightarrow R$*, where each object* $v \in V$ *belongs to one particular object type* $\tau(v) \in A$ *, each link* $e \in E$ *belongs to a particular relation* $\phi(e) \in R$ *, and if two links belong to the same relation type, the two links share the same starting object type as well as the ending object type.*

Examples of information networks include social networks, the World Wide Web, product co-purchasing networks, biological networks, terrorist network and so on. In particular, the World Wide Web is one of the biggest, heterogeneous, semi-structured, and multidimensional information networks. In fact the Web is the largest repository of shared and public available documents which describe a large variety of real-world entities and are related to a complete range of topics, from product to financial, public-record, scientific, hobby-related, and government.

The analysis of information networks has gained extremely wide attentions nowadays from researchers in several fields such as computer science, social science, physics, economics, bioinformatics, and so on, with exciting discoveries and successful applications across all the disciplines. Goal of information network

analysis is to extract patterns or regularities in relationships among interacting units which represent the structure of the network. For example, the application Data Mining or Graph Mining techniques on information networks allows researches to discover latent structures of real-world entities, such as the topics and communities they are involved in, the roles the entities play in these topics and communities, and the relations they potentially have with each other.

Although several methods and approaches exist for extracting knowledge from the information networks, most of the existing studies are based on the assumption that networks are homogeneous. However, real information networks contain heterogeneous and structured data (e.g., relational database data) [SH13]. Although a single link in the network could be noisy, unreliable and sometimes misleading, valuable knowledge about the structure of networks can be discovered reliably analyzing massive connections between objects [KHY+08].

1.6 Outline of Thesis

Vedere pagina 16 http://www0.cs.ucl.ac.uk/staff/J.Zhu/thesis.pdf

Chapter 2

Structured data and Logical List

Definire in capitolo 1, **Web Page**, **data record**, **Web List**

2.1 Introduction

The Web contains a large amount of structured data, most of which represent the main content of web pages. For example, given a Amazon web page that describes a DVD product, provided information about the item, such as prize, title, reviewers, ads, etc. are organized in a structured way.

Although humans can easily understand how such information are structured (e.g., distinguish between the structured data and the rest of the web content), machines cannot extract this semantic from the HTML code. Nowadays, to solve this issue two main approaches are applied:

- Using data markups to encode machine-readable knowledge. These markups are not related to the formatting of data but they just make the metadata and text enclosed within the XHTML tags more meaningful to computers.
- Data Mining methods automatically extract structural data based on structural features, visual features or both.

Although using data markups makes web pages machine-readable, there are several reasons why the first solution can not be applied on all websites. First, data intensive websites such as Deep Web Databases (e.g. Amazon.com, Trulia.com) are not favorable to make accessible all their information assets. Another important reason is that enriching web pages with XHTML tags requires human

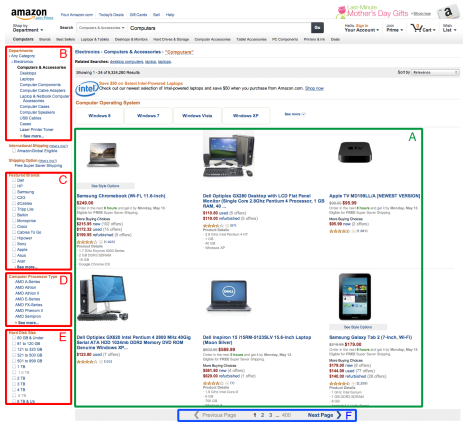

Figure 2.1: An example of Amazon Web page

efforts which do not involve direct earnings for organizations' websites.
Several methods to extract structured data have been presented in the literature. The first approaches were hand-crafted web scrapers based on regular expressions. An obvious disadvantage of these approaches is that different rule expressions need to be manually created for each website. Furthermore, an individual website may also change its structure or layout over time making this approach not scalable and in need of continuous maintenance.
Several supervised and unsupervised Data Mining methods were implemented later [LGZ04, MTH09, LGMK04, GBH+07, LMM10]. Although these methods close the challenge of automatic extraction of structured data from a web page, they fail to detect data record which span multiple Web pages. This is an open issue, because many web sites, especially data-intensive (e.g. Amazon, Trulia, AbeBooks,...), present their listings as *logical list*, that is, a list spanning multiple pages (*e.g.* computers, books, home listings)[1]. It is as if, each list represents a *view* of the same *logical list*. Similar to databases, where a *view* can represent a subset of the data contained in a table partitioned over a set of attributes, a *logical list* is split in multiple *views* (Web Pages) in order to avoid information overload and to facilitate users' navigation.

For example, Fig. 2.1 shows a Web page from Amazon.com that contains the results for the query "Computer". On this page, the boxes A, B, C, D, E, F are web lists. The list in the box A shows a *view* of the "Computers" products, that is the top six sorted by relevance, and F allows us to navigate to the other views of the products ordered by relevance. Thus navigating the links in F we

[1]The motivations behind this approach are as well technical (reducing bandwidth and latency), and non technical as avoiding information overload or maximizing page views.

can generate the *logical list* of the products for the query "Computer". Boxes B, C, D and E contain respectively the lists representing filters for "Department", "Featured Brands", "Computer Processor Type", and "Hard Disk Size", which are attributes of the *virtual* table "Computer". Moreover, the anchor-text links in boxes B, C, D and E stores valuable information which can be used to annotate data records, and thus to individuate new attributes. For example, the anchor-text links of web list C can be used to index data records based on "Computer brands". Traditionally, search engines use the proximity of terms on a page as a signal of relatedness; in this case the computer brand terms are highly related to some data records, even though they are distant.

Providing automated techniques for *logical list* extraction would be a significant advantage for data extraction and indexing services. Existing data record extraction methods [LGZ04, MTH09, GBH+07, LMM10] focus only in extracting *view* lists, while several commercial solutions[2] provide hand-coded rules to extract *logical lists*.

In this paper, we face this issue by proposing a novel unsupervised algorithm for automatic discovery and extraction of *logical lists* from the Web. Our method requires only one page containing a *view list*, and it is able to automatically extract the *logical list* containing the example *view list*. Moreover, during the process, it enriches the list's elements with the pair <*url, anchor-text*> used for the extraction task. We have validated our method on a several real websites, obtaining high effectiveness.

2.2 Definitions and Problem Formulation

In this section, we introduce a set of definitions we will use through the paper.

Definition 2 The Web Page Rendering *is the process of laying out a spatial position of all the text/images and other elements in a Web Page to be rendered.*

When an HTML document is rendered in a Web browser, the CSS2 visual formatting model [LB05] represents the elements of the document by rectangular boxes that are laid out one after the other or nested inside each other. By associating the document with a coordinate system whose origin is at the top-left corner, the spatial position of each text/image element on the Web page is fully determined by both the coordinates *(x,y)* of the top-left corner of its corresponding box, and the box's height and width.

[2] Lixto, Screen Scraper Studio, Mozenda Screen Scaper

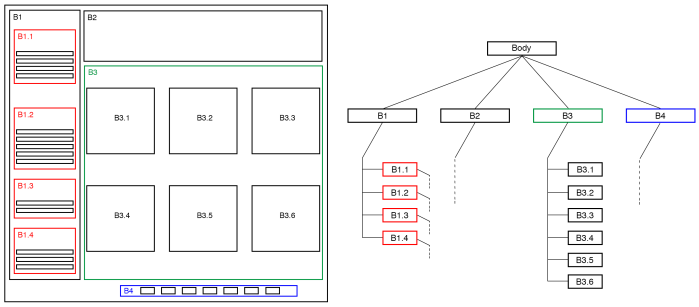

Figure 2.2: An example of Rendered Box Tree

Property 1 *The spatial positions of all text/image elements in a Web Page define the* Web Page Layout*.*

Property 2 *As defined in [FWB+11a], each* Web Page Layout *has a tree structure, called* Rendered Box Tree*, which reflects the hierarchical organization of HTML tags in the Web page.*

As we can see in Fig. 2.2, on the left there is the Web Page Layout of the Web page in Fig. 2.1. On the right, there is its Rendered Box Tree. The technical details of building Rendered Box Trees and their properties can be found in [FWB+11a]. Under the Web page layout model, and the Rendered Box Tree we can give the definition for Web lists.

Definition 3 ***Web List****: It is collection of two or more objects, under the same parent box and visually adjacent and aligned on a rendered web page. This alignment can occur via the x-axis (i.e. a vertical list), the y-axis (i.e. horizontal list), or in a tiled manner (i.e., aligned vertically and horizontally) [FWB+11b].*

For example the list A in Fig. 2.1 is a tiled list, while B is a vertical list and F is a horizontal list.

The list's elements can be called as *Data Records*. Similar to the concept of data records into database, data records into a web page are a set of similar and structured objects containing information. Typically, they are formatted using similar HTML tags (i.e. the same HTML structure).

Definition 4 ***Logical List****: It is a list whose Data Records are distributed on more then one Web Pages.*

An example is shown in Fig. 2.3, where the boxes A1 and A2 represent a part of a *logical list*.

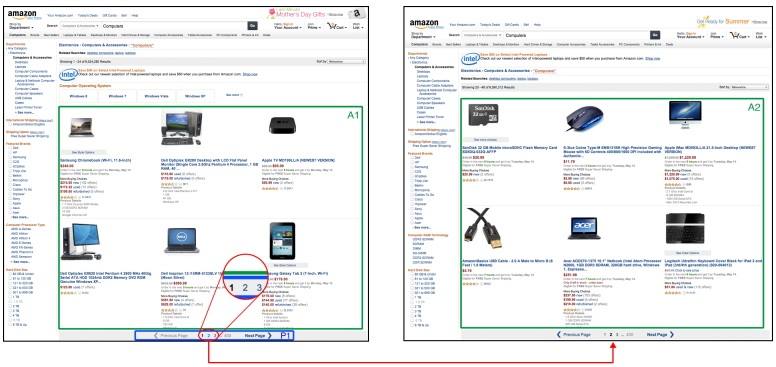

Figure 2.3: An example of *logical list* for Amazon's products.

Definition 5 ***View List****: It is a view of a logical list, whose Data Records are all contained in same Web page.*

For Example the list F in Fig. 2.1 is an example of a view list. In fact, it contains only some of data records belonging to its logical list (that is the *pagination list*).

Definition 6 ***Dominant List:*** *It is the view list of interest, containing data records from the logical list that we want to extract.*

For example the list A in Fig. 2.1 is the Dominant List for the given Web page.

2.3 Methodology

In this section we describe the methodology used for *logical list* extraction. The algorithm employs a three-step strategy. Let P a Web Page, it first extracts the set L^P of the lists contained in P; in the second step, it identifies the *dominant list* $l^P_{dom} \in L$; finally, it uses l^P_{dom} to discover the *logical list* LL which includes l^P_{dom} as sub-list. These steps are detailed in the following sub-sections.

2.3.1 List Extraction

Given a Web Page P as input, its lists $L = \{l_1, l_2, \ldots, l_n\}$ are extracted through running an improved version of HyLiEn [FWB+11b]. With respect to HyLiEn, we made several improvements. First, to render the Web pages we removed the dependency to the open source library *CSSBox*[3], because we found that this library was not able to correctly render several Web pages. We implemented a

[3] `http://cssbox.sourceforge.net`

WebKit wrapper, called *WebPageTraverser*[4], which is released as open source project. Given as input the url of a Web page P, *WebPageTraverser* outputs a JSON[5] representation of the P using the *rendered box tree model*. Second, to compute the similarity of two sub-trees in *rendered box tree* (see Prop. 2) we adopted the HTML tag distance, presented in [BLG11] instead of the string edit distance used by HyLiEn. Although, our current implementation uses the HyLiEn algorithm to obtain Web lists our solution is independent of any specific list-extraction algorithm. We used HyLiEn because it showed interesting result compared to the state of art algorithms for List Extraction [FWB+11b].

2.3.2 Dominant List Identification

Given a Web Page P and the set of list $L = \{l_1, l_2, \ldots, l_n\}$ extracted in the first step, we use three measures to identify the *dominant list* of P:

- **Centrality.** Given a list $l_i \in L$, the *centrality* of l_i w.r.t P is obtained by computing the Euclidean distance between the center of the parent-box of l_i and the center of root-box of P.
- **Area Ratio.** Given a list $l_i \in L$, the *area ratio* of l_i w.r.t P is the size of the box containing l_i divided the size of root-box of P.
- **Text-Tag Ratio.** Given a list $l_i \in L$, and let m the length of l_i, the *text-tag ratio* of l_i is computed as:

$$\frac{1}{m}\sum_{j=0}^{m}\frac{chars(l_i[j])}{tag(l_i[j])} \tag{2.1}$$

  where $tag(l_i[j])$ is the number of HTML tags contained in the j-th data record of l_i and $chars(l_i[j])$ is the total number of characters contained in $l_i[j]$. Before that the text-tag ratio is computed, *script* and *remark* tags are removed because this information should be not considered in the count of non-tag text.

In particular the *Dominant list* of P is the list with the highest sum of contributions:

$$\arg\max_{l_i \in L} \frac{\alpha_1}{centrality(l_i)} + \alpha_2 areaRatio(l_i) + \alpha_3 textTagRatio(l_i) \tag{2.2}$$

where $centrality(l_i), areaRatio(l_i)$ and $textTagRatio(l_i)$ are respectively the centrality measure, area ratio and text-tag ratio of a list l_i contained in L. $\alpha_1 = \alpha_2 = \alpha_3$ are set to 0.3 to give the same weight to each measure.

[4] `https://bitbucket.org/wheretolive/webpagetraverser`

[5] `http://www.json.org/`

Algorithm 1 `LogicalListDiscovery`

input : dominant list l^P_{dom}, set $L_- = \{L \setminus l^P_{dom}\}$
output: logical list LL

$LL = \{l^P_{dom}\}$ **forall the** $l \in L_-$ **do**
  forall the $u \in l$ **do**
    $L_u \leftarrow$ HyLiEn(u) L_u.filterSimilarity(l^P_{dom}, α) LL.add(L_u)
  end
end
$LL \leftarrow LL$.flatMap() $LL \leftarrow LL$.removeDuplicates() **return** LL

2.3.3 Logical List Discovery

Identified the dominant list l^P_{dom} of the Web Page P, the last step of the algorithm is to discover the logical list LL containing l^P_{dom}. This is done by taking advantage of the regularities of Web Sites. As described by Crescenzi et al. [CMM05], Web page links reflect the regularity of the web page structure. In other words, links that are grouped in collections with a uniform layout and presentation usually lead to similar pages. Link-based approaches are used in the literature for tasks strictly related to the one solved by our method. For instance, Lin et al. [LZW+10] used Web links to discover new attributes for web tables by exploring hyperlinks inside web tables. Lerman et al. [LGMK04] uses out-links to "detail web pages" in order to segment Web tables. In this paper, we successfully use links grouped as lists to navigate Web pages and to discover *logical lists*.

The algorithm 1 describes the approach used. It takes as input the dominant list l^P_{dom}, the minimum similarity threshold α, and the set of the lists L extracted from P. It iterates over all the lists in the set $L_- = \{L \setminus l^P_{dom}\}$ (line 1), and, for each url u in l_i it alternates, (i) the extraction of the set list L_u contained in the Web Page U having u as url (line 4) to, (ii) the filtering of all the lists in L_u which have a similarity with l^P_{dom} lower than α (line 6). At each iteration, all the lists resulting from step (ii) are added to LL (line 7). Finally, LL is flattened and all the duplicate elements are merged (lines 8-9). Moreover, during the process all the anchor text of url u are used as attributes to annotate the discovered *view* lists and are reported in the final *logical list* LL.

2.4 Experiments

In this section we presents the empirical evaluation of the proposed algorithm. We manually generated and verified a test dataset. In particular, for the experiment, we select 40 websites in different application domains (music shops, web journals, movies information, home listings, computer accessories, etc.) with list elements presented in different ways. For the deep-web databases, we performed a query for each of them and collected the first page of the results list, and for others we manually select a Web page. Table 1 shows in the first column the ground truth, that is, the number of data records which belong to the *logical list* to be extracted. The dataset is composed of 66.061 list elements extracted from 4405 Web pages. We rendered each Web page and we manually identified (i) *dominant list* and, (ii) following the out-links of the other lists in the pages we annotated *logical lists*. This task required around 7 days of 4 people.

To the best of our knowledge the task of *Logical List Discovery* is novel, and there are not any other methods to compare with. So, we evaluated the effectiveness of our algorithm by using *precision*, *recall* and *f-measure* metrics, computed over the number of *logical list* elements to be extracted (manually verified) w.r.t to the algorithm results. In particular, the precision is the measure of, how many of the extracted *view* lists belong to a *logical list*. The recall allows us to measure how many of the discovered *view* lists are true positive element of a *logical list*. We also included the *F-Measure* which is the weighted harmonic means of *precision* and *recall*. These metrics are evaluated counting how many data records of the *logical list* are found in the *view* lists.

$$precision = \frac{TP}{TP + FP},\ recall = \frac{TP}{TP + FN}, F{-}measure = \frac{2(precision \times recall)}{precision + recall} \tag{2.3}$$

2.4.1 Results

The execution of the algorithm requires two parameters which are empirically set to $\alpha = 0.6$ and $\beta = 50$. These parameters are need by the HyLiEn Algorithm. Our methods uses α during the *Logical List Discovery* step.

Table 1 presents the main results. The first column holds for each logical list the number of data records to extract. The second and the third columns contain the number of true positive and the number of false negatives data records. We do not plot the number of false positives, because our algorithm outputted always 0 false positives during the experiment evaluation. Finally, the fourth, fifth and sixth columns show the values for precision, recall and f-measure.

In general, the experimental results show that our algorithm is able to discover *logical lists* in a varying set of Web sites (that is, it is not domain dependent). Moreover, the quality of the results are not correlated to how the lists are rendered in Web pages (i.e. horizontal, vertical and tiled). In average, it achieves 100% for Precision, 95% for Recall and a F-Measure 97%. With respect to the ground truth, the algorithm does not extract any False Positive, and it outputs only 466 False Negatives. In general, it returns perfect results (100% precision and recall) for several kind of websites spanning different applications domain, but there are some of them which presents values for recall ranging from 81% and 91%. Considering "last.fm", which gave a recall equal to 81%, we found that the presentation of the data records is sometime quite different, because of the high variance in the number of the "similar to" tags (which are presented as HTML <a>) assigned to each listing. Analyzing other examples such as "IlSole24Ore.it" and "RealEstateSource.au" we found the same problem, that is, the presentation of the data records is quite variable across the Web pages, and so the HyLiEn algorithm sometimes misses some of the data records. Anyway we see that the proposed algorithms is effective is able to discover *logical lists* on different type of websites.

2.5 Conclusions and Future Works

In this chapter, it is presented a new method for *Logical List Extraction*. The proposed method solves the open issue of discover and extract lists which spans multiple Web pages. These *logical lists* are quite common in many websites, especially data-intensive, where their listings are split on multiples pages in order to avoid information overload and to facilitate users' navigation. However, the data stored in such *logical list* need to be automatically extracted to enable building services for market intelligence, synonyms discovery, question answering and data mashup. Experimental results show that the algorithm is extremely accurate and it is able to extract *logical lists* in a wide range of domains and websites with high precision and recall. Part of this future work will involve tasks such as indexing the Web based on lists and tables, answering queries from lists, and entity discovery and disambiguation using lists.

Website	*Ground*	*TP*	*FN*	*Precision*	*Recall*	*F-measure*
BariToday.it	904	904	0	100%	100%	100%
Subito.it	1000	1000	0	100%	100%	100%
GitHub.com	100	100	0	100%	100%	100%
TestoLegge.it	360	360	0	100%	100%	100%
Zoopla.co.uk	597	597	0	100%	100%	100%
FindAProperty.co.uk	60	60	0	100%	100%	100%
Savills.co.uk	232	232	0	100%	100%	100%
AutoTrader.co.uk	60	60	0	100%	100%	100%
EbayMotors.com	3925	3925	0	100%	100%	100%
Doogal.co.uk	38240	38240	0	100%	100%	100%
RealEstateSource.com	368	316	62	100%	85%	91%
AutoWeb.co.uk	180	180	0	100%	100%	100%
TechCrunch.com	434	422	12	100%	95%	98%
Landsend.com	1243	1243	0	100%	100%	100%
TMZ.com	300	300	0	100%	100%	100%
IlSole24Ore.it	510	445	65	100%	81%	86%
GoBari.it	350	340	10	100%	97%	98%
AGI.it	60	60	0	100%	100%	100%
BBCNews.co.uk	347	310	37	100%	89%	94%
milano.corriere.it	30	30	0	100%	100%	100%
torino.repubblica.it	70	68	2	100%	98%	99%
Ansa.it	1506	1479	27	100%	98%	99%
LeMonde.fr	445	418	27	100%	94%	97%
Time.com	377	377	0	100%	100%	100%
aur.ArchLinux.org	575	575	0	100%	100%	100%
Immobiliare.it	609	536	73	100%	86%	93%
bitbucket.org	130	130	0	100%	100%	100%
MyMovies.com	563	515	48	100%	92%	96%
Trulia.com	3300	3300	0	100%	100%	100%
YouTube.com	580	567	13	100%	98%	99%
FileStube.com	332	304	28	100%	91%	95%
Last.fm	60	41	19	100%	68%	81%
Bing.com	130	130	0	100%	100%	100%
addons.mozilla.org	984	939	45	100%	95%	97%
AutoScout24.com	840	840	0	100%	100%	100%
Facebook.com	2820	2820	0	100%	100%	100%
SlideShare.net	2037	2037	0	100%	100%	100%
Gazzetta.it	970	970	0	100%	100%	100%
ElPais.es	294	285	9	100%	98%	99%
StackOverflow	585	585	0	100%	100%	100%
Sums and Averages	66.527	66.061	466	100%	95%	97%

Table 2.1: Discovered *logical list* elements for Web sites dataset.

Chapter 3

Website hierarchies

vedi paper empirical study of web site navigation structure'impact on web site usability, ci sono diversi riferimenti interessanti.

3.1 Introduction

The research of information on the Web is characterized by two complementary paradigms: *searching* and *navigation* [Fur97, MSG97].

Keyword-based search is popular for quickly identifying pages containing specific information. Nowadays, most websites provide embedded search engines where given in input one or several key terms return an ordered list of web pages in descend ordering of relevance and accuracy. To express information needs to the search engine users have to think of appropriate key terms. This paradigm is useful when users are familiar with search domain (e.g., when they know the best terms to express their information needs and discriminate as much as possible their search domain from others domains). However, there are cases in which users do not know what they are looking for until the available options are presented or their information needs cannot be formulated in keywords [LLC05, OC03].

In the navigation paradigm, users start with the homepage or a web page found through a search engine or linked from another website, and then use the navigation systems (e.g. sitemaps, navbar, menu, etc.) provided by the website to find the desired information [FH07]. Moreover, studies reveal that users not randomly explore web pages in a website, but proceed in a top-down fashion, from a more general page to a detailed page using links of navigation systems with higher frequency compared to the other links of web graph [ADW01, ATM10, WL11, WBH12]. For this reason, the information ar-

chitecture of most websites, especially those with huge informative assets, is designed following both paradigms.

Goal of Information Architecture is to design websites in the way to codify information in a logical structure hierarchically organized, to create a better sense of orientation for users, facilitate the discovery and access of information, and enhance understanding the content of web pages [RM02]. Information Architecture is then crucial for the success of websites and their organizations.

Navigation systems are fundamental components of information architecture of websites, because they encourage the organization, labeling, and linking of website contents. Navigation systems can be classified in two categories: *embedding navigation systems* and *sitemaps* [Kal07].

Embedded navigation systems (e.g. navbar, menu, etc.) are typically wrapped around and infused within the content of the web site. They provide a local view of the website hierarchy, helping users to understand where they are and where they can go, and to easily find the needed information. However, sometimes users need of a global view (more or less detailed) of the website hierarchy.

Sitemaps, differently from embedding navigation systems, describe the global view of the logical structure of a website, allows the user to understand at a glance the global content of a website and contextual information such as relationships between web pages (e.g. relationships of super/sub topic). Using sitemaps users can jump directly to any page listed. Sitemaps are particular useful for websites with a great deal of contents and different types of visitors. In fact, in these cases embedded navigation systems are not able to codify every information needed for every visitor in every situation, yet a sitemap could help. Moreover, sitemaps can be used, in addition to navigation scopes, also to improve existing applications like search engines, to create new applications that integrate taxonomies in the presentation of search results [KMH13], or to cluster web pages having same type [LYHL10].

The sitemap construction is not a simple process especially for websites with a great amount of content and with a wide and deep logical hierarchy. In general, many websites have sitemaps which have been manually created by web designers at the deployment time. Differently from these static sitemaps, the logical hierarchy of most websites is updated frequently (e.g. by inserting, removing pages or adding new sections in the website). This means that such sitemaps could do not describe the correct and current structure of the website, becoming soon helpless and confusing for users.

Automatic generation of sitemaps solves this problem, helping both the web designer to track evolutions in the website's hierarchy and users to have al-

ways updated views of the content of the website. Moreover, analyzing the web log files and comparing them with the real sitemap of a website, it is possible to understand if users browse the site in ways that are different from the designer's expectations and view the website link structure differently from the designer [ADW01].

Several works face the problem of the automatic extraction or generation of sitemaps (also called hierarchies or taxonomies). They are usually based on text analysis, the hyperlink structure, the URL structure, heuristics, or a combination of these features [LNL04, YL09, LCC11, WBH12]. However, they are not able to extract the hierarchy efficiently and accurately. Goal of this chapter is to describe a novel approach which is able to extract the logical hierarchy of a website as designed by information architects [KMH13], that is the logical organization, in form of a tree structure, provided by the website's embedded navigational systems (which can be found on almost all websites). The idea is to discover frequent relationships among structured data, that is web lists, that describe hidden hierarchical relationships among web pages. Moreover, the returned hierarchy, based on the original development and structure of a website, is independent both from users behavior and words distribution in web pages.

3.2 Related Work

The problem of generating webisite hierarchies is part of more general Web Mining research. Although extracting website's sitemaps is an important task for many web applications, few studies focus on this problem. Existing state-of-art algorithms can be classified in three categories based on input data used.

The first category includes algorithms of Web Usage Mining. Goal of these algorithms is to identify patterns in web log files which describe how users frequently navigate the website. Most web usage mining techniques to extract sitemaps are based on Sequential Pattern Mining [BBA$^+$00, Mob07].

In general, sequential pattern mining algorithms on web logs can be classified in two sub-categories basing on the types of extracted sequences: *contiguous sequences* or *open sequences* [Mob07]. Contiguous sequences are a special form of sequential patterns in which the items appearing in the sequence must be adjacent with respect to the underlying ordering. They can be used to capture frequent navigational paths among user trails [SKS98, SF99]. In contrast, items appearing in open sequences, while preserving the underlying ordering, need not be adjacent, and thus they represent more general navigational patterns within the website. In [NM03] the authors compare models based on contiguous and

open sequences on web prediction. In particular, they claim that models based on contiguous sequences better perform on websites involving deeper structure and longer paths, while prediction models based on open sequences are better suited for personalization in websites with a higher degree of connectivity and shorter navigational depth (i.e., shallow logical hierarchies). However, in our knowledge there are not studies that analyze contiguous and open sequences in the context of sitemap generation. Sitemap generator algorithms based on web usage mining are characterized by several drawbacks, such as the creation of multiple sitemaps (i.e., one for each user profile) and the inclusion of just web pages having a user-specified number of visitors that reach them.

The second category includes algorithms of Web Content Mining. Hierarchies obtained by these methods represent the topical organization of web pages. Most of the existing techniques see the web pages as textual documents and they try to extract the hierarchical organization of web pages analyzing the words distribution on web pages and without considering other features, such as the hyperlinks structure. The problem with text-only hierarchy induction is that words often have multiple meanings and for heterogeneous websites it is difficult disambiguate words with multiple meaning and construct the proper taxonomy. In [WBH12] the authors propose a method to extract the document-topic hierarchies from websites that combine information from text and hyperlinks. The algorithm alternate two steps: 1)Random walking with restart from homepage and assigning an initial multinomial distribution to each node (i.e., document); 2) updating the multinomial distribution when the walker reaches a node. These steps are realized using Gibbs sampling algorithm. Several thousand Gibbs-iterations are usually required before a hierarchy emerges. The resulting hierarchy is obtained selecting for each node the most probable parent, in the way that nodes higher in the hierarchy contain more general terms, and nodes lower in the hierarchy contain terms more specific to their location.

The last category includes algorithms of Web Structure Mining. Web researchers and practitioners have used the hyperlink structure to organize web documents for many years. The PageRank and HITS algorithms are two representative link analysis algorithms and many link analysis algorithms are derived from both algorithms. The basic idea of link analysis algorithms is that if there is a hyperlink between two pages, then some semantic relation may exist between them [CMM05, LCC11, LFC+14]. Hyperlinks can be classified into four classes: *(1)*parent-to-child links, *(2)*upward links, *(3)* short-cuts, and *(4)*cross-topic links. Following this notion, hierarchy mining can be viewed as the process of selecting a sub set of hyperlinks indicating *parent-to-child* relationships to

connect the web pages under a tree structure [WBH12]. The simplest algorithm of sitemap generation is the breadth search. In the breadth search method, starting at the homepage, links are traversed level by level and each web page is put onto a conceptual level of the link hierarchy determined by the shortest path from the homepage to the page. The problem with this method is that the shortest path from the homepage to a page does not necessarily match parent-to-child relationships among pages in the path. In [KPT06, LCC11], authors proposed hierarchical topic segmentation of a website into topically cohesive regions that respect the hierarchical structure of the website. In particular, in [LCC11] tit is developed a system based on the HITS algorithm for the automatic generation of hierarchical sitemaps for websites. The idea is that to split web pages in blocks and identify those having high frequency and hub value which describe the sitemap. The accuracy of the method strongly depend from the task of blocks extraction. In fact, the algorithm require that blocks have the same structure on web pages. This assumption holds for blocks belonging to the highest levels of the hierarchy, but it is not true for deeper levels. In fact, in general navigation systems representing the layout template of web pages tend to stay in a constant position for enhancing consistency among web pages. However many websites change the the layout of secondary menus in the way to provide users a sense of progression through the website. For this type of websites the proposed algorithm is not able to extract the deeper levels of the hierarchy. Another drawback is due by the presence of false positive blocks that are included because recurrent (e.g., advertisements). In [YL09] a classifier is learned to extract parent-to-child relationships using several features such as URL structure, content and navigation features, information about anchors text and heuristics. Although the method is simple and results seem to be enough accurate, labeling examples require a lot of effort. Then the solution is not appropriate for an huge and heterogeneous network such as the Web. In [KN12] the author focus on the problem of extraction structured data such as menus to identify the main hierarchical structure and website boundaries analyzing cliques among in the web graph. As in [LCC11] the algorithm segment web pages of a given website in blocks and identify blocks that compose the maximal cliques as main menus of the website. Although the method is unsupervised and then suitable for heterogeneous websites, it is not able to extract the deeper levels of the hierarchy.

3.3 Problem Definition

To describe the methodology used to extract the hierarchy from a web graph we will give in the following some formal definition.

First, a website can be viewed as a direct graph $G = (V, E)$, where V is the set of web pages and E is the set of hyperlinks. In most case, the homepage h of a website representing the website's entry page allows the website to be viewed as a rooted directed graph.

Given a web graph $G = (V, E)$ rooted at h, a user threshold t and a cost function w, the problem of sitemap extraction is to find a tree $T = (V', E')$ rooted at h and having the following properties:

- $V' \subseteq V$;
- $E' \subseteq E$;
- $\forall e = (i, j) \in E' j \in webList(i)$, that is the url j is contained in some *web list* of the web page i (See Def. 3);
- $\forall e \in E',\ w(e) \geq t$;
- $T = \arg\max_{T_i} \Phi(T_i) = \sum_{e \in E'} w(e)$.

In the next section we will explain how calculate the cost function $w : E \to \mathbb{R}$ used to select among all possible trees T_i in G the sitemap T.

3.4 Methodology

In this section we describe the methodology used for sitemap extraction. The algorithm employs a three-step strategy. Let h the homepage of a website where we want extract the hidden sitemap, it first generate a sequence database SDB obtained on a stochastic process resembling random walks with restart (RWR) over the original document-graph; then in the second step an sequential pattern mining algorithm is applied to select frequent patterns in the way to obtain a simpler graph G'; finally a pruning method is applied to the partial solution obtained in the second step to select the best parent to each node in G' having multiple in-links in the way to obtain the sitemap T. These steps are detailed in the following sub-sections.

3.4.1 Sequence dataset generation

We begin with a web graph $G = (V, E)$ rooted at the homepage. In particular, we use the random walk theory on G to generate a database of short random

walks which are able to capture the graph structure.

In general, given a graph and a node as starting point, the random walker randomly chooses a neighbor of the given node and moves on it; then the process of choosing is repeated on this new node. The (random) sequence of selected nodes this way is a random walk on the graph. Several methods about random walks generation are been implemented in way to obtain random walks with different distributions and for different applications [].

In our algorithm we implement an evolution of *random walks with restart at home* proposed in []. As in [] we force a random walk to start and restart at the starting node, i.e. the homepage. Using this approach nodes closer to the homepage have higher probability to be reached than those at deeper positions. However, we limit the possible choices of the random walker on a k-th node v_k to a subset of its outlinks. In particular given the homepage, with probability $(1 - \alpha)$ the walker randomly walks to a connected web page whose url is contained in a web list of the current page or chooses to restart his walk at the homepage with probability α, where α is called the restart probability. Suppose for example that a random walker reaches at the $k - th$ wave the page having url u_k. Then the random walker can visit a random outlink of u_k in the set $L =$. This constraint is due by the assumption that the hierarchy is hidden in the navigation systems, which are rendered as structured data.

Algorithm 1 describes this process algorithmically. In particular given in input the sequence length $l \in \mathbb{N}$ and the database length $d \in \mathbb{N}$, the algorithm returns a sequence database SDB, composed by d random walks having length l.

3.4.2 Frequent sequences generation

Given a sequence database and and user threshold t, we apply a closed sequential pattern mining algorithm to extract all frequent sequences.

In sequential pattern mining, input data is a set of sequences, that is an ordered list of transactions, where each transaction is a set of literals, called itemset. The problem is to find all sequential patterns with a user-specified minimum support (or frequency), which is defined as the percentage of data-sequences that contain the pattern. Since the number of frequent sequences can be prohibitive, one can reduce the search space to closed sequential patterns, where a sequential pattern γ is closed if it has no proper super-sequence γ' with the same support.

To obtain more compact results, saving computational time and space costs we apply CloFAST [] a closed sequential pattern algorithm. CloFast, given in

input a sequence database returns a edge-weighted tree (T, w) of closed sequential patterns where each node is a frequent sequence and $w : E \to \mathbb{R}$ is a weight function. As defined in [] we identify two types of sequences: *contiguous sequences* and *non-contiguous sequences*. In the case of contiguous sequences, the cost function w is the *contiguous support*. In particular, given an edge $e = (\alpha, \beta)$ between the sequences $\alpha = \{a_1, a_2, ..., a_i\}$ and $\beta = \{a_1, a_2, ..., a_i, a_{i+1}\}$, $w(e)$ is the number of time the sequence β occurs in SDB and each item a_{j+1} is adjacent to a_j , with $1 \leq j \leq i$. In the case of non-contiguous sequences, the cost function w is the *support*, that is the number of time a sequence occurs in SDB without constraint among the adjacency of its items.

Although the CloFAST output is a tree it is possible identify multiple paths to reach an url u. That means there are two candidate paths that allow us to reach u in the hierarchy. For example, the Figure **??** shows that there are the paths u_0 and u_0, u_1 that reach the url u_2.

3.4.3 Pruning

The last task of the sitemap generation is the pruning process. In particular, since in the hierarchy there is a unique path for each node goal of this step is to select for each node in T the best path to reach T.

Chapter 4

Urls embedding

Bibliography

[ADW01] Corin R. Anderson, Pedro Domingos, and Daniel S. Weld. Adaptive web navigation for wireless devices. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, pages 879–884, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[AGM02] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. Technical Report 2002-40, Stanford InfoLab, July 2002.

[ATM10] Franck Amadieu, André Tricot, and Claudette Mariné. Interaction between prior knowledge and concept-map structure on hypertext comprehension, coherence of reading orders and disorientation. *Interact. Comput.*, 22(2):88–97, March 2010.

[BBA+00] Matthias Baumgarten, Alex G. Büchner, Sarabjot S. Anand, Maurice D. Mulvenna, and John G. Hughes. User-driven navigation pattern discovery from internet data. In *Revised Papers from the International Workshop on Web Usage Analysis and User Profiling*, WEBKDD '99, pages 74–91, London, UK, UK, 2000. Springer-Verlag.

[BLG11] Lidong Bing, Wai Lam, and Yuan Gu. Towards a unified solution: data record region detection and segmentation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 1265–1274, New York, NY, USA, 2011. ACM.

[CKGS06] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled F. Shaalan. A survey of web information extraction systems. *IEEE Trans. on Knowl. and Data Eng.*, 18(10):1411–1428, October 2006.

[CMM05] Valter Crescenzi, Paolo Merialdo, and Paolo Missier. Clustering web pages based on their structure. *Data Knowl. Eng.*, 54(3):279–299, September 2005.

[Eik99] Line Eikvil. Information extraction from world wide web - a survey, 1999.

[FH07] Xiang Fang and Clyde W. Holsapple. An empirical study of web site navigation structures' impacts on web site usability. *Decision Support Systems*, 43(2):476 – 491, 2007. Emerging Issues in Collaborative Commerce.

[Fur97] George W. Furnas. Effective view navigation. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, pages 367–374, New York, NY, USA, 1997. ACM.

[FWB+11a] Fabio Fumarola, Tim Weninger, Rick Barber, Donato Malerba, and Jiawei Han. Extracting general lists from web documents: A hybrid approach. In Kishan G. Mehrotra, Chilukuri K. Mohan, Jae C. Oh, Pramod K. Varshney, and Moonis Ali, editors, *IEA/AIE (1)*, volume 6703 of *Lecture Notes in Computer Science*, pages 285–294. Springer, 2011.

[FWB+11b] Fabio Fumarola, Tim Weninger, Rick Barber, Donato Malerba, and Jiawei Han. Hylien: a hybrid approach to general list extraction on the web. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *WWW (Companion Volume)*, pages 35–36. ACM, 2011.

[GBH+07] Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 71–80, New York, NY, USA, 2007. ACM.

[Kal07] James Kalbach. *Designing Web Navigation*. O'Reilly, first edition, 2007.

[KHY+08] Hillol Kargupta, Jiawei Han, Philip S. Yu, Rajeev Motwani, and Vipin Kumar. *Next Generation of Data Mining*. Chapman & Hall/CRC, 1 edition, 2008.

[KMH13] Matthias Keller, Patrick Mühlschlegel, and Hannes Hartenstein. Search result presentation: Supporting post-search navigation by integration of taxonomy data. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 1269–1274, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[KN12] Matthias Keller and Martin Nussbaumer. Menuminer: Revealing the information architecture of large web sites by analyzing maximal cliques. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 1025–1034, New York, NY, USA, 2012. ACM.

[KPT06] Ravi Kumar, Kunal Punera, and Andrew Tomkins. Hierarchical topic segmentation of websites. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 257–266, New York, NY, USA, 2006. ACM.

[Kus97] Nicholas Kushmerick. Wrapper induction for information extraction. In *Proc. IJCAI-97*, 1997.

[LB05] Hakon Wium Lie and Bert Bos. *Cascading Style Sheets: Designing for the Web (3rd Edition)*. Addison-Wesley Professional, 3 edition, 5 2005.

[LCC11] Shian-Hua Lin, Kuan-Pak Chu, and Chun-Ming Chiu. Automatic sitemaps generation: Exploring website structures using block extraction and hyperlink analysis. *Expert Systems with Applications*, 38(4):3944–3958, 2011.

[LFC+14] Pasqua Fabiana Lanotte, Fabio Fumarola, Michelangelo Ceci, Andrea Scarpino, MicheleDamiano Torelli, and Donato Malerba. Automatic extraction of logical web lists. In Troels Andreasen, Henning Christiansen, Juan-Carlos Cubero, and ZbigniewW. RaÅ›, editors, *Foundations of Intelligent Systems*, volume 8502 of *Lecture Notes in Computer Science*, pages 365–374. Springer International Publishing, 2014.

[LGMK04] Kristina Lerman, Lise Getoor, Steven Minton, and Craig Knoblock. Using the structure of web sites for automatic seg-

mentation of tables. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 119–130, New York, NY, USA, 2004. ACM.

[LGZ04] Bing Liu, Robert L. Grossman, and Yanhong Zhai. Mining web pages for data records. *IEEE Intelligent Systems*, 19(6):49–55, 2004.

[Liu06] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[LLC05] Uichin Lee, Zhenyu Liu, and Junghoo Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 391–400, New York, NY, USA, 2005. ACM.

[LMM10] Wei Liu, Xiaofeng Meng, and Weiyi Meng. Vide: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 22(3):447–460, 2010.

[LNL04] Zehua Liu, Wee Keong Ng, and Ee-Peng Lim. An automated algorithm for extracting website skeleton. In *Database Systems for Advanced Applications*, pages 799–811. Springer, 2004.

[LYHL10] Cindy Xide Lin, Yintao Yu, Jiawei Han, and Bing Liu. Hierarchical web-page clustering via in-page and cross-page link structures. In *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part II*, PAKDD'10, pages 222–229, Berlin, Heidelberg, 2010. Springer-Verlag.

[LZW+10] Cindy Xide Lin, Bo Zhao, Tim Weninger, Jiawei Han, and Bing Liu. Entity relation discovery from web tables and links. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1145–1146, New York, NY, USA, 2010. ACM.

[Mob07] Bamshad Mobasher. The adaptive web. chapter Data Mining for Web Personalization, pages 90–135. Springer-Verlag, Berlin, Heidelberg, 2007.

[MSG97] Udi Manber, Mike Smith, and Burra Gopal. Webglimpse: Combining browsing and searching. In *Proceedings of the Annual Confer-*

ence on USENIX Annual Technical Conference, ATEC '97, pages 15–15, Berkeley, CA, USA, 1997. USENIX Association.

[MTH09] Gengxin Miao, J Tatemura, and WP Hsiung. Extracting data records from the web using tag path clustering. *The World Wide Web Conference*, pages 981–990, 2009.

[NM03] Miki Nakagawa and Bamshad Mobasher. A hybrid web personalization model based on site connectivity. In *Proceedings of WebKDD*, pages 59–70, 2003.

[OC03] Christopher Olston and Ed H. Chi. Scenttrails: Integrating browsing and searching on the web. *ACM Trans. Comput.-Hum. Interact.*, 10(3):177–197, September 2003.

[QD09] Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2):12:1–12:31, February 2009.

[RM02] Louis Rosenfeld and Peter Morville. *Information Architecture for the World Wide Web*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2nd edition, 2002.

[SCDT00] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explor. Newsl.*, 1(2):12–23, January 2000.

[SF99] Myra Spiliopoulou and Lukas C Faulstich. Wum: a tool for web utilization analysis. In *The World Wide Web and Databases*, pages 184–203. Springer, 1999.

[SH13] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: A structural analysis approach. *SIGKDD Explor. Newsl.*, 14(2):20–28, April 2013.

[SKS98] Stuart Schechter, Murali Krishnan, and Michael D Smith. Using path profiles to predict http requests. *Computer Networks and ISDN Systems*, 30(1):457–467, 1998.

[WBH12] Tim Weninger, Yonatan Bisk, and Jiawei Han. Document-topic hierarchies from document graphs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 635–644, New York, NY, USA, 2012. ACM.

[WL11] Yao-Te Wang and Anthony J.T. Lee. Mining web navigation patterns with a path traversal graph. *Expert Syst. Appl.*, 38(6):7112–7122, June 2011.

[YL09] Christopher C. Yang and Nan Liu. Web site topic-hierarchy generation based on link structure. *J. Am. Soc. Inf. Sci. Technol.*, 60(3):495–508, March 2009.